An MILP-Based Wire Spreading Algorithm for PSM-Aware Layout Modification*

Ming-Chao Tsai, Yung-Chia Lin, Ting-Chi Wang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
{d938316@oz.nthu.edu.tw, g944338@alumni.oz.nthu.edu.tw, tcwang@cs.nthu.edu.tw}

Abstract

Phase shifting mask (PSM) is a promising resolution enhancement technique, which is used in the deep sub-wavelength lithography of the VLSI fabrication process. However, applying the PSM technique requires the layout to be free of phase conflicts. In this paper, we present a mixed integer linear programming (MILP) based layout modification algorithm which solves the phase conflict problem by wire spreading. Unlike existing layout modification methods which first solve the phase conflict problem by removing edges from the layout-associated conflict graphs and then try to revise the layout to match the resultant conflict graphs, our algorithm simultaneously considers the phase conflict problem and the feasibility of modifying the layout. The experimental results indicate that without increasing the chip size, the phase conflict problem can be well tackled with minimal perturbation to the layout.

I. Introduction

Optical lithography has been a critical step in the VLSI fabrication process. As the pitches of leading-edge products scale down, phase shifting mask (PSM) and immersion lithography are viewed as potential solutions to carry the 193nm lithography beyond 65nm node. The key of immersion lithography lies on high-index fluids which increase the numerical apertures of the lithography system. Nevertheless, ASML and Nikon recently announced that the numerical apertures had essentially reached their limit for water-based immersion lithography [1]. In the 45nm technology node, the pitch of metal 1 (M1) is reduced to 90nm, which necessitates the incorporation of immersion lithography and strong PSM techniques such as alternating PSM (altPSM).

Assume that the minimum spacing which can be resolved by applying conventional mask is B. Beyond this spacing, strong constructive diffraction effect interferes the imaging of critical features. Applying altPSM extends the limit of resolution to b ($=B/2$) by destructive interference. However, this resolution improvement can be achieved only if the apertures of two adjacent critical features are assigned to opposite phases. Although altPSM shows great potential in resolution enhancement, a layout must be compliant to the phase assignment constraint. *Fig. 1* shows a layout which cannot satisfy the phase assignment constraint. Since spacing between any two of the three rectangular wire segments in *Fig. 1* is critical, any two of these features must be in opposite phases. However, no matter what phase we assign to the bottom feature, it will be the same as one of the upper two features. Thus, this layout has the phase conflict problem.

Masks can be categorized into dark field masks and bright field masks. The dark field masks are mainly used for metal layers, and the bright field masks are usually used for poly layers. Targeting on the dark field altPSM, McCullen reported routing restrictions that enable the generation of phase-correct layout [2]. Berman *et al.* [3] proposed a graph based algorithm for solving the phase conflict problem. To obtain altPSM compliant layouts, conflict graphs are constructed according to given layouts, and then a minimum-weight set of edges is removed from each graph to ensure the resultant graph 2-colorable. The edge deletions in the conflict graphs are accomplished by changing the placement of the features. Although their algorithm is efficient, it is hard to assign edge weights since we cannot compute how far each layout object needs to be moved before replacement is done. Moreover, this approach may induce area overhead.

To handle the phase conflict problem for bright field masks, Cao *et al.* proposed a Boolean satisfiability based method to generate PSM compliant and composiable cell libraries [4]. Although their algorithm considered the phase conflict problem within and between library cells, it does not consider phase conflicts among interconnections. In addition, the areas of resultant library cells increase. Chiang *et al.* proposed a layout correction algorithm for standard-cell layouts [5]. The proposed algorithm targets on bright field AAPSM (alternating aperture PSM) for the poly layer. It inserts end-to-end spaces into layouts to solve the phase conflict problem. Although it completely eliminates phase conflicts in a given layout, unfortunately it also induces area increasing.

