Mixed-Crossing-Avoided Escape Routing of Mixed-Pattern Signals on Staggered-Pin-Array PCBs

Kan Wang, *Student Member, IEEE,* Sheqin Dong, *Member, IEEE,* Huaxi Wang, Qian Chen, and Tao Lin

***Abstract*—Escape routing has become a critical issue in high-speed PCB routing. Most of the previous work paid attention to either differential-pair escape routing or single-signal escape routing, but few considered them together. In this paper, a significant three-stage algorithm is proposed to solve the problem of escape routing of both differential pairs and single signals (mixed-pattern signals). First, differential pairs are preconditioned to reduce the complication of the problem. Then, a unified ILP model is used to formulate the problem and a novel Boolean coding-driven algorithm is proposed to avoid mixed crossings. Finally, a slice-based method is presented to prune the variables and speed up the algorithm. Experimental results show that the proposed method is very effective. For single-pattern escape routing, it can solve all the test cases in short time and reduce wire length and chip area by 16.1% and 15.5%, respectively. For mixed-pattern escape routing, it can increase the routability by 17.5% and reduce the wire length by 14.1% compared to a two-stage method. At the same time, the proposed method can effectively avoid mixed crossings with only a little increase on wire length. Furthermore, with slice-based speedup strategy, the method can reduce the solving time by 76.7%.**

***Index Terms*—Differential pair, escape routing, mixed crossing, mixed-pattern signals, single signal, staggered pin array.**

I. Introduction

HIGH speed printed circuit board (PCB) routing has become more and more difficult for manual design due to the increased pin count and dwindling routing resource. As one of the most important issues, escape routing [1] can greatly influence the whole quantity of PCB routing. To address the escape routing problem, many methods were proposed, including pin array structure [2], [3] and escape routing algorithms [4]–[12].

For pin array structure, although grid pin array (GPA) [4] has been widely used, it cannot satisfy the demands of the ever-increasing pin number on PCB. Then another pin array structure, staggered pin array (SPA) was formed [2]. Compared to GPA, the SPA can increase pin density greatly under the similar number of pins and the same area constraints [3]. Due to the different routing resources and constraints, the routing on SPA will be much different.

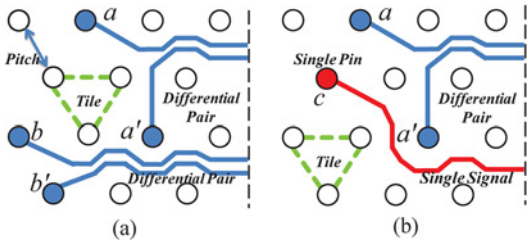


Fig. 1. Escape routing pattern. (a) Single-pattern escape routing on staggered pin array. (b) Mixed-pattern escape routing on staggered pin array.

For escape routing, differential-pair escape routing (two nets for each signal) [5]–[7] and single-signal escape routing (one net for each signal) [8]–[12] have been developed on GPA or SPA in recent years. However, all previous researches focused on only differential-pair escape routing or only single-signal escape routing and did not consider them together.

Due to high noise immunity and low electromagnetic interference, differential pairs are always used for the high-speed signal transmission on PCB [5]. However, due to the increased pin count and limitation of resources, not all signals can be transmitted by differential pairs. As a result, the signals of differential pairs and single signals will coexist on board and hence the research on escape routing of both of them is quite valuable. For convenience, the escape routing only for differential pairs or only for single signals is referred as single-pattern escape routing and the escape routing of both differential pairs and single signals simultaneously is referred as mixed-pattern escape routing, as shown in Fig. 1. Note that single-pattern escape routing is just a special case of mixed-pattern escape routing.

Unlike traditional single-pattern escape routing, the mixed-pattern escape routing is much more difficult. First, for single-pattern escape routing, the problem can be transferred into network flow problem as there is only one signal source, while for mixed-pattern escape routing, there will be two independent signal sources. Second, the two kinds of signals take different routing resources: each differential pair takes two

Manuscript received June 24, 2013; revised September 30, 2013 and November 11, 2013; accepted December 27, 2013. Date of current version March 17, 2014. This work was supported in part by the MOST of China project under Grant 2011DFA60290 and in part by the NSFC under Grant 61176022. This paper was recommended by Associate Editor C. Sze.

K. Wang, S. Dong, H. Wang, and Q. Chen are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: wangkan09@mails.tsinghua.edu.cn; dongsq@mail.tsinghua.edu.cn).

T. Lin is now with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: lintao87@gmail.com).

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.2014.2301676

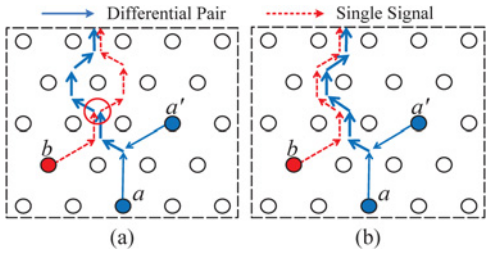


Fig. 2. Escape routing solution. (a) With mixed-crossing. (b) Without mixed-crossing.

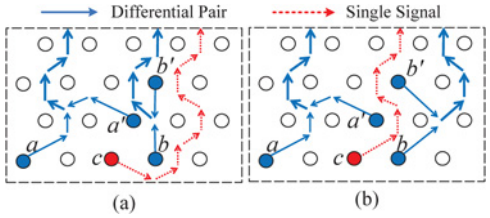


Fig. 3. Routing examples. (a) Single-pattern escape routing greedily optimizes the routing of b and b' which can lead to congestion of single signal c while (b) mixed-pattern escape routing can optimize both signals.

units of resources while each single signal takes one. Third, the two signals are required to satisfy different constraints, e.g., differential pairs should satisfy the length-matching constraint [6] while single signals need not. As a result, the mixed-pattern escape routing becomes two-commodity flow problem which has been proven to be NP-Complete. Besides, for mixed-pattern escape routing on one-routing-layer PCB, the nets of two signals may be crossed, as shown in Fig. 2(a). However, this is not allowed for mixed-pattern escape routing. For convenience, we refer to this kind of crossings as mixed crossings of differential pairs and single signals and they should be avoided in mixed-pattern escape routing.

One feasible solution for mixed-pattern escape routing is to solve differential-pair routing and single-signal routing separately. However, the routing result cannot be optimized because the routing of the two kinds of signals influence each other. As a result, the routing of one signal will probably fail if the routing resource is occupied by the other. This is hard to avoid even though congestion-aware method is used. Furthermore, it cannot avoid the mixed crossings either. Fig. 3(a) shows an example where the optimization of differential-pair routing leads to the congestion of routing resources and longer path of the single-signal routing. And a mixed-pattern escape routing, which can optimize both differential-pair routing and single-signal routing, as shown in Fig. 3(b), is required.

A. Related Work

The related work can be classified into two parts: 1) the modeling and escape routing algorithms for SPA and 2) the escape routing algorithms including differential-pair escape routing and single-signal escape routing for GPA.

On one hand, Yan and Wong [1] analyzed the properties of SPA and proposed three escape routing strategies for SPA. Ho *et al.* [3] proposed an escape routing algorithm based on SPA and showed the advantages of SPA compared to GPA. Two recent works [11], [12] proposed a multilayer escape routing algorithm on SPA and a network flow modeling for escape routing on SPA respectively. However, they did not consider differential pairs.

On the other hand, the authors in [8]–[10] proposed network flow-based escape routing algorithms on GPA. However, they only focused on the escape routing problem of single signals. A work on chip-package-board co-design [7] considered differential pairs, but it paid attention to co-design instead of the optimization of escape routing. Reference [5] proposed a negotiated congestion-based differential-pair routing but it did not take the length-matching constraint into account. A recent work [6] proposed a five-stage algorithm for escape routing of differential pairs considering the length-matching constraint. However, it is based on GPA and cannot be applied to SPA directly.

Furthermore, all of these researches were based on the single-pattern escape routing and did not consider the problem of mixed-pattern escape routing. Recently, we proposed a mixed-pattern escape routing algorithm to solve the problem [13]. However, the mixed crossings were not considered.

B. Our Contributions

In this paper, a mixed-pattern escape routing algorithm (MPERA) is proposed to solve the problem of the mixed-pattern escape routing while considering the mixed-crossing avoidance. The major contributions of this paper are summarized as follows.

1) The mixed-pattern escape routing problem (MPER) is defined and formulated. The objective of MPER is to escape all marked pins to the array boundary with minimized total wire-length while all the constraints are satisfied.
2) A three-stage algorithm of mixed-pattern escape routing on staggered pin array is proposed, including differential pair preconditioning, unified modeling and slice-based speedup. Experimental results show that it is very effective on both single-pattern escape routing and mixed-pattern escape routing.
3) A unified ILP model is formulated for mixed-pattern escape routing problem. The constraints of differential pair protection, noncrossing and length-matching [6] are also considered.
4) A novel Boolean-coding-driven algorithm is proposed to formulate the mixed-crossing constraints and transfer them into linear constraints in ILP. By solving the ILP problem with these constraints, the mixed-crossings can be avoided.
5) A slice-based heuristic method is presented to prune the variables of ILP and speed up the solving. Experimental results show that it can reduce the running time by about 77%.

The remainder of this paper is organized as follows. Section II defines the problem of MPER and the formulation. Section III describes the overview flow of the proposed mixed-pattern escape routing algorithm and the algorithm is introduced in Sections IV–VII. Experimental results are shown in Section VIII and conclusions are provided in Section IX.

