Correctly Modeling the Diagonal Capacity in Escape Routing

Tan Yan and Martin D. F. Wong, *Fellow, IEEE*

***Abstract*—Escape routing for packages and printed circuit boards (PCBs) has been studied extensively in the past. Network flow is pervasively used to model this problem. However, none of the previous works correctly models the diagonal capacity, which is essential for 45° routing in most packages and PCBs. As a result, existing algorithms may either produce routing solutions that violate the diagonal capacity or fail to output a legal routing even though there exists one. In this paper, we propose a new network flow model that guarantees the correctness when diagonal capacity is taken into consideration. This model leads to the first optimal algorithm for escape routing. We also extend our model to handle missing pins.**

***Index Terms*—Diagonal capacity, escape routing, missing pin, network flow, package routing, printed circuit board (PCB) routing, 45° routing.**

I. Introduction

ESCAPE ROUTING is an important problem in package and printed circuit board (PCB) design. Its purpose is to route from specified pins in a pin array to the boundary of the array (see Fig. 1). It can be further classified into three categories.

1) Unordered escape is to route from pins inside one pin array to the boundary of the array without considering the pin ordering along the boundary. Previous works on this topic include [2]–[12].
2) Ordered escape also considers only one pin array. However, it requires the escape routing to conform to specified ordering along the boundary of the pin array. Previous works on this topic include [13]–[17].
3) Simultaneous escape considers escape routing of two pin arrays. The orderings of the escaped nets along the boundaries of the two pin arrays are required to match each other in order to provide a planar topology for later length-matching routing. Previous works on this topic include [18]–[21].

Manuscript received March 10, 2011; revised July 6, 2011; accepted August 21, 2011. Date of current version January 20, 2012. This work was supported in part by the National Science Foundation, under Grant CCF-1017516, and by a grant from the Fujitsu Laboratories. A preliminary version of this paper appeared in the Proceedings of DAC'09 [1]. This paper was recommended by Associate Editor D. Z. Pan.

T. Yan is with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: tanyan@synopsys.com).

M. D. F. Wong is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61821 USA (e-mail: mdfwong@illinois.edu).

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.2011.2169258

The three types of escape routing problems have different applications in package and PCB routing. In this paper, we focus on the unordered escape problem.

In the pin array, the design rules limit the number of wires between two orthogonally or diagonally adjacent pins. We call such constraints orthogonal capacity and diagonal capacity, respectively, or *O-cap* and *D-cap* for short (see Fig. 1). If we consider a *tile* of the pin array, which is the square formed by four adjacent pins, we can see that *O-cap* limits the number of wires passing through its four sides while *D-cap* limits the number of wires passing through its two diagonals.

Network flow is pervasively used to model this problem [2], [3], [6], [8]–[10], [12]. The idea is to view each routing path as a unit flow from the pin to the boundary. Since no ordering is specified, a flow solution always corresponds to some noncrossing routing. However, none of the previous network flow models correctly captures the diagonal capacity, which is essential for 45° routing in most packages and PCBs. The models in [8] and [10] simply ignore the diagonal capacity and may therefore result in design rule violations. On the other hand, the models [2], [3] set a limit on the number of wires inside a tile. Since tile capacity does not correctly reflect diagonal capacity, their models may miss the optimal solution. Detailed discussion about these models can be found in the next section. The other models are not correct either. Reference [9] used triangulation, which captures only one of the two diagonals in a tile. Therefore, its solution may still contain capacity violations on the other diagonal. Network of [6] only gives an upper bound estimation on the number of routable nets. The network in [12] is constructed by overlaying a 45° routing grid on top of an orthogonal routing grid. Even though it is claimed to be optimal in [12], we found that the network might have flow solutions that violate the spacing rule. (We will give such a counterexample in the next section.) In summary, none of the previous network flow models is able to correctly model the diagonal capacity.

Nonflow solutions were also proposed in some early works [7], [11]. Those works assume that all the pins in the array must be escaped on a single layer and the routing is symmetrical. This symmetry assumption is the foundation of their algorithms. However, due to the high pin density in modern packages and PCB designs, not all pins in the array can be escaped on one layer and the escape routing is very likely to be asymmetrical. Therefore, their algorithms are not applicable to the more general escape routing problem we discuss here

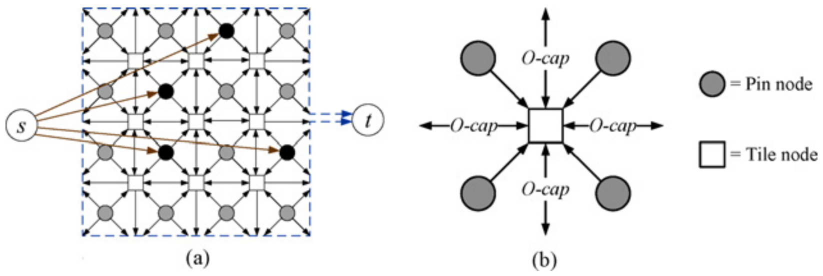


Fig. 2. Traditional network flow model used in previous works. Black pins denote the to-be-escaped pins and there are unit capacity edges from the super source s to them. Edges extending outside the boundary of the pin array are all incident to the super sink t. (a) Flow model for a pin grid. (b) Inside a tile.

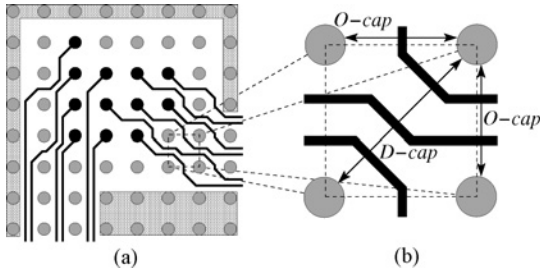


Fig. 1. (a) Example of the escape routing problem and (b) enlarged view of a tile. Shaded areas denote blockages. Specified pins (black) are escaped to the boundary of the pin grid. In this example, *O-cap* = 2 and *D-cap* = 3.