Since the pitch of critical metal layers continues to scale down, it becomes urgent to find a solution to solve the phase conflict problem for critical metal layers. In addition, the advanced VLSI technology adopts the dual damascene (DD) process flow to accomplish the copper metallization. Therefore, in this paper, we focus on the dark field altPSM which pertains to the DD process. Although [3] has presented an efficient algorithm to remove phase conflicts, unfortunately the algorithm not only induces area overhead but also changes the placement, which is costly from the viewpoint of a physical design flow. To cope with these drawbacks, we propose a new layout modification algorithm to solve the phase conflict problem. Our algorithm uses the wire spreading technique to adjust the wire segment positions on the critical metal layers within the predefined die size. In addition, to effectively reduce the perturbation to a layout, our algorithm tries to reduce phase conflicts as many as possible and to minimize the total amount of

* This work was partially supported by Ministry of Economic Affairs under Grant No. MOEA-95-EC-17-A-01-S1-031 and National Science Council under Grant No. NSC-96-2220-E-007-025.

wire segment movement. Our algorithm is designed based on mixed integer linear programming (MILP), and is applicable to both standard-cell and custom layouts. The MILP based algorithm is flexible enough to take other practical issues, such as the lengths of critical nets, into consideration as well. Different from existing works which first solve the phase conflict problem by removing edges from the layout-associated conflict graphs to make the resultant graphs 2-colorable and then try to revise the layout to match the resultant conflict graphs, our algorithm *directly* fixes the phase conflict problem through wire spreading while at the same time minimizing the total amount of wire segment movement. Compared with the number of edges deleted from conflict graphs, the total amount of wire segment movement is a more direct metric to reflect the cost for revising a layout. Furthermore, the recently remarkable improvement on linear programming solvers [6] has made our algorithm become competitive, and the experimental results well support the effectiveness of our algorithm.

The rest of this paper is organized as follows. In Section II, we describe the preliminaries and our strategy to solve the phase conflict problem using wire spreading. Section III describes our algorithm. Section IV provides the experimental results which demonstrate the effectiveness of our approach. Finally, this paper ends with the conclusions in Section V.

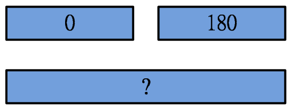


Fig. 1 A layout which has the phase conflict problem.

II. Preliminaries and Problem Formulation

A. *Preliminaries*

The phase assignment of wire segments on a metal layer can be done by coloring the corresponding conflict graph with two colors. In the graph each node represents a wire segment. If the spacing between a pair of adjacent wire segments falls in [b, B], the two wire segments must be assigned to opposite phases and we create an edge between their corresponding nodes. The layout is said to be *altPSM compliant* if and only if all associated conflict graphs are 2-colorable. If two nodes connected by an edge have the same color, we say they incur a *phase conflict*.

For simplicity and clarity, we assume all wire segments are rectangles[1]. The construction of a conflict graph is described as follows. First, we bloat each wire segment by the distance b. Next, we apply the line sweeping algorithm to check if two bloated regions intersect or not. Then, the edges are constructed between pairs of nodes whose corresponding bloated segments intersect. However, according to our empirical study, we observed that if a wire segment is placed at the lower-left or lower-right position of another wire segment and their bloated regions intersect, the phases of these two wire segments can be identical without causing unresolvable aerial image. To demonstrate this observation, we use a layout with four wire segments A, B, C and D as shown in *Fig. 2*(*a*). The dashed rectangles show their bloated regions, and the shaded area highlights the bloated region of C. The four red solid lines and two red dashed lines are the edges of the conflict graph. *Fig. 2*(*b*) shows the simulated aerial image of the four wire segments as well as their phases. The simulation is performed by Silvaco Athena [7], where the wavelength and NA are 193nm and 0.85, respectively, and the wire width and spacing are both 90nm. From the result, we can see that although the bloated regions of B (the upper-left wire segment) and C (the lower-right wire segment) overlap, no image bridging occurs between them and therefore they both can be assigned to the same phase. As a result, we can remove the edge between their corresponding nodes from the conflict graph (i.e., the red dashed edge connecting B and C in *Fig. 2* (a) can be removed). The same observation applies to A (the upper-right wire segment) and D (the lower-left wire segment) as well, and therefore the edge (A, D) in *Fig. 2* (a) can be removed. By removing these "crossing" edges, we can guarantee the planarity of the conflict graph. Throughout the rest of this paper, we will assume conflict graphs are all planar.