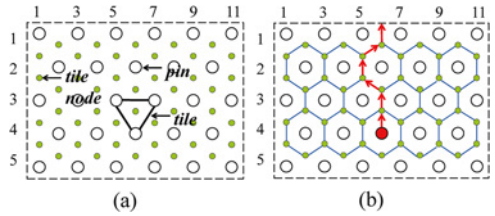


Fig. 4. Staggered pin array and a routing example via tile network. (a) Staggered pin array. (b) Routing example.

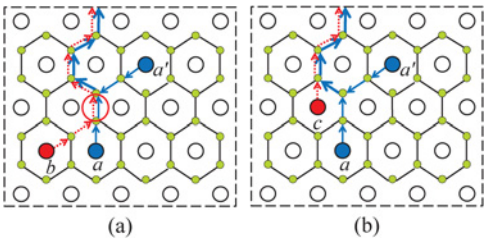


Fig. 5. Differential-pair protection constraint in MPER. The case of (a) is not allowed due to differential pair protection constraint, while (b) is a legal one.

II. Problem Definition

A. Preliminaries

Before we present the problem, the definitions for routing network are introduced first.

Definition 1 (Staggered Pin Array): An $m \times n$ staggered pin array (SPA) is composed of n rows, and in each row there are m (in odd rows) or $m-1$ pins (in even rows). A triangular tile is composed of three adjacent pins and there is a tile node in each tile, as shown in Fig. 4(a).

Definition 2 (Tile Network): The tile network is generated by connecting triangular tiles with each other in the form of hexagons, as shown in Fig. 4(b). The edges of tile hexagons are the channels for escape routing and the angle between the routing channels is 120-degree. An example of routing via tile network is shown in Fig. 4(b).

There are several constraints for mixed-pattern escape routing. First, the routing of differential pairs should satisfy the length-matching constraint [6], which means the two nets of a differential pair should be routed with similar wire lengths. Without considering this, large signal skews will be created, which can lead to degradation of performance. Second, to avoid signal crosstalk, before the two signals of a differential pair meet with each other, other signals are not allowed be close to. That is, the routing path between differential-pair pins and median points do not allow other signals to cross or go along. In this paper, the constraint is named as differential-pair protection constraint, as shown in Fig. 5. Besides, the mixed crossings should be avoided and the total wire width of all signals on each edge should not exceed the allowable capacity, which are referred as mixed-crossing-avoided constraint and wire-width constraint [1], respectively. Since the crossing of one kind of signals can be removed by exchanging each other at the meeting point, we do not consider it in our problem.

B. Problem Formulation

The problem of mixed-pattern escape routing (MPER) is defined as follows.

Problem 1 (MPER): Given: 1) an $m \times n$ staggered pin array; 2) n_d differential pairs and n_s single signals to be routed to the boundary; and 3) constraints of wire length matching of differential pairs, mixed-crossing-avoided constraint, differential-pair protection constraint and wire-width constraint, the objective is to escape all marked pins to the array boundary with minimized total wire length via the tile network. At the same time, make sure that all of the constraints are satisfied and guarantee 100% routability.

The problem can be formulated by the objective function as follows:

$$\begin{aligned} &Minimize\,,\ totalLength \\ &s.t.\ \ tskew = 0,\ \ mcrossNum = 0 \\ &\qquad dpcrossNum = 0,\ \ widthExceeding = 0 \end{aligned} \tag{1}$$

where $totalLength$ denotes the total wire-length of all signals; $tskew$ denotes the total skews of differential pairs caused by length mismatching while $mcrossNum$ denotes the total number of mixed crossings; $dpcrossNum$ denotes the total number of single signals that go across or go along with the routing paths between differential pair pins and median points; $widthExceeding$ denotes the total amount of wire widths that exceed the edge capacity over all edges. The objective of (1) is to minimize the total wire-length with no length mismatching of differential pairs, no mixed crossings and no violation against the differential-pair protection constraint and the wire-width constraint.

III. Overview Flow of Mixed-Pattern Escape Routing Algorithm

One feasible strategy to solve the problem is by using a unified model such as network flow or ILP. However, it is very difficult to take the length-matching constraint and protection constraint of differential pairs into a unified model since they are only for differential pairs. To address it, the two constraints are solved separately. Considering that the length-matching constraint only works at the beginning of differential-pair escape routing, before two signals of differential pair meet with each other, the escape routing can be divided into two stages.

1) Pin-median-point routing which aims to route from two pins of each differential pair to the median point [6], where two signals of the differential pair meet with each other with the same length.
2) Escape routing of both single signals and median points of differential pairs to boundary simultaneously.

Based on this consideration, in this paper, a three-stage mixed-pattern escape routing algorithm (MPERA) is proposed to solve the problem of MPER. The overview flow of MPERA is shown in Fig. 6. Given the routing requirement, tile network is generated first. Then, the differential pairs are handled beforehand to solve the pin-median-point routing, including median point searching, routing path finding and pin-median-point path determination. After that, a unified ILP modeling

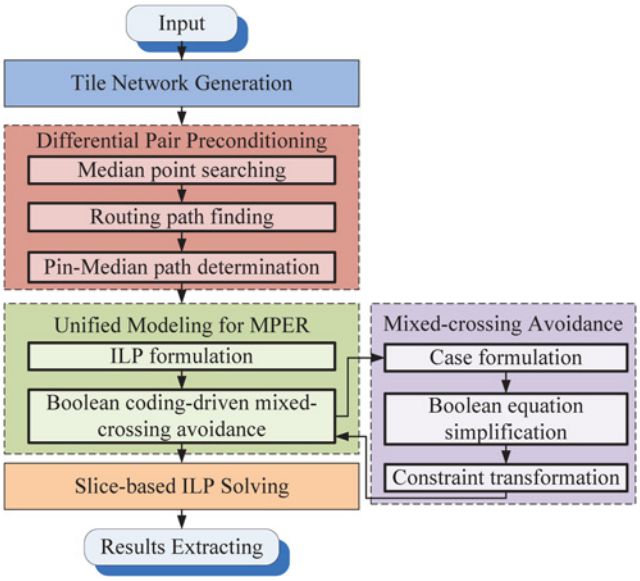


Fig. 6. Overview flow of MPERA.

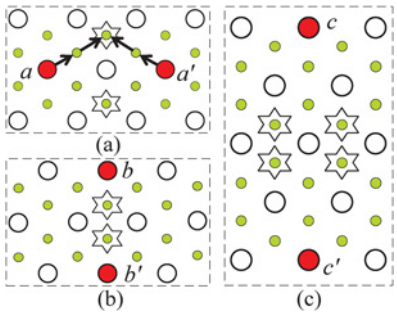


Fig. 7. Simple cases for median points searching.

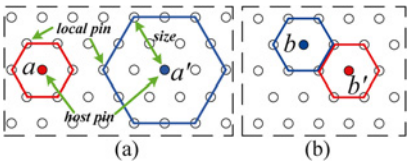


Fig. 8. Pin hexagon. (a) Examples of pin hexagon. (b) Minimum intersecting hexagons.

is proposed to formulate the problem of escape routing for both single signals and median points to boundary. A Boolean coding-driven mixed-crossing avoidance method ensures there is no mixed crossings, which contains three steps of case formulation, Boolean equation simplification and constraint transformation. Finally, a slice-based heuristic algorithm is performed to solve the ILP problem in reasonable time.

The details of MPERA are described in the following sections. Section IV introduces the differential pair preconditioning. Then, in Section V, the ILP-based unified modeling is presented to solve the simultaneous escape routing of both single signals and median points while the mixed-crossing-avoided method is proposed in Section VI. The slice-based heuristic algorithm is illustrated in Section VII.

IV. Differential Pair Preconditioning

In this section, the differential pair preconditioning is proposed to solve the problem of pin-median-point routing. The median point routing, which routes the median points to boundary, is solved with the single-signal escape routing together through a unified modeling. This will be introduced in Section V. The preconditioning is divided into three steps: 1) median point searching searches for all median points with minimum distance to the pins of each differential pair; 2) path candidate finding finds all shortest path candidates from pins to median points; and 3) path determination selects the final path from all path candidates. The detailed algorithm is introduced as follows.

A. Median Point Searching Algorithm

An effective method was proposed to find median point candidates for each differential pair [6]. However, it is based on GPA and cannot be adopted to SPA due to the different structure and different constraints. In this section, a median point searching method for staggered pin array is proposed.

Assume a and a' are the two pins of a differential pair, and (h_a,v_a) and $(h_{a'},v_{a'})$ are the coordinates of two pins where h and v are the horizontal ordinate and vertical ordinate on pin array respectively. Based on the features of staggered pin array, the median point searching can be solved according to the following six cases.

Case 1: $v_a = v_{a'}$: there are two median point candidates, which lie on the midperpendicular between two pins, as shown in Fig. 7(a).

Case 2: $h_a = h_{a'}$ and $|v_a - v_{a'}|$ is not multiple of 4; there are two min-cost median point candidates, which lie on the line between two pins, as shown in Fig. 7(b).

Case 3: $h_a = h_{a'}$ and $|v_a - v_{a'}|$ is multiple of 4; there are four median point candidates, which lie around the middle pin of the two pins, as shown in Fig. 7(c).

Before we formally investigate the following three cases, the pin hexagon and minimum intersecting hexagon are first defined:

Definition 3 (Pin Hexagon): A pin hexagon is composed of certain pins which have the same distance to this pin. The pin in the center is called as host pin while the pins around are called as local pins, as shown in Fig. 8(a). The size of a pin hexagon is determined by the distance between local pins and host pin.