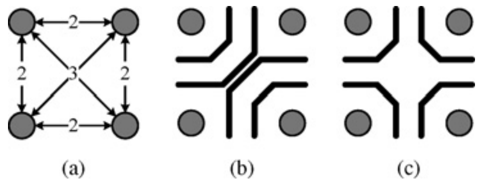


Fig. 3. Previous network flow models cannot handle *D-cap* correctly. For (a) case *O-cap* = 2, *D-cap* = 3, models in [8] and [10] may produce (b) illegal routing and models in [2] and [3] cannot capture (c) valid routing.

(we do not require all the pins to be escaped on one layer or the routing to be symmetrical).

In this paper, we propose a network flow model that correctly models the diagonal capacity. Our model guarantees to give a legal routing if one exists. We then build an algorithm based on this model. As far as we know, this is the first algorithm that guarantees optimality. We also extend our model to handle missing pins, which are unused pins removed to increase the routing resource. Experimental results show that our algorithm has very short runtime.

The rest of this paper is organized as follows. Section II introduces necessary background information. Section III presents our network flow model and escape routing algorithm. Section IV extends our model to deal with missing pins. Finally, experimental results are given in Section V and concluding remarks are given in Section VI.

II. Background

In this section, we will first formulate the escape routing problem and then introduce the traditional network flow model used by some previous works.

A. Problem Formulation

The input to the escape routing problem is an $m \times n$ pin array with p pins specified as to-be-escaped pins. Certain areas of the pin array are marked as blockages, which do not allow any routing inside. Such blockages are used to model prerouted nets or to guide the routing to escape through certain preferred boundaries. *O-cap* and *D-cap* are also given to specify the orthogonal capacity and diagonal capacity in the tile. We can safely assume that $O\text{-}cap \leq D\text{-}cap \leq 2 \cdot O\text{-}cap$ for all our inputs due to the following two facts.

1) The diagonal of a square tile is longer than the side of the square. Therefore, $D\text{-}cap \geq O\text{-}cap$.
2) The *O-cap* constraint already implies that at most $2 \cdot O\text{-}cap$ wires can pass the diagonal. So setting *D-cap* to be larger than $2 \cdot O\text{-}cap$ is meaningless.

The expected output of the problem is an octilinear planar routing from the to-be-escaped pins to the boundary of the pin array satisfying the capacity constraints and avoiding the blockages. We would also like the total length of the routing to be minimized.

B. Previous Works

In the traditional network flow model used by [2], [3], [8], and [10],[1] each pin is represented by a pin node and each tile is represented by a tile node. Edges are added between horizontally and vertically adjacent tile nodes and from pin nodes to their adjacent tile nodes (see Fig. 2). Edges extending out of the pin grid boundary are connected to a super sink. There are also edges from a super source to the pin nodes that are expected to be escaped.

This model works fine if we consider only the orthogonal capacity *O-cap* because we can add capacity constraints on the orthogonal edges in the network to realize *O-cap*. However, diagonal capacity *D-cap* is not captured by this model. Consider the case where *O-cap* = 2 and *D-cap* = 3, which is very common for PCB routing. Since the model has no control over the number of wires passing through the diagonals of the tile, it may produce routing like Fig. 3(b) which violates *D-cap*. In [2] and [3], the number of wires inside a tile is also limited by adding node capacities to tile nodes. However, such node capacity does not correctly reflect the diagonal capacity. No matter how we set the tile node capacity, there are always counterexamples.

[1]Some of these works assume monotonic and/or symmetric routing. Therefore, some orthogonal edges are unidirectional in their models. However, the network structure is the same as what we show here.

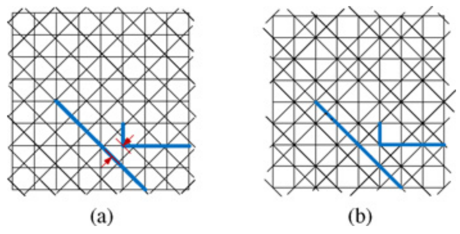


Fig. 4. Flow network in [12]. (a) Original network obtained by overlaying a rotated grid on an orthogonal grid. (b) Network after line shift. The thick flow shown in the network violates the spacing rule.

1) Tile node capacity ≥ 4. This is the same as having no tile capacity because *O-cap* = 2 already implies at most four wires in a cell. The network may produce the routing in Fig. 3(b), which violates *D-cap*.
2) Tile node capacity ≤ 3. The network cannot model the routing in Fig. 3(c) because there are four wires inside the tile. However, the routing itself is legal because only two wires pass each diagonal. As a result, the network model may fail to capture a legal routing solution even when one exists.

The network in [12] is different from the traditional network model in Fig. 2. It is essentially a rotated rectangular routing grid overlaid on top of another rectangular routing grid (see Fig. 4). The rotated grid lines provide the wiring tracks for 45° routing. However, this introduces many unwanted intersections between the 45° routing tracks and the orthogonal routing tracks. Therefore, a "line shift" technique is proposed in [12] to shift the 45° tracks to the closest orthogonal grid nodes so that the excessive intersections are removed. Such "line shift" is only topological. The actual 45° routing tracks remain at the original locations.

In [12], it is claimed that all min-cost flow solutions of the network will always satisfy the spacing rule. Unfortunately, this is not true. Take a look at the thick flow in Fig. 4. We can see that the flow solution on the right already has minimal cost (the cost is defined as the path length in [12]). However, the corresponding routing paths on the left have distance less than the spacing between adjacent wiring tracks, meaning that the spacing rule is violated.

From the discussion above, we can see that none of the previous network models can model the diagonal capacity correctly. In the next section, we will present a network model that captures the diagonal capacity correctly.

III. Our Network Flow Model

In our proposed network flow model, each tile contains five nodes, namely N-node on the north, E-node on the east, S-node on the south, W-node on the west, and C-node in the center (see Fig. 5). The first four nodes are called peripheral nodes and the last node is called the center node.

We give the center node a capacity of $D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor$. Node capacity can be realized by splitting the node into two and adding an edge with corresponding capacity between them (see Fig. 6).