[1] In general, wires can be rectilinear. However, we can always slice any complex rectilinear polygon into a set of rectangular segments and enforce the phase consistency among these resultant segments.

It is not hard to see that a metal layer is altPSM compliant if and only if its conflict graph does not have any odd face[2]. On the other hand, the minimum number of phase conflicts among all possible phase assignment solutions of a metal layer is bounded by half the number of the odd faces in the corresponding conflict graph, which can be derived from [8]. Therefore, we use the number of odd faces as a metric to measure the amount of phase conflicts.

In this paper, we use wire spreading to remove phase conflicts among wire segments. A proper wire spreading solution induces edge deletion to remove odd faces of a conflict graph. Thus, our goal is to use the wire spreading technique to reduce the number of odd faces as many as possible. For an easier presentation, we only focus on wire spreading along the vertical direction throughout the rest of this paper, unless stated otherwise. In order to maintain the correct circuit functionality and avoid over distortion for a given layout, the following constraints must be satisfied after wire spreading:

1. Die region constraint:
   Since the dimensions of a die have been specified in the very beginning of a physical design flow, it is costly to fix phase conflicts by relaxing the die size. Thus, all the wire segments must locate within the original die region.
2. Fixed pin constraint:
   Since we do not alter the placement results, the I/O pins of cells or hard macros are thought to be immovable. If one end of a wire segment connects with an immovable pin, then the coordinate of this end must be left unchanged.
3. Connection constraint:
   If wire segments in different layers are connected through vias, then these connections must be maintained.
4. Vertical order constraint:
   To ensure that routing patterns will not be drastically changed, if the horizontal spans of two wire segments in the same layer overlap, then their relative positions in the vertical direction must be held.
5. Minimum spacing constraint:
   The spacing between any two wire segments cannot be smaller than minimum spacing b.
6. Maximum movement constraint
   Each wire segment can be moved only within a user defined range.

Besides, it is favorable to size down the conflict graphs and not to over modify the original layout. Therefore, the following constraint should also be satisfied after wire spreading:

[2] For any face in a planar graph, it is said to be an odd face if the number of edges along the face is odd; otherwise it is said to be an even face.

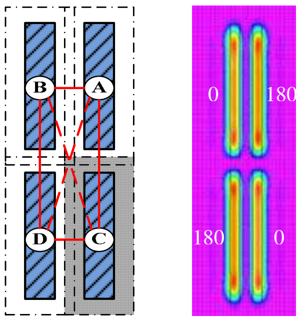


Fig. 2 (a) The conflict graph of four wire segments. (b) The simulated aerial image.

7. Graph simplicity constraint
   If the distance between two wire segments is longer than *B*, then this spatial relation should be kept in the revised layout. Otherwise, it will introduce a new edge in the conflict graph. This constraint ensures not to add edges into conflict graphs.

A wire spreading solution is said to be *feasible* if it meets the above seven constraints.

B. *Problem Formulation*

The problem of applying wire spreading to remove phase conflicts is stated as follows:

Given a layout which contains *n* metal layers, we first construct a conflict graph for each of the *n* layers. The problem asks for finding a new arrangement for the wire segments lying in each of the *n* layers such that the sum of the numbers of odd faces of the resultant conflict graphs is minimized and the total amount of wire segment movement is also minimized. In addition, the wire spreading solution must satisfy all the constraints stated in Section II.A.