Definition 4 (The Minimum Intersecting Hexagon): Given a differential pair, the intersecting hexagons of the pair are two intersecting or adjacent hexagons with the same size and the minimum intersecting hexagons are those with the minimum size, as shown in Fig. 8(b).

Based on the definitions, the following three cases can be formulated.

Case 4: $h_a \neq h_{a'}$, $v_a \neq v_{a'}$ and the minimum intersecting hexagons are adjacent: the candidates lie in each hexagon and beside the adjacent line of the hexagons, as shown in Fig. 9(a).

Case 5: $h_a \neq h_{a'}$, $v_a \neq v_{a'}$ and the minimum intersecting hexagons are intersecting to each other: the

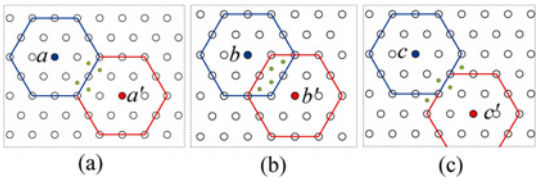


Fig. 9. Complicated cases for median points searching.

Algorithm 1 Path Finding Algorithm

```
Require: pin1, pin2, mpoint, maxLength;
Ensure: pathSet;
 1: calMaxLength(maxLength);
 2: adjTiles1 = calPinAdjTiles(pin1);
 3: adjTiles2 = calPinAdjTiles(pin2);
 4: removeRedundantTiles(adjTiles1, adjTiles2);
 5: region = calRoutingRegion(pin1, pin2);
 6: isdone = false;
 7: while isdone == false do
 8:    for each t1 ∈ adjTiles1 do
 9:       path1.push_back(t1);
10:       findPath(pathSet1, path1, t1, mpoint, region,
          maxLength);
11:       path1.pop_back();
12:    end for
13:    for each t2 ∈ adjTiles2 do
14:       path2.push_back(t2);
15:       findPath(pathSet2, path2, t2, mpoint, region,
          maxLength);
16:       path2.pop_back();
17:    end for
18:    if pathSet1 is not null and pathSet2 is not null then
19:       for each p1 ∈ pathSet1 do
20:          for each p2 ∈ pathSet2 do
21:             path = mergePaths(p1, p2);
22:             removeRedundantSubPaths(path);
23:             pathSet.push_back(path);
24:          end for
25:       end for
26:       isdone = true;
27:    else
28:       maxLength++;
29:    end if
30: end while
```

candidates lie in the intersecting region of minimum intersecting hexagons of two pins, as shown in Fig. 9(b).

Case 6: $h_a \neq h_{a'}$, $v_a \neq v_{a'}$ and the minimum intersecting hexagons are partly adjacent: the candidates lie around the middle point of two pins and beside the adjacent line of the hexagons, as shown in Fig. 9(c).

In order not to result in congestion for the routing of single signals, the median points which are close to single signal pins are not selected. In this paper, the allowed distance between median points and single signal pins is set to be at least two units of tile nodes.

B. Path Candidate Finding

Based on above cases, several median point candidates can be obtained, and then the corresponding pin-median paths can be found according to the relative positions between pins and median points. Since the pin-median-point paths occupies routing resource, they will directly influence the solution quality of mixed-pattern escape routing. Therefore, it is quite necessary to reduce the total length of pin-median-point paths. In this paper, a path finding algorithm is proposed to find paths candidates with length-matching constraint, as shown in Algorithms 1 and 2. The length of one path is evaluated as the number of tile nodes that the path goes through.

Algorithm 2 *findPath*(*pSet*, *path*, *tile*, *mp*, *region*, *l*)

```
Require: pSet, path, tile, mp, region, l;
 1: if l ≤ 1 then
 2:    if tile == mp then
 3:       pSet.push_back(path);
 4:    end if
 5: else
 6:    adjTiles = calTileAdjTiles(tile);
 7:    for each tile ∈ adjTiles do
 8:       if tile ∈ region and tile ∉ path then
 9:          path.push_back(tile);
10:          findPath(pSet, path, tile, mp, region, l − 1);
11:          path.pop_back();
12:       end if
13:    end for
14: end if
```

To be specific, first, calculate the acceptable maximum length of path according to the positions of pins and find the adjacent tile nodes around two pins (line 1–5). Then, for each adjacent tile node of the pin, invoke the function *findPath* to find paths with the length of *maxLength* (line 8–12); this is also done for the other pin (line 13–17). After that, merge the two path sets and generate the paths from one pin to the other via median point (line 18–26). If there is no path of *maxLength*, then try another value of *maxLength* added by one (line 27–29). Finally, the obtained paths are treated as path candidates and the following section introduces how to select the finally used path from the candidates.

C. Pin-Median-Point Path Determination

1) *Path Weight Calculation:* Though a lot of path candidates are found in the previous section, some of them are not legal. It is easy to find that based on SPA, the acute-angle (60-degree) paths are possibly generated, especially for the differential pairs whose pins are close to each other. The acute-angle routing results can reduce the strength of signals and even cause undercutting or over etching of the circuitry. Therefore, it is necessary to reduce the numbers of acute-angle routing. In this paper, we bring in path weight for the selection of path candidates. For each path p, if p has acute-angles, then the weight w_p is set to 1, otherwise, w_p is set to 0. The lower weight a path possesses, the higher possibility it will be selected as a routing solution. Fig. 10(a) shows an example. There are three path candidates with length of six for this case. p_2 and p_3 have acute-angles while p_1 has not. Therefore, we set $w_{p_1} = 0$, $w_{p_2} = 1$ and $w_{p_3} = 1$. These weights are important criterions for the path determination. By this method, the acute-angles can be removed by about 50%. For the rest unavoidable cases, the 60-degree angle can be split into two 120-degree angles by adding an additional segment with a little wire length sacrificed, as shown in Fig. 10(b).

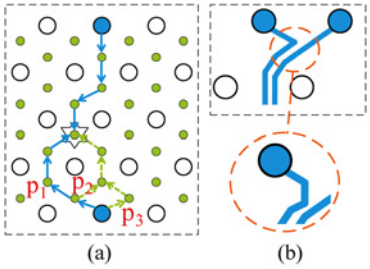


Fig. 10. (a) Acute-angle avoided path selections. (b) The rest unavoidable acute-angles can be split into 120-degree angles by adding additional segments.

2) *ILP Based Path Determination:* After path weight calculation, the next step is to determine the path to be used from the candidates. For this, differential pairs are classified into K groups according to the crossing possibility of path candidates while K is the maximum value that makes the paths in groups without crossing with each other. A similar method as [6] can be used to solve it. After that an ILP formulation is used to determine pin-median-point paths for all differential pairs in each group. Note that once the path is determined, the corresponding median point is also determined.

Assume that each group G_k contains a_k differential pairs, and for each pair i, there are n_{ki} path candidates. Assume l_p denotes the sum of length of path p from differential pair pin to median point and the distance of median point to the nearest boundary. The objective is to minimize the total path length and total number of acute-angles.

For group G_k, binary decision variables φ_{ip} ($1 \le i \le a_k$, $1 \le p \le n_{ki}$) are defined such that φ_{ip} is 1 if path p is selected for differential pair i, and φ_{ip} is 0 otherwise. For each differential pair, one single path is assigned, so

$$\sum_{p=1}^{n_{ki}} \varphi_{ip} = 1 \; s.t. \; \varphi_{ip} = 0 \; or \; 1, \forall 1 \le i \le a_k. \tag{2}$$

Let PCC_k stand for the set of crossed paths in group G_k. Then, for each crossed pair of $path_{ip}$ and $path_{jq}$ in G_k, φ_{ip} and φ_{jq} should satisfy

$$\varphi_{ip} + \varphi_{jq} \le 1, (ip, jq) \in PCC_k. \tag{3}$$

Based on (2) and (3), the objective can be written by

$$Min \quad \sum_{i=1}^{a_k} \sum_{p=1}^{n_{ki}} (\varphi_{ip} \cdot l_p + M_2 \cdot \varphi_{ip} \cdot w_p) \tag{4}$$

where w_p is the weight of path p and M_2 is a penalty item. In this paper, M_2 is set as 100. By solving the ILP problem, the median points for differential pairs and the shortest paths for pins to median points can be determined with acute-angle avoidance considered.

V. Unified Modeling of Mixed-Pattern Escape Routing

After differential pair preconditioning, the problem of escape routing of both single signals and median points can be solved together by a unified modeling.