We create bidirectional edges (each bidirectional edge is realized by two directed edges: a forward edge and a

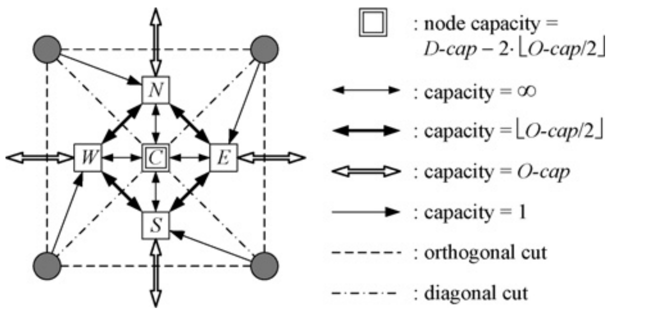

N
W
C
E
S
: node capacity = $D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor$
: capacity = ∞
: capacity = $\lfloor O\text{-}cap/2 \rfloor$
: capacity = $O\text{-}cap$
: capacity = 1
: orthogonal cut
: diagonal cut

Fig. 5. Our network flow model inside a tile.

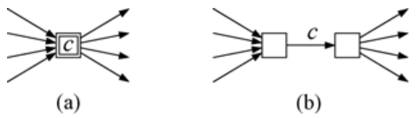


Fig. 6. Turning a (a) node with capacity c into an (b) edge with the same capacity.

backward edge) between every peripheral node and the center node and give these edges infinite capacity. We also introduce bidirectional edges between peripheral node pairs (N, E), (E, S), (S, W), and (W, N). We call such edges peripheral edges and give each of them capacity $\lfloor O\text{-}cap/2 \rfloor$. The five nodes and the edges between them compose an intratile network. Connections between tiles are also necessary. We use bidirectional edges to connect the N-node of a tile to the S-node of the tile above it as well as the E-node of a tile to the W-node of the tile to the right of it. Such edges are called intertile edges and have capacity *O-cap*. In order to escape the pins, we also create a pin node for each pin and create directed edges from the four pins at the corners of a tile to the four peripheral nodes in the tile. The edges are from the pin at the NW corner to N-node, from the pin at the NE corner to E-node, from the pin at the SE corner to S-node, and from the pin at the SW corner to W-node. We call these edges pin edges. All pin edges have capacity 1. Of course, if any nodes or edges lie in the blockage, we will not create them.

Similar to the traditional model, we also introduce a super source s and a super sink t. All edges from the boundary tiles to the outside of the pin array are connected to t. Finally, we add edges with capacity 1 from s to the pin nodes of all the to-be-escaped pins.

The intuition behind this network model is that we need the peripheral edges to capture diagonal routes of different directions. The center node itself cannot distinguish the 45° route and 135° route, which is the reason why the traditional model cannot distinguish the case between Fig. 3(b) and (c). With the peripheral edges which have capacity $\lfloor O\text{-}cap/2 \rfloor$, we are able to distinguish the case. Then, in order to ensure the diagonal capacity constraint, we give the center node a capacity of $D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor$ so that the diagonal cuts have capacity of exactly *D-cap*.

A. Correctness of the Model

We call a flow of the network *legal* if the flow through every edge is an integer that does not exceed the edge capacity. The

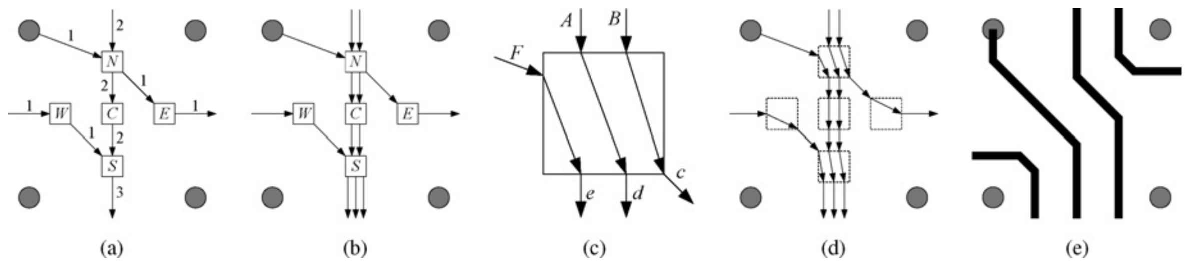


Fig. 7. Convert a flow solution into escape routing. (a) Flow inside a tile. (b) Flow splitting. (c) In-node connection. (d) Topological routing. (e) Detailed routing.

TABLE I
CONNECT FLOW IN FIG. 7(C)

Iter.	Seq.	Action
1	$\{A, B, c, d, e, F\}$	Connect B to c, remove them from seq.
2	$\{A, d, e, F\}$	Connect A to d, remove them from seq.
3	$\{e, F\}$	Connect F to e, remove them from seq.
4	$\emptyset$	N/A

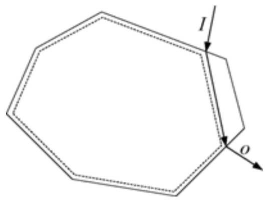


Fig. 8. Induction proof of our procedure.

total flow from source s to sink t is called the *value* of the flow. We call a routing *legal* if it satisfies *O-cap* and *D-cap* constraints in all tiles.

Any legal flow of the network can be decomposed into a collection of unit flow which corresponds to routing paths. Therefore, there is a correspondence between a flow solution and a routing solution.

Lemma 1: Any legal flow of value k corresponds to a legal escape routing of k pins.

Proof: We can transform the flow into escape routing by three steps (see Fig. 7).

1) *Split the Flow:* For any edge e, if $flow(e) > 1$, we split the flow into $flow(e)$ copies of unit flow [Fig. 7(b)]. Notice that a legal flow is an integer flow. It is always possible to split any integer flow into unit flow.