To reduce the number of odd faces in a conflict graph, we need to delete edges from the graph. Each edge in a conflict graph can be exactly categorized into one of the following four types:

(1) Shared by two odd faces: If the edge is deleted, it will merge two odd faces into one even face and therefore the number of odd faces can be reduced by two.
(2) Shared by two even faces: If the edge is deleted, it will merge two even faces into one even face and therefore the number of odd faces will be unchanged.
(3) Shared by one odd face and one even face: If the edge is deleted, it will merge the two faces into one odd face and therefore the number of odd faces will be unchanged.
(4) Not shared by two faces: If the edge is deleted, it does not merge any face and therefore the number of odd faces will be unchanged.

From the above observations, we have the following lemma.

Lemma 1: For each edge deletion, the number of odd faces can be reduced if and only if the deleted edge is shared by two odd faces.

Although merging an odd face with an even face by deleting one of their common edges does not immediately reduce the number of odd faces, it makes the odd face grow. After several iterations, this growing odd face may directly abut with another odd face and then we can delete one of their common edges to merge them into an even face. The series of mergings can be regarded as pairing two odd faces through a series of even faces. *Fig. 3* shows an example of how to pair two odd faces which are not directly abutted. Initially, two odd faces F_1 and F_3 are not abutted as seen in (a). By sequentially removing edges 7 and 8 from the graph, F_1, F_2 and F_3 gradually merge into an even face F_5 which is shown in (*b*). In this paper, we aim to reduce the number of odd faces by pairing odd faces (which may or may not be directly abutted at the beginning). In the following, we define a special edge set called *merging set*.

Definition 1 (Merging set): Given a conflict graph, a merging set is a set of edges whose deletion from the graph can merge two odd faces into one even face. For a pair of odd faces, there may exist more than one merging set. A merging set is said to be *minimal* if there exists no edge *e* in the set such that after deleting the edge *e*, the resultant set is still a merging set.

According to the above definition and Lemma 1, deleting all edges in a minimal merging set from a conflict graph reduces the number of odd faces by 2. It is also worth noting that a set of odd face pairings may not always reduce the number of odd faces by twice the number of pairings, or may not be always achievable by wire spreading. We use *Fig. 4*(*a*) and *Fig. 4*(*b*) to explain these two cases, respectively. In *Fig. 4*(*a*), we have the merging sets {(*u*, *v*)} and {(*v*, *w*)} for odd face pairs (*A*, *B*) and (*B*, *C*), respectively. Deleting edge (*u*, *v*) or edge (*v*, *w*) alone merges faces *A* and *B*, or *B* and *C* into one even face. Intuitively, deleting two merging sets simultaneously implies to eliminate two pairs of odd faces, and the number of odd faces should be reduced by four. However, because the two odd face pairs share a common face *B*, simultaneously deleting (*u*, *v*) and (*v*, *w*) merely merges the three odd faces into one odd face and thus the number of odd faces is reduced only by two. *Fig. 4*(*b*) shows an example where simultaneously performing two odd face pairings may be infeasible. According to the conflict graph shown in *Fig. 4*(*b*), both odd face pairs in the upper and the lower parts can be merged individually by deleting edges (*p*, *q*) and (*q*, *r*), respectively. However, the wire segments *p* and *r* are connected with immovable pins indicated by black squares, and therefore *p* and *r* cannot be moved through wire spreading. Now, if the vertical moving range for *q* (i.e., the range between the bottom boundary of *p* and the top boundary of *r*) is not large enough to move *q* to a location such that both edges (*p*, *q*) and (*q*, *r*) can be removed from the conflict graph, then simultaneously performing two odd face pairings in this graph becomes impossible.