TABLE I
NOTATIONS OF MPER

Parameters of nodes on network graph	
T_D	The set of determined median point of differential pairs
P_S	The set of escaped single signal pins
T_S	The set of tile nodes adjacent to single signal pins
$Tile$	The set of routing tile nodes
$STile$	The set of routing tile nodes adjacent with single signals
S_0	The source node of total network
S_D	The source node for differential pair
S_S	The source node for single signal pins
S_t	The sink node of total network
$Path_{DP}$	The set of tile nodes which have been occupied by pin-median routing of differential Pair
Parameters of edges on network graph	
E	The set of all edges
e_{S_0,S_D}	The directed edge from S_0 to S_D
e_{S_0,S_S}	The directed edge from S_0 to S_S
e_{S_D,t_i}	Directed edge from S_D to a median point t_i
e_{S_S,p_i}	Directed edge from S_S to a single signal pin p_i
e_{p_i,t_i}	Directed edge from signal pin p_i to nearby tile t_i
e_{t_i,t_j}	Directed edge from an tile node t_i to another t_j
e_{t_i,S_t}	Directed edge from boundary tile t_i to sink node
Parameters of constraints	
$l(e_{i,j})$	The length of edge $e_{i,j}$
$f_d(e_{i,j})$	The flow of differential pair on edge $e_{i,j}$
$f_s(e_{i,j})$	The flow of single signals on edge $e_{i,j}$
$f(e_{i,j})$	The total flow on edge $e_{i,j}$
$c(e_{i,j})$	The capacity of edge $e_{i,j}$
$Integer$	The set of integers

A network flow based ILP formulation method was proposed for single-signal routing problem [3]. However, it did not consider the differential pairs. In this paper, we propose an ILP-based modeling for mixed-pattern escape routing of both signals. Different from single-pattern escape routing, for mixed-pattern escape routing, there are two kinds of input sources including both differential pairs and single signals and as the two kinds of signals take different network resources, more constraints should be brought in to distinguish them. Before we present the ILP formulation, some notations are defined for the routing network, as shown in Table I.

Thereafter, the problem can be formulated as the following ILP objective function:

$$Min \quad \omega \times \sum_{e_{i,j} \in E} l(e_{i,j}) \times f_d(e_{i,j}) + \sum_{e_{i,j} \in E} l(e_{i,j}) \times f_s(e_{i,j}) \tag{5}$$

subject to

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = 1, \; \forall i \in T_D \bigcup P_S \tag{6}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |T_D|, \; \forall i \in S_D \tag{7}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |P_S|, \; \forall i \in S_S \tag{8}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |T_D| + |P_S|, \; \forall i \in S_0 \tag{9}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |T_D| + |P_S|, \; \forall j \in S_t \tag{10}$$

$$\sum_{e_{i,j} \in E} f_d(e_{i,j}) = \sum_{e_{j,k} \in E} f_d(e_{j,k}), \; \forall j \in Tile \tag{11}$$

$$\sum_{e_{i,j} \in E} f_s(e_{i,j}) = \sum_{e_{j,k} \in E} f_s(e_{j,k}), \; \forall j \in T_S \bigcup Tile \tag{12}$$

$$2 \times f_d(e_{i,j}) + f_s(e_{i,j}) \le c(e_{i,j}) \tag{13}$$

$$\sum_{e_{i,j} \in Path_{DP}} f_s(e_{i,j}) \le 0 \tag{14}$$

$$f_d(e_{i,j}) \ge 0, \;\; f_s(e_{i,j}) \ge 0 \tag{15}$$

$$f_d(e_{i,j}), \;\; f_s(e_{i,j}) \in Integer. \tag{16}$$

The objective of (5) is to minimize the total wire-length including both differential pairs and single signals. ω is used to adjust the relative weights between differential pairs and single signals, and in this paper, ω is set to 2. Three source nodes are defined to drive the flows. The S_0 node is used for source of all flows and two subsource nodes S_D, S_S are used for differential pairs and single signals, respectively. For each edge, there are two kinds of flows, $f_d(e_{i,j})$ for differential pairs and $f_s(e_{i,j})$ for single signals. Constraints (6)–(10) ensure the 100% routability of each flow by forcing all escaped pins to be routed to the sink and constraints (11) and (12) ensure that each flow satisfies the flow conservation constraint. Constraints (13) and (14) guarantee the wire width constraint and differential pair protection constraint. Constraints (15) and (16) make the flows on all edges nonnegative integers.

Considering that the multicommodity problem can be modified to single commodity problem by adding some constraints, the problem can be divided to the following three cases.

A. Modified MPERA for Single-Pattern Escape Routing

Considering the single-pattern escape routing for either differential pairs or single signals, the problem can be transformed to the LP problem in [3] and can be solved in polynomial time.

B. Modified MPERA Considering Crosstalk Between Single Signals

Considering the crosstalk issues between single signals, the single signals are not allowed to share the routing channels because the close distance can lead to crosstalk. On the other hand, differential pairs can share the same routing channel as the signals of differential pair are protected by themselves. Especially for the tile capacity of 2, the constraint of (13) can be transformed into

$$f_d(e_{i,j}) + f_s(e_{i,j}) \le 1. \tag{17}$$

Then, all the constraints satisfy the unimodularity property [16] and the problem can be transformed to LP formulation without constraint (16) because the optimal solution is guaranteed to be an integer solution according to unimodularity property. The LP problem can be solved in polynomial time.

C. MPERA Not Considering Crosstalk Between Single Signals

As complement to the above two cases, if the crosstalk issues between single signals are not considered, the problem does not satisfy the unimodularity property any more. Then it will be solved by the ILP solver. In this paper, a slice-based heuristic method is proposed to speed up the algorithm, which will be introduced in Section VII.

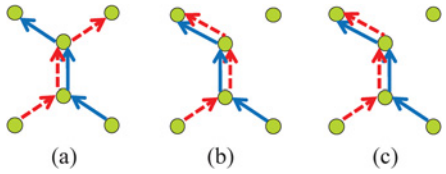


Fig. 11. Three kinds of mixed crossing. (a) Signals are crossed with each other when they separate after sharing a segment together. (b) Signals are crossed after meeting with each other at the meeting point. (c) Two signals exchange their relative positions during the routing path to boundary.

VI. Boolean Coding-Driven Mixed-Crossing Avoidance

If the capacity of edges in the tile network is no more than two, as a result of the limitation of routing resource, at most one kind of signals is allowed on each edge and hence the ILP formulation ensures that there is no mixed crossing.

However, when the capacity of each edge is larger than two, the two kinds of signals may be mixed up. There are three kinds of mixed crossings: 1) crossed when the two signals separate with each other after sharing a segment together, as shown in Fig. 11(a); 2) crossed when the two signals meet with each other at a meeting point, as shown in Fig. 11(b); and 3) the relative position relationship of two signals changed during the routing to boundary, as shown in Fig. 11(c). These cannot be avoided by the ILP formulation in Section V because it cannot catch the relative position relationship of two kinds of signals on the same edge.

To address the problem, in this paper, a novel Boolean coding-driven method is proposed to catch the relative position relationship of signals and avoid the mixed crossings, as shown in Algorithm 3. There are three kinds of tile nodes: 1) normal ones with three adjacent tile nodes; 2) normal ones with two or less adjacent tile nodes, e.g. the node on the boundary; and 3) special tile nodes adjacent with single pins, where there may be another input flow on the edge from single pin to this node. For each normal tile node t with three adjacent tile nodes, the adjacent tile nodes and all mixed-crossing cases are first found (line 1–5). Then, for each case, a Boolean equation is used to formulate it and the obtained Boolean equation is transferred into linear constraints by Boolean algebra operations and inequality substitution (line 6–10). If the tile node is adjacent with two nodes, we just need to set the Boolean variables of adjacent edges to be the same values (line 11–12). If the tile node is special one adjacent with single pins, then the mixed-crossing cases finding will be solved separately and the rest steps are the same with normal ones (line 16–23). Finally, all constraints are integrated into the ILP problem in Section V and solved together (line 25–26).

The details of the algorithm are introduced in the following sections. First, we introduce the Boolean coding of mixed-crossing and then we show how to use Boolean equations to formulate a case with mixed crossings, and how to transfer the equations to linear constraints in ILP. Finally, we enumerate all mix-crossing cases and provide the corresponding linear constraints. However, for the cases with large capacity, the number of cases of mixed crossings will increase dramatically.

Algorithm 3 Mixed-crossing Avoidance

```
for each t ∈ Tiles do
  if t ∉ STiles then
    tileNum = findAdjtiles(t);
    if tileNum ≥ 3 then
      caseSet = findCrossingCases(t);
      for each case ∈ caseSet do
        caseFormulation(case);
        BooleanEqSimplification(case);
        transferToLinearConstraint(case);
      end for
    else if tileNum == 2 then
      setAdjedgesState();
    else
      continue;
    end if
  else
    caseSet2 = findAllCrossingCasesWithSingle(t);
    for each case2 ∈ caseSet2 do
      caseFormulation(case2);
      BooleanEqSimplification(case2);
      transferToLinearConstraint(case2);
    end for
  end if
end for
addConstriantsToILP();
solveILP();
```

In this paper, we focus on the case with edge capacity of three and show how to formulate the mixed crossing and how to avoid them in ILP. For the cases with capacity of larger than three, the Boolean coding-driven method can be used similarly.

A. Boolean Coding of Mixed-Crossing

For each directed edge e in the tile network, a four-Boolean-variable group is defined: $e(e_d, e_s, e_b, e_f)$. e_d denote if there are differential pairs on edge e while e_s denote if there are single signals. If $e_d = 1$, there will be at least one differential pair on the edge of e, and similarly if $e_s = 1$, there will be at least one single signal on e.

To denote the relative positions of two kinds of signals, another two flag variables e_b and e_f are defined for each edge. The opposite edge of e is also defined as e'. $e_b = 1$ if and only if $e_d = 1$ and $e_s = 1$ and the differential pair is on the right of the single signal along the direction of differential pair; otherwise, $e_b = 0$. $e_f = 1$ if and only if $e_d = 1$ and $e'_s = 1$ and the differential pair is on the right of the single signal. Otherwise, $e_f = 0$. These can be formulated as the following Boolean equations:

$$e_d \cdot e_s + \bar{e_b} = 1 \tag{18}$$

$$e_d \cdot e'_s + \bar{e_f} = 1 \tag{19}$$

where $+$, $\cdot$ and $\bar{}$ denote OR, AND and NOT in Boolean operations, respectively.