2) *Connect Flow Inside Each Tile Node:* Let us regard a tile node as a square. We need to connect the incoming flow to the outgoing flow without introducing any intersections. This can be done by the following procedure. First, we obtain a sequence of incoming and outgoing flow by going around the four sides of the square clockwise starting at the northwest corner. For example, the sequence for Fig. 7(c) is $\{A, B, c, d, e, F\}$ (we use uppercase letter to denote incoming flow and lowercase letter to denote outgoing flow). Then, we repeatedly find an incoming flow and an outgoing flow that are adjacent in the sequence, connect the incoming flow to the outgoing flow, and remove both flow from the sequence. We keep doing this until the sequence is empty (which means all incoming flow is connected to the outgoing flow). The corresponding procedure for Fig. 7(c) is illustrated in Table I.

According to the flow conservation law, we have the same number of incoming flow and outgoing flow. Therefore, the above procedure can always connect all the flow inside the node square. We still need to show that such connection will not result in any intersections. This can be proved by induction. We show that for any convex shape (square is a convex shape), if we have the same number of incoming flow and outgoing flow along its boundary, the above procedure would produce a crossing-free connection. The base case is easy to show, any straight line segment with two ends on the boundary of a convex shape stays inside the convex shape. Now suppose our procedure works for n incoming flow and n outgoing flow. For a shape with $n+1$ incoming flow and $n+1$ outgoing flow, connecting adjacent incoming flow I to outgoing flow o will cut the convex shape into two convex shapes (see Fig. 8). One shape (the dashed one in the figure) contains n incoming flow and n outgoing flow while the other is empty because I and o are adjacent in the sequence. By induction, we know that the n pairs of flow can be connected without intersection inside the dashed convex shape. Furthermore, straight line segments with two ends inside a convex shape stay inside the convex shape, meaning that none of the n connections can intersect with connection between I and o. Therefore, our procedure produce intersection-free result.

3) *Transform Topological Routing into Detailed Routing:* The connected flow obtained in the previous step gives a planar topology of the routing [Fig. 7(d)]. We call such a flow a topological routing. It can be seen in Fig. 5 that any orthogonal cut[2] cuts only one intertile edge with capacity exactly *O-cap*. Any diagonal cut[3] cuts two peripheral edges plus the center node. The total capacity of this cut is $2 \cdot \lfloor O\text{-}cap/2 \rfloor + D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor = D\text{-}cap$. Therefore, if the flow is legal, then the topological routing also satisfies all the *O-cap* and *D-cap* constraints. We can then use algorithms in [22] and [23] to transform the topological routing into octilinear detailed routing. ■

Notice that the same steps in the above proof can be used to transform flow into detailed escape routing for traditional model too. However, the resultant escape routing may not

[2]Orthogonal cut is the cut between orthogonally adjacent pins, see the dashed line segments in Fig. 5.

[3]Diagonal cut is the cut between diagonally adjacent pins, see the dash-dot line segments in Fig. 5.

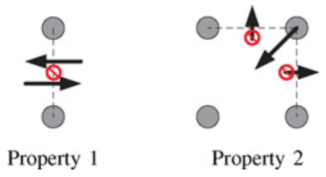


Fig. 9. Two properties of a directed routing with minimum number of crossings with all orthogonal cuts.

satisfy the diagonal capacity because traditional model cannot capture it.

On the other hand, there exists a legal flow for any legal escape routing.

Lemma 2: If there exists a legal escape routing of k pins, then our model has a legal flow of value k.

Proof: The proof is by construction. We show that at least one legal routing of k pins can be converted into a legal flow of value k.

We call a legal routing minimal if all its wires have the minimum number of crossings with all the orthogonal cuts. We assign a direction to each wire, which is from the pin to the array boundary. Such a directed minimal routing has the following properties (see also Fig. 9).

1) No two wires of opposite directions can pass the same orthogonal cut.
2) If a wire is routed from a pin into one tile, then no wire can exit the tile from the two orthogonal cuts incident to that pin.

Let us show why these two properties hold. If we have two wires of opposite directions passing one orthogonal cut, we can reconnect the two wires as shown in Fig. 10 and reduce the number of crossings with the orthogonal cut. Notice that the other cuts are not affected by this change. Therefore, the routing after the reconnection is still legal. This contradicts our assumption that the minimal routing already has minimum number of crossings with all the orthogonal cuts. Therefore, Property 1 is true.

For Property 2, let us assume that there is one wire exiting the tile from an orthogonal cut incident to the pin. There are two situations: 1) the wire is from that pin, and 2) the wire is from some other pin. For the first situation, we can simply shift the wire to the neighboring tile to reduce the number of crossings between the wires and the orthogonal cuts (see Fig. 11). Note that this change has no impact on other cuts and thus the resultant routing is still legal. This contradicts our assumption that minimal routing already has the minimum number of crossings with all the orthogonal cuts. For the second situation, we reconnect the wires as shown in Fig. 12 and reduce it to the first situation. Therefore, Property 2 is true.

Now we construct a legal flow solution from a minimal routing in two steps.

1) We construct the flow on intertile edges and pin edges.
2) We construct the flow in the intratile network.

If we view each directed wire in the routing as a unit flow from a pin to the array boundary, we can obtain the flow crossing each orthogonal cut. We then assign the same flow

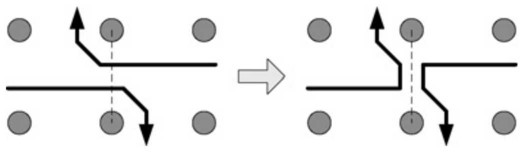


Fig. 10. By reconnecting the wires, we can reduce the number of crossings between the wires and the orthogonal cuts without affecting the legality of the routing.

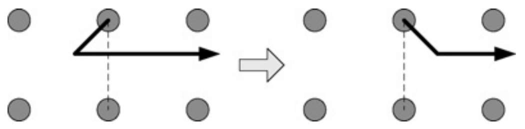


Fig. 11. By shifting the wire from the pin to the neighboring tile, we can reduce the number of crossings between the wires and the orthogonal cuts without affecting the legality of the routing.