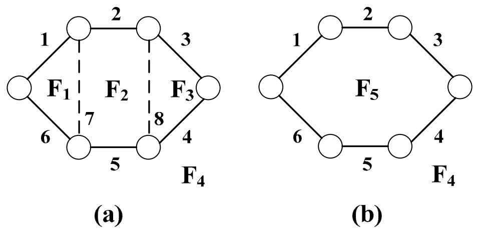


Fig. 3 Iteratively removing the edges which are shared by odd faces and even faces may eventually eliminate two odd faces which are not abutted.

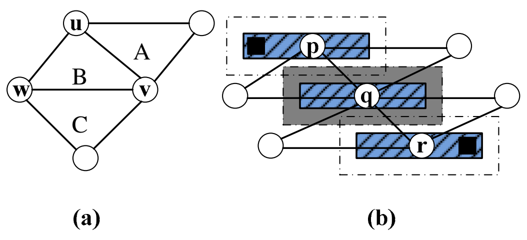


Fig. 4. (a) A case which overestimates the number of eliminated odd faces. (b) A layout infeasible case.

We next define another special edge set called *orthogonal set*.

Definition 2 (Orthogonal set): Given the conflict graphs corresponding to a layout, an orthogonal set is a union of a collection of minimal merging sets and satisfies the following two conditions: (1) In this set, edges belonging to different minimal merging sets do not share any common face. (2) There exists at least a feasible wire spreading solution such that all the edges in this set are no longer in the new conflict graphs. The cardinality of an orthogonal set is defined to be the number of merging sets involved in the set, which is also equal to half the amount of eliminated odd faces after removing from the conflict graphs all the edges in the set.

Definition 3 (Deviation): The deviation induced by an orthogonal set is the minimum total amount of wire segment movement among all feasible wire spreading solutions corresponding to this orthogonal set.

In this paper, instead of solving the general wire spreading problem as described at the beginning of this subsection, we only focus on *the multi-layer odd face pairing problem* which asks to find an orthogonal set with maximum cardinality and a corresponding wire spreading solution with the minimum deviation.

III. Algorithm

In this section, we first introduce another special edge set called *candidate removal set* and show the relation between a candidate removal set and an orthogonal set. We then derive an MILP model to find a candidate removal set (and thus an orthogonal set as well) and a corresponding wire spreading solution such that the total cardinality and the total deviation are both optimized.

A. *Candidate Removal Set*

Candidate removal set is defined as follows:

Definition 4 (Candidate removal set): Assume that we are given n conflict graphs $G_i=(V_i, E_i)$'s, $1\leqq i\leqq n$, corresponding to a n-layer layout. Let $G=(V, E)$, where $V=V_1\cup V_2\ldots\cup V_n$, and $E=E_1\cup E_2\ldots\cup E_n$. A candidate removal set E' is a subset of E such that there exists at least a feasible wire spreading solution to remove all the edges in E' from G. In addition, E' must satisfy the following constraints:

(1) For every odd face OF in G, either there is exactly one edge in E' which is also in OF, or no edge in E' is also in OF.
(2) For every even face EF in G, either there are exactly two edges in E' which are also in EF, or no edge in E' is also in EF.

In the remaining subsection, we step-by-step explain the relation between a candidate removal set and an orthogonal set. We first have the following lemma which can be proved based on Lemma 1, Definition 1 and the concept of graph duality.

Lemma 2: Given a conflict graph G_i and its dual graph D_i, a set M of edges is a minimal merging set in G_i if and only if the set M^* of the dual edges of M forms a path in D_i such that each of the two end nodes of the path corresponds to an odd face in G_i and each of the other nodes in the path corresponds to an even face in G_i.