For each tile node t, there are at most three adjacent tile nodes and six adjacent edges. Without loss of generality, assume the three tile nodes are t_x, t_y, and t_z, and the six edges are x, y, z, x', y', z', as shown in Fig. 12. The former three edges denote the ones flowing into the node while the latter three denote the ones flowing out of the node. Then, for each directed edge, a four-Boolean variable group is used, as shown in Fig. 12.

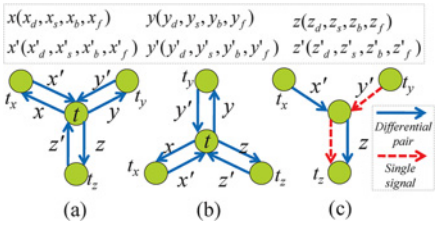


Fig. 12. Boolean coding for each tile group. (a) Triangular structure of four tile nodes. (b) Inverted triangular structure of four tile nodes. (c) Routing example with mixed crossing.

B. Example of Mixed-Crossing Avoidance Method

Based on the Boolean coding, a three-stage method is proposed to describe each routing case. First, case formulation describes each case with a Boolean equation. Then, equation simplification simplifies the equations by removing the redundant Boolean variables. After that, constraint transformation transfers the Boolean equations into linear constraints in ILP.

Take Fig. 12(c) for example and we show the details of the process as follows.

1) *Case formulation:* According to Fig. 12(c), the Boolean variable groups can be assigned with the corresponding values as $x(0,0,0,0)$, $x'(1,0,0,0)$, $y(0,0,0,0)$, $y'(0,1,0,0)$, $z(1,1,0,0)$ and $z'(0,0,0,0)$. We can use the following Boolean equation to stand for:

$$\begin{aligned} &\bar{x_d} \cdot \bar{x_s} \cdot x'_d \cdot \bar{x'_s} \cdot \bar{x_b} \cdot \bar{x_f} \cdot \bar{x'_b} \cdot \bar{x'_f} \cdot \\ &\bar{y_d} \cdot \bar{y_s} \cdot \bar{y'_d} \cdot y'_s \cdot \bar{y_b} \cdot \bar{y_f} \cdot \bar{y'_b} \cdot \bar{y'_f} \cdot \\ &z_d \cdot z_s \cdot \bar{z'_d} \cdot \bar{z'_s} \cdot \bar{z_b} \cdot \bar{z_f} \cdot \bar{z'_b} \cdot \bar{z'_f} = 1. \end{aligned} \tag{20}$$

2) *Boolean equation simplification:* Although the description is exact, some Boolean variables are not necessary in the equation. For example, since each flow satisfies the flow conservation constraint, then y'_s becomes a redundant variable and can be removed because $z_s = 1$ and $x'_s = 0$. $z_b = 0$ is needed because the differential pair is on the left of single signal while x'_b and y'_b can be removed because $x'_s = 0$ and $y'_d = 0$. Besides, $x_d = 0$ can be removed because $x'_d = 1$ due to the capacity constrains in ILP. Similarly, the other redundant variables can be removed. After that, (20) is replaced by the following equation containing five key variables:

$$x'_d \cdot \bar{x'_s} \cdot z_d \cdot z_s \cdot \bar{z_b} = 1. \tag{21}$$

If we set the Boolean expression as

$$x'_d \cdot \bar{x'_s} \cdot z_d \cdot z_s \cdot \bar{z_b} = 0 \tag{22}$$

then the case in Fig. 12(c) can be avoided, which can be also written as

$$\bar{x'_d} + x'_s + \bar{z_d} + \bar{z_s} + z_b = 1. \tag{23}$$

3) *Constraint Transformation:* Consider that the capacity of each edge is 3, then for edge e, the corresponding Boolean variables e_d, e_s, e_b, e_f can be assigned with practical values $\{0, 1\}$, $\{0, 1, 2, 3\}$, $\{0, 1\}$ and $\{0, 1\}$, respectively, and the

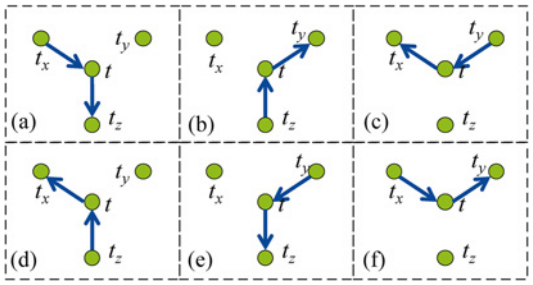


Fig. 13. Six possible cases that differential pair go through (a) $x \longrightarrow z$, (b) $z \longrightarrow y$, (c) $y \longrightarrow x$, (d) $z \longrightarrow x$, (e) $y \longrightarrow z$, and (f) $x \longrightarrow y$.

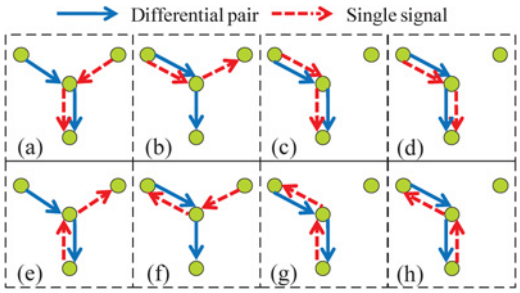


Fig. 14. All possible mixed-crossing cases of $x \longrightarrow z$.

Boolean equations can be transferred into linear constraints according to following rules:

1) each item ε remains unchanged;
2) for $\forall \varepsilon \in [0, 1]$, the items $\bar{\varepsilon}$ are changed to the $(1 - \varepsilon)$;
3) for $\forall \varepsilon \in [0, 3]$, the items $\bar{\varepsilon}$ are changed to the $(1 - \varepsilon/3)$;
4) the '=' sign is changed to the $\geq$.

For example, the Boolean (23) can be transferred into

$$(1 - x'_d) + x'_s + (1 - z_d) + (1 - z_s/3) + z_b \geq 1 \tag{24}$$

where $x'_d \in \{0, 1\}$, $x'_s \in \{0, 1, 2, 3\}$, $z_d \in \{0, 1\}$, $z_s \in \{0, 1, 2, 3\}$, and $z_b \in \{0, 1\}$.

By solving ILP with this constraint, the case of Fig. 12(c) can be avoided.

C. Case Enumeration

In this section, we enumerate all possible mixed-crossing cases and their corresponding constraints which can be obtained according to the above method.

Assume $x \longrightarrow z$ denotes that there is a differential pair going through tile node t from t_x to t_z. Then, for each normal tile node t, there are six possible cases according to the path of differential pair, that is $x \longrightarrow z$, $z \longrightarrow x$, $y \longrightarrow x$, $x \longrightarrow y$, $z \longrightarrow y$, and $y \longrightarrow z$, as shown in Fig. 13. As the six cases are highly symmetrical, we only need to analyze one case and the other five will be obtained similarly. For example, if we get the cases of $x \longrightarrow z$, we can get cases of $z \longrightarrow x$ by altering the directions of all the edges of $x \longrightarrow z$ and get the cases of $y \longrightarrow x$ by rotating $x \longrightarrow z$ by 120 degree clockwise around the center of t. Here, we select $x \longrightarrow z$ as research object and Fig. 14 shows that there are eight cases of it.

However, there are still some special nodes. First, some tile nodes have only one adjacent tile node and no adjacent signal pins. Since the tile node will not be any routing channel, and hence no treatment is needed on them.

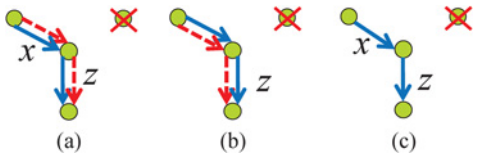


Fig. 15. Possible cases of tile node connected with two adjacent tiles. (a) $x'_b = 1$, $x'_f = 0$ and $z_b = 1$, $z_f = 0$. (b) $x'_b = 0$, $x'_f = 0$ and $z_b = 0$, $z_f = 0$. (c) $x'_b = z_b = 0$, $x'_f = z_f = 0$.

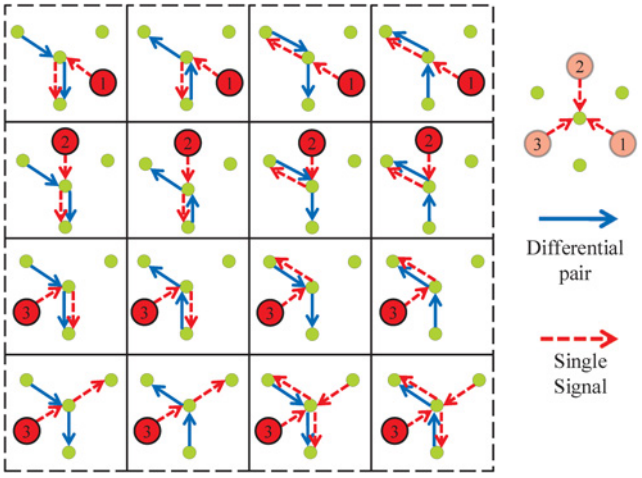


Fig. 16. Mixed-crossing cases of $x \longrightarrow z$ and $z \longrightarrow x$ caused by the injection of single signals from three directions.