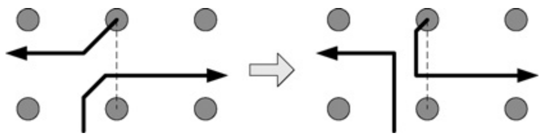


Fig. 12. By reconnecting the wires, we can reduce the second situation to the first situation.

to the intertile edges corresponding to the orthogonal cuts. If there is a wire from a pin to a tile, we assign a unit flow to the corresponding pin edge. Notice that such a flow assignment would not violate the capacity constraint because at most *O-cap* wires can cross an orthogonal cut and at most one wire can be routed from a pin in a legal routing.

Now with all the flow on intertile edges and pin edges determined, we know the flow going in and coming out of any intratile network. We also know that total incoming flow = total outgoing flow for an intratile network because of the continuity of the routing. Therefore, we can apply the flow algorithm to obtain a flow solution for the intratile network. Here, we show that for any incoming flow and outgoing flow configurations obtained from Step 1, we can always obtain a legal flow solution for the intratile network.

We can classify the possible configurations of the incoming/outgoing flow of the intratile network into three categories.

1) The flow comes into the intratile network at only one peripheral node and out at one or more peripheral nodes. Without loss of generality, we assume that the flow comes in at N-node. There are two cases: either the incoming flow contains only flow from the intertile edge [Fig. 13(b)] or the incoming flow also includes a flow from the pin [Fig. 13(c)]. In the first case, the total incoming flow is bounded by *O-cap* because at most *O-cap* wires can pass an orthogonal cut in a legal routing. In the second case, we know that no flow can come out of the tile from W-node due to Property 2. Correspondingly, in the routing, the wires come into the tile from the top boundary of the tile and from the pin. These wires must exit the tile at the bottom and/or right boundary. As a result, all wires must cross the diagonal cut shown in

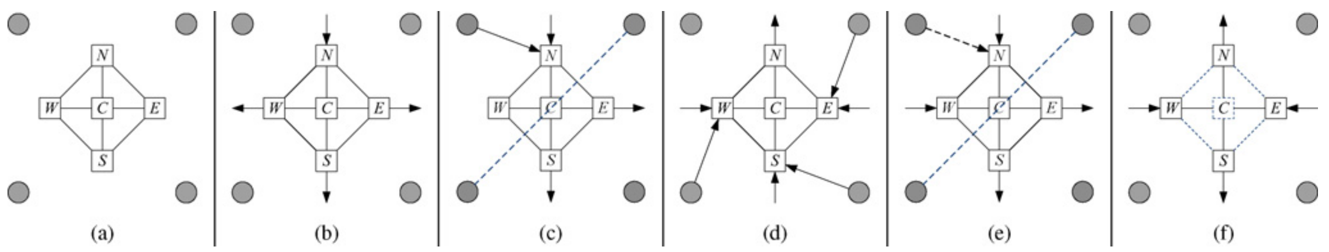


Fig. 13. Constructing a flow solution of the intratile network. (a) Intratile network. (b)–(f) Analysis of the possible flow configurations of the intratile network.

Fig. 13(c). Therefore, the total number of wires in the tile cannot exceed *D-cap*, meaning that the total incoming flow cannot exceed *D-cap*.

2) The flow comes into the intratile network at one or more peripheral nodes but exits the network at only one peripheral node [Fig. 13(d)]. In this case, the total outgoing flow is bounded by *O-cap* because at most *O-cap* wires can pass an orthogonal cut in a legal routing.
3) The flow comes into the intratile network at exactly two peripheral nodes and exits the network at exactly two peripheral nodes. There are two cases: either the two incoming nodes are adjacent to each other [Fig. 13(e)] or the two incoming nodes are not adjacent [Fig. 13(f)]. For the first case [Fig. 13(e)], we assume that the flow comes in at N, W nodes without loss of generality. According to Property 2, there should be no flow from any pin to the W, S, and E nodes. (However, there could be flow from the pin to the N node as shown by the dashed arrow.) All the wires in the tile must cross the diagonal cut (dashed cut in the figure). Therefore, the total number of wires in the tile cannot exceed *D-cap*, meaning that the incoming flow is bounded by *D-cap*. We will discuss the second case [Fig. 13(f)] later.

From the above discussion, we can see that total incoming flow $\leq$ *D-cap* for all the cases except the one in Fig. 13(f). On the other hand, we can verify by enumeration that any cut of the intratile network (a cut of a network is a set of edges that separate the network into two disconnected components) has a capacity of at least *D-cap*. According to the max-flow min-cut theorem [24], we know that there must exist a legal flow solution of the intratile network.

Now let us discuss the situation in Fig. 13(f), in which the flow comes in at two nonadjacent peripheral nodes and exits at the two other peripheral nodes. Without loss of generality, we assume the flow comes in at W and E. By Property 2, we know that there exists no flow from any pin in this tile. The total incoming flow is bounded by $2 \cdot$ *O-cap* because each orthogonal cut allows at most *O-cap* wires. The min-cut between the incoming nodes W, E and the outgoing nodes N, S includes the four peripheral edges and the center node C. [The dashed edges and node in Fig. 13(f) illustrate the cut.] Therefore, the capacity of the min-cut, which we denote as MC, is

$$4 \cdot \lfloor O\text{-}cap/2 \rfloor + D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor = 2 \cdot \lfloor O\text{-}cap/2 \rfloor + D\text{-}cap.$$

If *O-cap* is even, then $2 \cdot \lfloor O\text{-}cap/2 \rfloor = O\text{-}cap$.

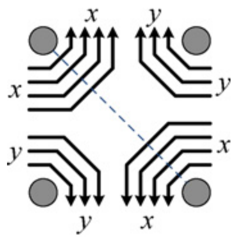


Fig. 14. If *O-cap* is odd and there are $2 \cdot O\text{-}cap$ wires passing the tile, then $D\text{-}cap \geq 2x \geq O\text{-}cap + 1$.

Therefore

$$MC = O\text{-}cap + D\text{-}cap \geq 2 \cdot O\text{-}cap.$$

Since the total incoming flow is bounded by $2 \cdot$ *O-cap*, we know that there must exist a legal flow through the max-flow min-cut theorem.