We use *Fig. 5* to facilitate the understanding of the above lemma. *Fig. 5* (*a*) depicts a conflict graph which includes 8 nodes, 11 edges and 5 faces. The faces are denoted by F_1 to F_5, where F_1 and F_2 are the only two odd faces. A minimal merging set M to merge F_1 and F_2 into an even face consists of the edges *4*, *6* and *11*. *Fig. 5*(*b*) illustrates the dual graph. The nodes N_1 to N_5 are the corresponding nodes of faces F_1 to F_5, respectively. Same tag number is used for each edge and its dual edge, except a superscript of star mark is used to distinguish a dual-graph edge and a conflict-graph edge. The edges 4^*, 6^* and 11^* which are dual edges of M form a path N_1-N_3-N_4-N_2 in the dual graph. The two end nodes N_1 and N_2 correspond to the odd faces F_1 and F_2 while the other nodes N_3 and N_4 correspond to the even faces F_3 and F_4.

Lemma 3: If two minimal merging sets of a conflict graph are involved in the same orthogonal set, their corresponding paths in the dual graph are node-disjoint.

Proof: By Lemma 2, the dual edges of each minimal merging set form a path. By Definition 2, the two minimal merging sets do not share any common face. Thus, the corresponding paths of the two merging sets are node-disjoint in the dual graph. □

Theorem 1 (Orthogonal set inclusion theorem): Any candidate removal set contains an orthogonal set.

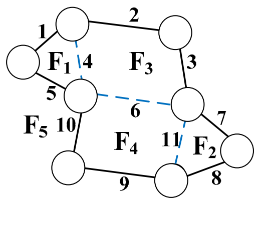
Proof: According to the concept of graph duality and the constraints (1) and (2) imposed on a candidate removal set, the dual edges of a candidate removal set E' can only compose two kinds of connected components - paths and cycles. We can further uniquely decompose E' into two disjoint edge sets P and C, and use P^* and C^* to denote their dual edge sets, respectively, where P^* is composed of a set of node-disjoint paths and C^* is composed of a set of cycles. Moreover, the two end nodes of each path in P^* correspond to odd faces in G and the other nodes in the path correspond to even faces in G. Therefore, according to Lemma 2 and Lemma 3, P is a collection of minimal merging sets which do not share common faces in G. Besides, if S is a feasible wire spreading solution for E', then S is also a feasible wire spreading solution for P, according to Definitions 2 and 4. As a result, P is an orthogonal set. □

It is not hard to see that in the proof of Theorem 1, the number of paths in P^* is half the number of odd-degree nodes in P^*. Thus, the cardinality of P is half the number of odd-degree nodes in P^*. For convenience, we define the *equivalent cardinality* of the candidate removal set E' to be the number of paths in P^*. Based on Theorem 1, the multi-layer odd face pairing problem will now be tackled by solving the following problem.

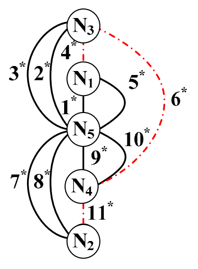
Candidate removal set problem: Given an n-layer layout, this problem asks for finding a candidate removal set and a corresponding feasible wire spreading solution such that the equivalent cardinality of the candidate removal set is maximized and the induced deviation is minimized.

B. *An MILP-Based Wire Spreading Algorithm*

Table 1 lists the notations and the definitions of the variables to be used in the description of our MILP based algorithm. First, we construct a conflict graph G_i for the ith layer of the layout. Each node $N_{i,j}$ is represented by a five tulpe ($x_{0,i,j}$, $y_{0,i,j}$, $x_{1,i,j}$, $y_{1,i,j}$, $w_{i,j}$) which records the geometric information of the wire segment of $N_{i,j}$. Then, the candidate removal set problem is formulated as an MILP model below.



(a)



(b)

Fig. 5: An illustration of a conflict graph and its dual graph.