Second, some tile nodes are connected with two adjacent tile nodes. If the third tile node is along the median-pair-path which is not available according to the differential-pair protection constraint, as shown in Fig. 15, then we just set

$$x'_b = z_b, \;\; x_b = z'_b, \;\; x'_f = z_f \;\; , \;\; x_f = z'_f. \tag{25}$$

Besides, some tile nodes are adjacent with three tile nodes and also connected with single pins. In this case, we need to treat separately. As shown in Fig. 16, there are three possible positions of single pins. For $x \longrightarrow z$ and $z \longrightarrow x$, there are 16 cases with mixed crossings. Some cases are not considered since they can be transferred to the existing cases in Figs. 14 and 16. Take the first one for example, and the key Boolean variables are

$$x'_d = 1, x'_s = 0, z_d = 1, z_s = 1, z_b = 0, f_1 = 1 \tag{26}$$

where f_1 denotes if the single pin selects the tile as outflow node. We can use the following Boolean equation to stand for the case:

$$x'_d \cdot \bar{x'_s} \cdot z_d \cdot z_s \cdot \bar{z_b} \cdot f_1 = 1. \tag{27}$$

If we want this case to be avoided, we just need to set the expression to be 0

$$x'_d \cdot \bar{x'_s} \cdot z_d \cdot z_s \cdot \bar{z_b} \cdot f_1 = 0 \tag{28}$$

which can be also written as

$$\bar{f_1} + \bar{x'_d} + x'_s + \bar{z_d} + \bar{z_s} + z_b = 1. \tag{29}$$

For the special tile nodes which connect two tile nodes and that are adjacent with single pins, we just need to set the values of all variables on the missing edge to be 0 and use the same constraints as the ones with three adjacent tile nodes.

In this paper, we do not enumerate all the processes of the Boolean coding-driven method adopted on each case. We just give the final linear constraints for each tile node. Actually, for each normal tile node, there are 48 linear constraints to avoid mixed crossing. For $x \rightarrow z$ and $z \rightarrow x$, there are 16 cases as follows:

$$x'_b + z_d + z_s/3 - z_b \leq 2 \tag{30}$$

$$z'_b + x_d + x_s/3 - x_b \leq 2 \tag{31}$$

$$z_b + x'_d + x'_s/3 - x'_b \leq 2 \tag{32}$$

$$x_b + z'_d + z'_s/3 - z'_b \leq 2 \tag{33}$$

$$z_d + z_s/3 + x'_d - z_b - x'_s \leq 2 \tag{34}$$

$$x'_d + x'_s/3 + z_d - x'_b - z_s \leq 2 \tag{35}$$

$$z'_b + x_d - x_s \leq 1 \tag{36}$$

$$x_b + z'_d - z'_s \leq 1 \tag{37}$$

$$x'_d + z_d + z'_s/3 - x_s - z_f \leq 2 \tag{38}$$

$$x'_d + z_d + x_s/3 - z'_s - x'_f \leq 2 \tag{39}$$

$$z'_f + x_d - x'_s \leq 1 \tag{40}$$

$$x_f + z'_d - z_s \leq 1 \tag{41}$$

$$x'_f + z_d + z'_s/3 - z_f \leq 2 \tag{42}$$

$$z_f + x'_d + x_s/3 - x'_f \leq 2 \tag{43}$$

$$z'_f + x_d + x'_s/3 - x_f \leq 2 \tag{44}$$

$$x_f + z'_d + z_s/3 - z'_f \leq 2. \tag{45}$$

The rest 32 constraints for $y \rightarrow x$, $x \rightarrow y$, $z \rightarrow y$, and $y \rightarrow z$ can be obtained similarly.

For the special tile node adjacent with a single pin, not only the above 48 linear constraints but also 48 more constraints should be considered. Let f_1, f_2, and f_3 denote the three possible positions between z and y, y and x and x and z as shown in Fig. 16. For $x \rightarrow z$ and $z \rightarrow x$, the following constraints should be satisfied:

$$f_1 + x'_d + z_d + z_s/3 - z_b - x'_s \leq 3 \tag{46}$$

$$f_1 + x_d + z'_f - x'_s \leq 2 \tag{47}$$

$$f_1 + x'_d + z_d + x_s/3 - z'_s - x'_f \leq 3 \tag{48}$$

$$f_1 + x_b + z'_d - z'_s \leq 2 \tag{49}$$

$$f_2 + x'_d + z_d + z_s/3 - z_b - x'_s \leq 3 \tag{50}$$

$$f_2 + x_d + z'_f - x'_s \leq 2 \tag{51}$$

$$f_2 + x'_d + z_d + x_s/3 - z'_s - x'_f \leq 3 \tag{52}$$

$$f_2 + x_b + z'_d - z'_s \leq 2 \tag{53}$$

$$f_3 + z_b + x'_d - x_s \leq 2 \tag{54}$$

$$f_3 + z'_d + x_d + z_s/3 - x_s - z'_f \leq 3 \tag{55}$$

$$f_3 + z_d + x'_f - z_s \leq 2 \tag{56}$$

$$f_3 + z'_d + x_d + x_s/3 - z_s - x_b \leq 3 \tag{57}$$

$$f_3 + z_d + x'_d - x_s - z_s \leq 2 \tag{58}$$

$$f_3 + z'_d + x_d - z_s - x_s \leq 2 \tag{59}$$

$$f_3 + z_b + x'_f + y'_s/3 \leq 3 \tag{60}$$

$$f_3 + x_d + z'_d + y'_s/3 - z'_f - x_b \leq 3. \tag{61}$$

The constraints for $y \rightarrow x$, $x \rightarrow y$, $z \rightarrow y$, and $y \rightarrow z$ can be obtained similarly since they are highly symmetrical.

Furthermore, for each e, the following constraints should be added to satisfy (18) and (19):

$$e_d + 3e_s - 4e_b \geq 0 \tag{62}$$

$$e_d + 3e'_s - 4e_f \geq 0. \tag{63}$$

After that, the constraints are integrated into the ILP formulation in Section V and solved together.

VII. Slice-based MPERA Without Considering Crosstalk Between Single Signals

Directly solving the ILP problem is computationally expensive since ILP is NP-complete [14]. To address the problem, we present a heuristic method to reduce the number of variables and constraints.

As it is well known, when a stream of water falls down to the ground, the water will spread to all around and if the ground is smooth, the spread will be even enough. Inspired by this physical phenomenon, if the pins of differential pairs and single signals are uniform distributed, the routing paths will be evenly sent out to the boundary, that is, the escape routing satisfies divergence, which can be observed from existing work, such as [3]–[6]. In this paper, we refer the property as divergence property of escape routing.

TABLE II
EFFECTIVENESS OF OUR METHOD ON SINGLE-PATTERN ESCAPE ROUTING ON STAGGERED PIN ARRAY

Benchmark information			[5]			Ours			
Benchmark	Dpair num	Array size	Avg. len.	Run-time (s)	Area	Pin Array len.	Avg. len.	Run-time (s)	Area
case1	10	11x8	2.7	≤1	70	2.67	2.31	≤1	60.62
case2	18	22x7	4.0	≤1	126	4.00	3.46	≤1	109.12
case3	18	18x12	4.8	≤1	187	4.74	4.10	≤1	161.95
case4	8	11x3	2.0	≤1	20	2.00	1.15	≤1	17.32
case5	11	14x3	2.6	≤1	26	2.60	1.52	≤1	22.52
case6	11	17x6	4.1	≤1	80	4.18	3.62	≤1	69.28
case7	18	17x6	3.0	≤1	80	3.04	2.63	≤1	69.28
case8	20	9x16	3.5	≤1	120	3.50	3.03	≤1	103.92
case9	20	8x15	3.3	≤1	98	3.30	2.86	≤1	84.87
case10	60	35x35	12.2	≤1	1156	13.50	11.68	≤1	1001.13
Ratio			1.161	1	1.155	1.198	1	1	1

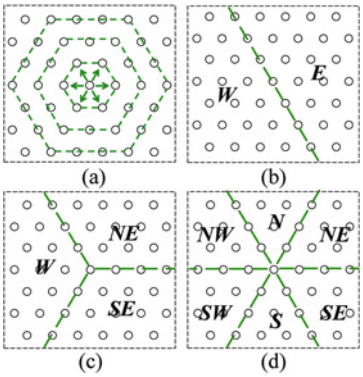


Fig. 17. Region division according to (a) symmetrical characteristic of SPA, the chip can be partitioned into (b) two regions, (c) three regions, and (d) six regions.

Considering divergence property, the problem can be divided into several subproblems and solved individually. Specifically, for the case of SPA, the whole region can be divided into 2, 3, or 6 subregions according to the characteristics of the hexagons, as illustrated in Fig. 17. Due to this, for each subproblem, the variables of ILP can be reduced by at least 50% to 87%.

The detailed algorithm is shown in Algorithm 4. Initially, the pin array and tile network are generated (line 1–2). Then, the chip is partitioned into *apartNum* regions (line 3), each of which will solve the ILP independently. Some initialization is done to store the information of pin array and tile array in each region (line 7). Then, differential pairs and single signals are also classified into each region according to coordinates of median point and pin, respectively (line 8–9). Correspondingly, the IDs of pins are also remapped to a new set (line 10). After that, edges are generated for the network flow based ILP formulation (line 11). Then, an ILP solver is performed to solve the problem (line 12). If the ILP is optimally solved for all regions, then go to end. However, the region is not always equally partitioned. As a result, some regions may be congested and not able to get a feasible solution. To address it, the failed signals in congested regions are redistributed heuristically into the nearby region which has the most routing resource, until the problem is solved successfully (line 15–19). Finally, the results in each region are merged and final routing results are stored (line 21–22).