If *O-cap* is odd, then $2 \cdot \lfloor O\text{-}cap/2 \rfloor = O\text{-}cap - 1$. Therefore

$$MC = O\text{-}cap + D\text{-}cap - 1 \geq 2 \cdot O\text{-}cap - 1.$$

So if the total incoming flow does not exceed $2 \cdot O\text{-}cap - 1$, there must exist a legal flow solution of the intratile network. If the total incoming flow is exactly $2 \cdot$ *O-cap*, then in the routing, there are exactly *O-cap* wires coming in from the left side of the tile and exactly *O-cap* wires coming in from the right side of the tile (see Fig. 14).

Suppose x wires go from left to top and y wires go from left to bottom ($x + y = O\text{-}cap$). In order to satisfy the orthogonal capacity, y wires from the right must go to the top side and x wires from the right must go to the bottom. As a result, we have $2x$ wires crossing one diagonal cut and $2y$ wires crossing another diagonal cut. Notice that *O-cap* is odd, $x \neq y$. Without loss of generality, let us assume $x > y$. Then $2x > x + y = O\text{-}cap$. Since x is an integer, we know $2x \geq O\text{-}cap + 1$. Because the routing is legal, we have $D\text{-}cap \geq 2x \geq O\text{-}cap + 1$. Therefore

$$MC = O\text{-}cap + D\text{-}cap - 1 \geq 2 \cdot O\text{-}cap.$$

So $MC \geq 2 \cdot O\text{-}cap$ but total incoming flow is bounded by $2 \cdot O\text{-}cap$. By the max-flow min-cut theorem, we know that there must exist a legal flow solution of the intratile network. ■

Lemma 1 shows that the flow solution of our model can always be turned into a legal detailed routing solution while Lemma 2 shows that if there exists a legal detailed routing

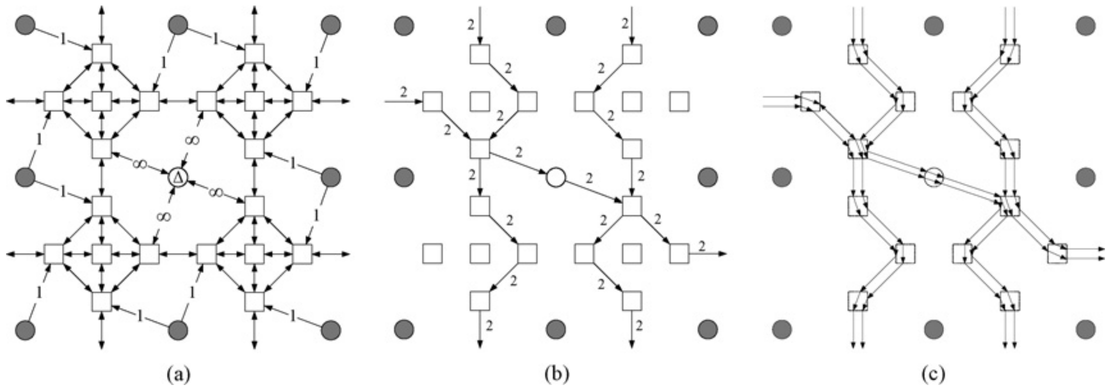


Fig. 16. Modeling missing pins in our network. (a) Extended model to consider missing pin. (b) Flow solution of the model corresponding to Fig. 15. (c) Topological routing derived from the flow solution.

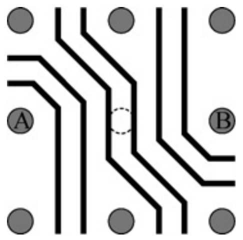


Fig. 15. Missing pins increase the routing resource.

solution, then our model will always capture one. As a result, we have the following theorem.

Theorem 1: A given escape routing problem with k to-be-escaped pins has a legal routing solution *iff* our network model has a legal flow of value k. Furthermore, the legal flow of value k can be transformed into a legal routing solution.

In order to minimize the wire length, we can assign cost 1 to the intertile edges (the hollow edges in Fig. 5) and zero cost to all other edges. If we compute the min-cost max-flow of the network, we can minimize the number of tiles each wire traverses and thus the total wire length can be minimized. The min-cost max-flow solution can then be converted into detailed routing through the transformation in the proof of Lemma 1.

IV. Modeling the Missing Pins

In practical PCB designs, some unused pins might be removed to increase the routing resource. Fig. 15 shows such an example in which *O-cap* = 2 and *D-cap* = 4. It can be seen that by removing the pin at the center, the maximum number of wires allowed between A and B increases from 4 to 6. We call the difference, 2, the extra horizontal capacity of the missing pin. Similarly, we can define extra vertical capacity and extra diagonal capacity. We define the extra capacity to be the minimum of the extra horizontal capacity, extra vertical capacity and extra diagonal capacity and denote it by Δ. This extra capacity is a conservative estimation of how much extra routing resource we gain from removing the pin and can be derived from the design rules. It is given by the user.

We extend our model to consider the missing pins as follows. We use a resource node to replace the pin node [see Fig. 16(a)]. We set the capacity of the resource node to Δ to capture the extra capacity created by the missing pin. The cost of the resource node needs to correspond to the length of the wire passing the missing pin. The length of the wire is bounded between 1 and 2 (remember that we use the number of tiles passed by the wire as unit of length). But the exact length depends on how the wire passes the missing pin (horizontally, vertically, or diagonally), so the cost of the resource node can only be an estimation of the actual length. In our model, we set the cost of the resource node to be 1.4. Note that the node capacity and cost will eventually be translated into edge capacity and cost when we perform node splitting as shown in Fig. 6. The directed edges from the pin node to the peripheral nodes are replaced with bidirectional edges between the resource node and the peripheral nodes. We give such edges infinite capacity and zero cost.

Theorem 1 is no longer guaranteed when missing pins are taken into consideration. In other words, our model is no longer optimal. There are several reasons to it.