Algorithm 4 Variable Pruning for ILP

Require: *width*, *height*, *sinpins*, *dpairs*, *apartNum*;
1: *generatePinArray*();
2: *generateTileArray*();
3: *createRegions*(apartNum);
4: *isdone* = *false*;
5: **while** *isdone* == *false* **do**
6: **for** each $i \in [1, apartNum]$ **do**
7: *initalization*(*i*);
8: *generateDpair*(*i*);
9: *generaterSinpins*(*i*);
10: *IDCoorMapping*(*i*);
11: *createEdges*(*i*);
12: *ILPEscapeRouting*(*i*);
13: *resultsExtracting*(*i*);
14: **end for**
15: **if** ILP is optimized solved for all regions **then**
16: *isdone* = *true*;
17: **else**
18: *redistributeFailurePoints*();
19: **end if**
20: **end while**
21: *mergeResults*();
22: *outputAllPaths*();

VIII. EXPERIMENTAL RESULTS

We perform four experiments to show the effectiveness of the proposed algorithm. All experiments are implemented in C++ on a workstation with Intel Xeon 2.40GHz CPU and 12GB physical memory. *gurobi optimizer* [17] is used for the ILP solving. One routing layer is considered and the wirelengths of experimental results are all normalized values.

A. Effectiveness of MPERA on Single Pattern on Staggered Pin Array

In this section, we show the effectiveness of the proposed escape routing algorithm on single-pattern escape routing based on staggered pin array.

In Table II, *Dpair num* denotes the number of differential pairs. *Array size* shows the size of the staggered pin array. *Avg.len* shows the average number of tiles from two pins to the boundary. To make the comparison fairly, we create ten similar grid pin arrays according to the test cases in [6] and generate the ten benchmarks by shifting specific columns of the grid pin arrays. For the benchmarks with larger scale or more escaped pins, the escape routing problem can be

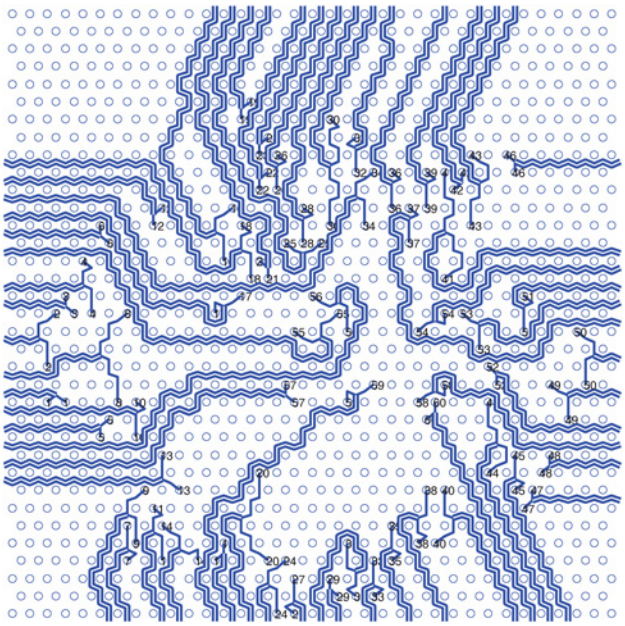


Fig. 18. Escape routing results of case10.

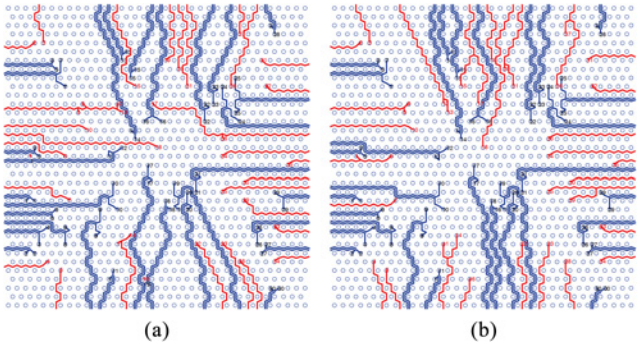


Fig. 19. Escape routing results of case 10-2. (a) Two-stage network algorithm. (b) Proposed algorithm.

partitioned into several subproblems by using multilayers. In this paper, we only focus on the escape routing problem on single layer as in [6]. Table II shows that the error of our generated grid pin array compared to [6] is only 3.7%. *Runtime*(*s*) gives the running time for routing in second and *Area* gives the total chip area based on the pin array pattern.

The proposed algorithm is very effective on differential-pair escape routing. It can solve the routing problem in quite short time. Furthermore, compared to traditional pin array, the staggered pin array can reduce wire length by about 16.1% on average and chip area by 15.5%. Fig. 18 shows the routing result of case 10.

B. Effectiveness of MPERA on Mixed-Pattern Escape Routing

Table III shows the effectiveness of the proposed algorithm on mixed-pattern escape routing. We also implement a two-stage network flow-based routing algorithm named TS to be a competitor to show the advantage of our method. Specifically, TS is to solve differential-pair routing and single-signal routing, respectively.

Since we do not have any industry benchmark of mixed-pattern escape routing, we generate fourteen benchmarks.

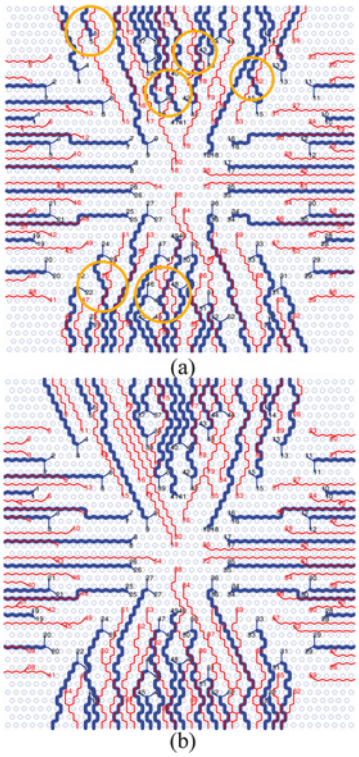


Fig. 20. Escape routing results of case-b-4 (a) without mixed-crossing avoidance, the paths of differential pairs and single signals may be crossed. The circles in figure show some of the mixed crossings and (b) with the proposed mixed-crossing-avoided method, the mixed crossings can be avoided.

Sin.num denotes the number of single signals. From Table III, we can see that TS cannot solve all test cases, while the proposed method can solve all single-signal escape routing. The routability is increased by about 17.5%. For large benchmarks, the improvement is much greater. Furthermore, for the routed pins, the proposed method can also reduce the average wire length of single signals by 14.1% on average and 26.0% at most. Case-b-4 is a little violated because there are too many failed single pins which are mainly in the middle of the chip. At the same time, the proposed method can simultaneously solve both differential pairs and single signals in short time.

Fig. 19 shows the escape routing result of case 10-2 using two-stage network algorithm [Fig. 19(a)] and the proposed algorithm [Fig. 19(b)]. The thick lines in figure show the routing path of differential pairs and the thin lines shows the path of single signals.

C. Effectiveness of the Proposed Mixed-Crossing Avoidance Method

In Table IV, the effectiveness of the proposed mixed-crossing-avoided algorithm is tested. Without mixed-crossing avoidance, all test cases have at least seven mixed crossings, while the proposed method can avoid all mixed crossings with only 2.8% increase on the length of single signals. Even for the edges where both signals share, the relative position relationship remains unchanged. As a result of the added constraints to avoid mixed crossings, the ILP solving is longer for mixed-crossing-avoided algorithm. However, it can still be solved in three minutes for case-b-4.

TABLE III

COMPARISON OF TWO-STAGE ALGORITHM AND OUR ALGORITHM ON MIXED-PATTERN ESCAPE ROUTING

Benchmark information					Two-stage method				Mixed-pattern algorithm			
Benchmark	Dpair num	Sin. num	Array size	Capacity	DPair length	Sin. length	Routability (%)	Runtime (s)	DPair length	Sin. length	Routability (%)	Runtime (s)
case1-2	5	5	11x8	2	8.80	8.50	80%	1	8.80	8.20	100%	1
case2-2	9	9	22x7	2	5.78	6.10	88%	1	5.78	5.67	100%	1
case3-2	9	9	18x12	2	10.44	7.50	88%	1	10.44	7.33	100%	1
case4-2	5	5	11x3	2	2.40	2.60	90%	1	2.40	2.20	100%	1
case5-2	6	5	14x3	2	3.67	2.00	100%	1	3.67	2.00	100%	1
case6-2	6	7	17x6	2	7.17	5.67	85%	1	7.67	4.43	100%	1
case7-2	9	9	17x6	2	4.67	4.88	88%	1	4.78	4.55	100%	1
case8-2	10	10	9x16	2	6.40	7.00	90%	1	6.40	6.50	100%	1
case9-2	10	10	8x15	2	5.70	6.63	80%	1	5.80	6.10	100%	1
case10-2	30	36	35x35	2	18.33	16.12	88%	2	18.33	13.44	100%	3
case-b-1	30	36	35x35	2	18.43	17.56	88%	2	18.43	13.72	100%	3
case-b-2	32	60	35x35	2	14.44	16.63	87%	2	14.50	12.30	100%	3
case-b-3	50	66	35x35	2	15.92	12.61	69%	2	15.96	11.24	100%	5
case-b-4	52	88	43x43	2	19.87	15.87	70%	2	19.87	16.01	100%	6
Ratio					1	1.141	1	-	1.006	1	1.175	-