1) The extra capacity Δ is a conservative estimation of the actual resource created by the missing pin because we take the minimum of extra horizontal capacity, extra vertical capacity, and extra diagonal capacity.
2) The concept of "tile" becomes unclear when a continuous block of pins are missing. For the extreme case, imagine that all the pins in the array are missing. It is hard to define a tile in this case. As a result, it is impossible for our tile-based flow model to produce optimal solution.

Fortunately, the above drawbacks of our model do not pose serious issues in practice.

1) A typical pin is usually either circular or octagonal in practical PCB or package design. Therefore, it is likely that the horizontal extra capacity, vertical extra capacity, and diagonal extra capacity are the same. Taking the minimum of them may not underestimate the actual extra routing resource.
2) Missing pins are usually scarce and sparse in practical designs. It is very rare to see a large block of missing pins. One exception is that we might see blank space in the center of certain types of pin arrays as illustrated in Fig. 17. The space is usually reserved for the silicon chip and it bonding wires. In this case, we would not have

TABLE II
EXPERIMENTAL RESULTS OF OUR PROPOSED NETWORK FLOW MODEL

	Array	Escape	Missing	Capacities			Our Model		Traditional Model	
	$w \times h$	Pin No.	Pin No.	O	D	Δ	*D-cap* vio.	Runtime	*D-cap* vio.	Runtime
*industrial*_1	14 × 16	78	13	2	3	3	0	0.16 s	0	0.14 s
*industrial*_2	29 × 11	66	20	2	3	3	0	0.22 s	10	0.17 s
*industrial*_3	33 × 14	120	46	2	3	3	0	0.33 s	6	0.28 s
*industrial*_4	35 × 17	160	30	2	3	3	0	0.61 s	9	0.28 s
*industrial*_5	35 × 35	108	105	2	3	3	0	0.86 s	1	0.68 s
*industrial*_6	35 × 17	143	38	2	3	3	0	0.39 s	0	0.30 s
*industrial*_7	35 × 35	220	106	2	3	3	0	1.01 s	53	0.79 s
*modified*_8	35 × 35	225	33	2	3	3	0	0.99 s	53	0.79 s

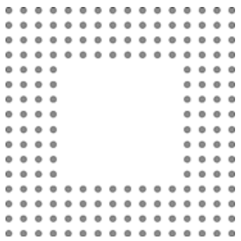


Fig. 17. Pin array with blank space in the center.

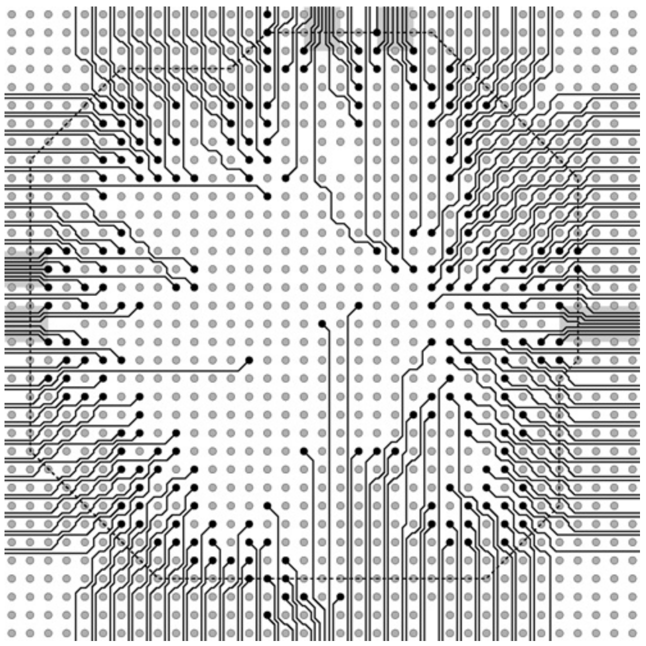


Fig. 18. Routing solution of *modified*_8. The shaded zones highlight the spaces of the missing pins that are fully utilized by our router. The dashed polygon is drawn on top of the result to show that the routing uses up almost all routing resources.

very dense routing in the center because wires are usually escaping toward the outer boundary of the array on a PCB. Not being able to capture the optimal solution might not affect the effectiveness of our model in this situation.

In fact, our model performs very well on practical designs. As we will see in the experiments, our model can efficiently utilize the empty space created by missing pins in a congested design.

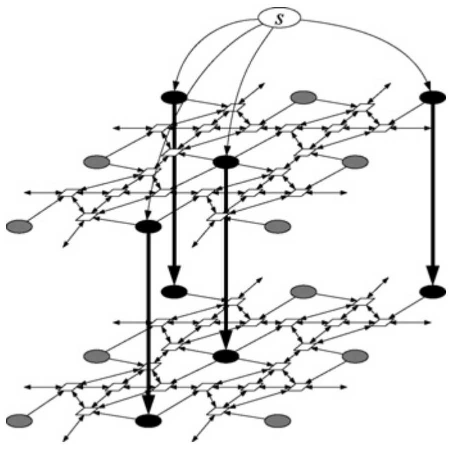


Fig. 19. Multilayer network. Black pins are to be escaped and thick edges are interlayer edges.

V. EXPERIMENTAL RESULTS

We implement our network flow-based escape routing algorithm in C++ and test it on several industrial data sets. We use the min-cost flow solver CS2 [25] to obtain the min-cost flow solution of our network model. All experiments are performed on a workstation with two 3.0 GHz Intel Xeon processors and 4 GB memory. The operating system is RedHat Linux 2.6.9.

We test our router on eight data sets and the result is reported in Table II.

Among the eight data sets, *industrial*_1 to *industrial*_7 are actual industrial data and *modified*_8 is derived from industrial data with some modification. The left five columns of the table give the information on the data including the name, the pin array size, the number of to-be-escaped pins, the number of missing pins, and the capacity rules (*O-cap*, *D-cap*, and extra capacity Δ). The next two columns show the number of *D-cap* constraint violations in our result as well as the runtime of our router. The last two columns show the number of *D-cap* violations and the runtime of the traditional model in [8] and [10].