TABLE IV

COMPARISON OF OUR ALGORITHM WITHOUT AND WITH MIXED-CROSSING AVOIDANCE

		Without mixed-crossing avoidance				With mixed-crossing avoidance			
Benchmark	capacity	DPair length	Sin. length	Mixed-crossing number	Runtime (s)	DPair length	Sin. length	Mixed-crossing number	Runtime (s)
case-b-1	3	18.43	13.72	7	2	18.43	14.31	0	27
case-b-2	3	14.44	12.28	7	2	14.44	12.78	0	28
case-b-3	3	15.92	11.21	16	2	15.92	11.24	0	68
case-b-4	3	19.87	15.91	17	3	19.87	16.27	0	146
Ratio	-	1	1	-	-	1	1.028	-	-

TABLE V

COMPARISON OF OUR ALGORITHM WITHOUT AND WITH SLICE-BASED METHOD UNDER CAPACITY OF TWO AND THREE

	Pure ILP solving						Three-slices						Six-slices					
	Capacity 2			Capacity 3			Capacity 2			Capacity 3			Capacity 2			Capacity 3		
Benchmark	Dp.l	Sin.l	Rt(s)	Dp.l	Sin.l	Rt(s)	Dp.l	Sin.l	Rt(s)	Dp.l	Sin.l	Rt(s)	Dp.l	Sin.l	Rt(s)	Dp.l	Sin.l	Rt(s)
case-b-1	18.43	13.72	3	18.43	13.72	28	18.43	13.89	2	18.43	14.33	16	18.57	14.50	1	18.43	14.33	11
case-b-2	14.5	12.30	3	14.44	12.28	28	14.47	12.45	1	14.44	12.78	14	14.47	12.63	2	14.44	12.78	11
case-b-3	15.96	11.24	5	15.92	11.21	68	15.98	11.27	4	15.94	11.24	20	16.00	11.32	1	15.96	11.26	12
case-b-4	19.87	16.01	5	19.87	15.91	146	19.94	16.50	6	19.90	16.38	83	19.98	16.57	5	19.92	16.40	28
Ratio	1	1	1	-	-	-	1.001	1.016	0.81	-	-	-	1.004	1.033	0.56	-	-	-
-	-	-	-	1	1	1	-	-	-	1.001	1.030	0.49	-	-	-	1.001	1.030	0.233

Fig. 20 shows the escape routing result of case-b-4 without mixed-crossing avoidance [Fig. 20(a)] and with the proposed mixed-crossing-avoided algorithm [Fig. 20(b)]. Note that we just provide the global routing result on board, and we do not separate the signals on the same edge. Actually, the relative position relationship can be maintained exactly by the proposed method, even for the edges with both signals of differential pairs and single signals.

D. Effectiveness of Slice-Based MPERA

Table V shows the comparison of our algorithm without and with slice-based speedup method. The results are divided into three patterns: 1) pure ILP solving which solves the ILP without any speedup method; 2) three-slices which partitions the problem into three-subproblems as shown in Fig. 17(c); and 3) six-slices which partitions the problem into six-subproblems as shown in Fig. 17(d).

In Table V, *Dp.l* denotes the average length of all differential pairs, while *Sin.l* denotes the average length of all single signals, *Rt*(*s*) denotes the total running time in seconds. From Table V we can see that the slice-based method can reduce the running time by about 44% for capacity 2 and 76.7% for capacity 3. Compared to the significant improvement on the running time, the increase on length of single signals (3%) is acceptable.

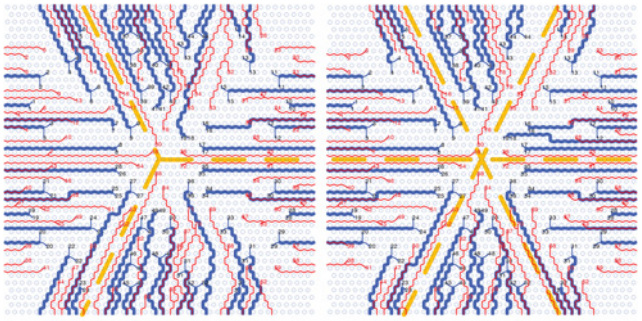


Fig. 21. Escape routing results of case-b-4 with capacity of 3. (a) Three-slice based escape routing. (b) Six-slice based escape routing. The dashed lines are the boundaries of sliced regions and the pins and tile nodes on the boundaries are allocated to the region with larger area.

Fig. 21 shows the escape routing result of case-b-4 based on three subregions [Fig. 21(a)] and six subregions [Fig. 21(b)].

IX. Conclusion

Staggered pin array has become more and more popular for PCB, due to the higher pin density than traditional grid pin array. At the same time, differential pairs are good methods to increase noise immunity of signals for high-speed signal transmission. In this paper, an algorithm for escape routing of simultaneously differential pairs and single signals is proposed on staggered pin array-based PCB while considering the mixed-crossing avoidance. Experimental results show that the proposed method can solve both single-pattern and mixed-pattern effectively while avoiding mixed crossings of both signals.

References

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

[2] R. Shi and C.-K. Cheng, "Efficient escape routing for hexagonal array of high density I/Os," in *Proc. DAC*, 2006, pp. 1003–1008.

[3] Y. Ho, H. Lee, and Y. Chang, "Escape routing for staggered-pin-array PCBs," in *Proc. ICCAD*, 2011, pp. 306–309.

[4] T. Yan and M. D.-F. Wong, "Recent research development in PCB layout," in *Proc. ICCAD*, 2010, pp. 398-403.

[5] T. Yan, P.-C. Wu, Q. Ma, and M. D. F. Wong, "On the escape routing of differential pairs," in *Proc. ICCAD*, 2010, pp. 614–620.

[6] T. Li, W. Chen, X. Cai, and T. Chen, "Escape routing of differential pairs considering length matching," in *Proc. ASP-DAC*, 2012, pp. 139–144.

[7] J.-W. Fang, K.-H. Ho, and Y.-W. Chang, "Routing for chip-package-board co-design considering differential pairs," in *Proc. ICCAD*, 2008, pp. 512–517.

[8] 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 Design*, vol. 26, no. 8, pp. 1417–1429, Aug. 2007.

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

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

[11] Y. Ho, X. Shih, Y. Chang, and C. Cheng, "Layer minimization in escape routing for staggered-pin-array PCBs," in *Proc. ASPDAC*, 2013, pp. 187–192.

[12] P. Wu and M. Wong, "Network flow modeling for escape routing on staggered pin arrays," in *Proc. ASPDAC*, 2013, pp. 193–198.

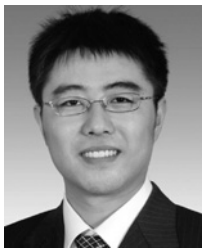
[13] K. Wang, H. Wang, and S. Dong, "Escape routing of mixed-pattern signals based on staggered-pin-array PCBs," in *Proc. ISPD*, 2013, pp. 93–100.

[14] M. R. Garey and D. S. Johnson, *A Guide to the Theory of NPCompleteness*. San Francisco, CA, USA: Freeman, 1979.

[15] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable multicommodity flow problems," in *Proc. SFCS*, 1975, pp. 184–193.

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

[17] (2013, Feb. 2). *GurobiOptimizer* [Online]. Available: http://www.gurobi.com/download/gurobi-optimizer

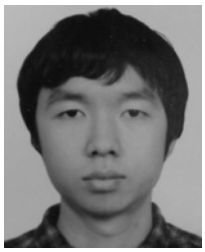


Kan Wang (S'13) Received the B.S. degree in soft engineering from the Dalian University of Technology, Dalian, China, in 2009, and is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China.

His current research interests include algorithms for VLSI Physical Design, especially floorplanning, thermal optimization, low power design for 2-D and 3-D NoC, and escape routing.

Sheqin Dong (M'13) received the B.S. degree (honors) in computer science, the M.S. degree in semiconductor physics and device, and the Ph.D. degree in mechatronic control and automation from the Harbin Institute of Technology, Harbin, China, in 1985, 1988, and 1996, respectively.

From 1997 to 1999, he was a Post-Doctoral Fellow with the State Key Laboratory of CAD and CG in Zhejiang University, Zhejiang, China. He is currently an Associate Professor and the director of the EDA Laboratory with the Department of Computer Science and Technology of Tsinghua University, Beijing, China. His current research interests include VLSI physical design, SoC design and packaging, optimization algorithm and its application, and application specific architecture design and chip design.

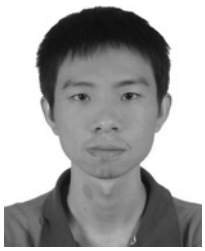


Huaxi Wang received the B.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012.

He is currently a Software Engineer in Macau, China, focusing on online gaming.

Qian Chen received the B.S. degree in communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2010, and he is currently a master's student with the Department of Computer Science and Technology, Tsinghua University, Beijing, China.

His current research interests include algorithms for VLSI physical design, especially floorplanning and placement.



Tao Lin received the bachelor's degree from the Department of Computer Science, East China University of Science and Technique, Beijing, China, and the master's degree from computer science department of Tsinghua University, Beijing, in 2008 and 2011, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA.

His current research interests include combinatorial optimization in very large scale integrated computer-aided design.