It can be seen that our model gives zero *D-cap* violations while the traditional model leads to as many as 53 *D-cap* violations because it ignores the diagonal capacity. Since the total runtime is only 1 s or less, the runtime difference of the two methods is insignificant.

Fig. 18 shows our routing solution of *modified*_8. We can see that there are several missing pins on the north, west, and east side of the array (highlighted by the shaded zones), and their spaces are fully utilized in our result. To show that our router can handle dense designs, we draw a dashed polygon on the routing result. It can be seen that almost all the *O-cap* and *D-cap* along the polygon are used up by our routing, which indicates that the routing is very dense.

VI. Conclusion

In this paper, we presented a novel network flow model that correctly models the diagonal capacity which is essential for 45° routing in packages and PCBs. We proved the correctness of our model. We also showed how to extend our model to handle missing pins, which appear frequently in practical designs.

Notice that if the max-flow solution of our network has a value less than the number of to-be-escaped pins, then not all these pins can be escaped on a single layer. We have to use multiple layers to escape all of them. In this case, we can extend our model to find out the minimum number of layers to escape all the pins. Supposing we are given k layers, we can build k copies of our network. Then interlayer edges are added between adjacent layers to model the vias. Each interlayer edge is from a to-be-escaped pin to the corresponding pin on the layer below (see Fig. 19). The capacity of the edge is 1 and the cost is the cost of a via. The edges from the super source s connect only to the to-be-escaped pin on the top layer.

If the max-flow of this network and the number of to-be-escaped pins are the same, then k layers are enough to escape all the pins. Otherwise, we need more layers. We can use binary search on k to find out the minimum layer number such that all the pins can be escaped.

References

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

[2] 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.

[3] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design," in *Proc. Int. Conf. Comput.-Aided Des.*, 2008, pp. 518–522.

[4] W.-T. Chan and F. Y. L. Chin, "Efficient algorithms for finding the maximum number of disjoint paths in grids," *J. Algorithms*, vol. 34, no. 2, pp. 337–369, 2000.

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

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

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

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

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

[10] W.-T. Chan, F. Y. L. Chin, and H.-F. Ting, "A faster algorithm for finding disjoint paths in grids," in *Proc. Int. Symp. Algorithms Comput.*, 1999, pp. 393–402.

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

[12] R. Wang and C.-K. Cheng, "Octilinear redistributive routing in bump arrays," in *Proc. Great Lakes Symp. Very Large Scale Integr.*, 2009, pp. 191–196.

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

[14] Y. Kubo and A. Takahashi, "Global routing by iterative improvements for two-layer ball grid array packages," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 725–733, Apr. 2006.

[15] Y. Kubo and A. Takahashi, "A global routing method for 2-layer ball grid array packages," in *Proc. Int. Symp. Phys. Des.*, 2005, pp. 36–43.

[16] Y. Tomioka and A. Takahashi, "Monotonic parallel and orthogonal routing for single-layer ball grid array packages," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2006, pp. 642–647.

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

[18] M. M. Ozdal and M. D. F. Wong, "Simultaneous escape routing and layer assignment for dense PCBs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 822–829.

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

[20] Q. Ma, T. Yan, and M. D. F. Wong, "A negotiated congestion based router for simultaneous escape routing," in *Proc. Int. Symp. Qual. Electron. Des.*, 2010, pp. 606–610.

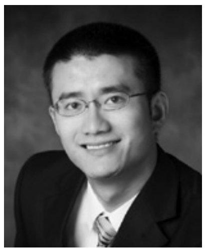
[21] L. Luo, T. Yan, Q. Ma, M. D. F. Wong, and T. Shibuya, "B-escape: A simultaneous escape routing algorithm based on boundary routing," in *Proc. Int. Symp. Phys. Des.*, 2010, pp. 19–25.

[22] D. J. Staepelaere, "Geometric transformations for a rubber-band sketch," M.S. thesis, Dept. Comput. Eng., Univ. California, Santa Cruz, CA, Sep. 1992.

[23] D. Staepelaere, J. Jue, T. Dayan, and W. W.-M. Dai, "SURF: Rubber-band routing system for multichip modules," *IEEE Des. Test. Comput.*, vol. 10, no. 4, pp. 18–26, Dec. 1993.

[24] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[25] B. Cherkassky and A. Goldberg. (1997). *CS2: Min-Cost Flow Solver* [Online]. Available: http://www.igsystems.com/cs2/index.html



Tan Yan received the B.S. degree in computer science and technology from Fudan University, Shanghai, China, the M.Eng. degree in information engineering from the University of Kitakyushu, Kitakyushu, Japan, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, in 2010.

He is currently a Senior Research and Development Engineer with Synopsys, Inc., Mountain View, CA. His current research interests include integrated circuit routing. However, he has a broad interest in various topics in electronic design automation including physical design, parallel computer-aided design (CAD), and CAD for emerging nanotechnologies.

Dr. Yan was the recipient of the Best Paper Award at ISPD 2010 and the the Gold Medalist of the ACM Student Research Competition at DAC 2010.

Martin D. F. Wong (F'06) received the B.Sc. degree in mathematics from the University of Toronto, Toronto, ON, Canada, the M.S. degree in mathematics from the University of Illinois at Urbana-Champaign (UIUC), Urbana, and the Ph.D. degree in computer science from UIUC in January 1987.

From 1987 to 2002, he was a Faculty Member in computer science with the University of Texas, Austin. He returned to UIUC in 2002 where he is currently a Professor of electrical and computer engineering with the Department of Electrical and Computer Engineering. He has published approximately 400 technical papers and graduated 41 Ph.D. students in the area of electronic design automation (EDA).

Dr. Wong has won a few best paper awards for his works in EDA. He has served on many technical program committees of leading EDA conferences. He has also served on the editorial boards of the IEEE Transactions on Computers, the IEEE Transactions on Computer-Aided Design, and the *ACM Transactions on Design Automation of Electronic Systems*. He was an IEEE Distinguished Lecturer from 2005 to 2006.