Maximize $\beta\sum_{i=1}^{n}\sum_{l=1}^{o_i}(f_{i,l} \mid F_{i,l} \text{ is an odd face}) - \sum_{i=1}^{n}\sum_{j=1}^{m_i}(\delta_{0,i,j}+\delta_{1,i,j}+\varepsilon_{0,i,j}+\varepsilon_{1,i,j})$

subject to:

$$y_{0,i,j} - w_{i,j} \geq 0, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (1)$$

$$y_{0,i,j} + w_{i,j} \leq H, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (2)$$

$$y_{0,i,j} = y^*_{0,i,j}, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i, \quad (3)$$
$$T_{0,i,j} \text{ connects with a fixed pin}$$

$$y_{1,i,j} = y^*_{1,i,j}, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i, \quad (4)$$
$$T_{1,i,j} \text{ connects with a fixed pin}$$

$$y^*_{0,i,j} - movement \leq y_{0,i,j} \leq y^*_{0,i,j} + movement, \quad (5)$$
$$\forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i$$

$$y^*_{1,i,j} - movement \leq y_{1,i,j} \leq y^*_{1,i,j} + movement, \quad (6)$$
$$\forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i$$

$$y_{0,i,j} = y_{0,i+1,k}, \forall i,j,k : 1 \leq i \leq n-1, 1 \leq j \leq m_i, \quad (7)$$
$$1 \leq k \leq m_{i+1}, T_{0,i,j} \text{ connects } T_{0,i+1,k}$$

$$y_{0,i,j} - w_{i,j} - w_{i,k} - y_{1,i,k} \geq b, \forall i,j,k : 1 \leq i \leq n, 1 \leq j \leq m_i, \quad (8)$$
$$j < k \leq m_i, y_{0,i,j} \geq y_{1,i,k}, \text{the horizontal spans of the}$$
$$\text{bloated regions of } N_{i,j} \text{ and } N_{i,k} \text{ overlap}$$

$$y_{0,i,j} = y_{1,i,j}, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i, \quad (9)$$
$$N_{i,j} \text{ is a horizotal wire segment.}$$

$$y_{0,i,j} - w_{i,j} - w_{i,k} - y_{1,i,k} \geq B, \forall i,j,k : 1 \leq i \leq n, 1 \leq j \leq m_i, \quad (10)$$
$$j < k \leq m_i, E_{i,j,k} \notin G_i, y^*_{0,i,j} \geq y^*_{1,i,k}$$

$$y_{0,i,j} - w_{i,j} - w_{i,k} - y_{1,i,k} \geq B \times e_{i,j,k}, \forall i,j,k : 1 \leq i \leq n, \quad (11)$$
$$1 \leq j \leq m_i,\ j < k \leq m_i, E_{i,j,k} \in G_i, y^*_{0,i,j} \geq y^*_{1,i,k}$$

$$f_{i,l} = \sum_{j=1}^{m_i}\sum_{k>j}^{m_i}(e_{i,j,k} \mid E_{i,j,k} \in F_{i,l} \text{ and } F_{i,l} \text{ is an odd face}) \quad (12)$$
$$\forall i,l : 1 \leq i \leq n, 1 \leq l \leq o_i$$

$$2 \times f_{i,l} = \sum_{j=1}^{m_i}\sum_{k>j}^{m_i}(e_{i,j,k} \mid E_{i,j,k} \in F_{i,l} \text{ and } F_{i,l} \text{ is an even face}) \quad (13)$$
$$\forall i,l : 1 \leq i \leq n, 1 \leq l \leq o_i$$

$$\delta_{0,i,j} \geq 0, \text{and } y^*_{0,i,j} - y_{0,i,j} + \delta_{0,i,j} \geq 0,\ \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (14)$$

$$\varepsilon_{0,i,j} \geq 0, \text{and } y_{0,i,j} - y^*_{0,i,j} + \varepsilon_{0,i,j} \geq 0,\ \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (15)$$

$$\delta_{1,i,j} \geq 0, \text{and } y^*_{1,i,j} - y_{1,i,j} + \delta_{1,i,j} \geq 0,\ \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (16)$$

$$\varepsilon_{1,i,j} \geq 0, \text{and } y_{1,i,j} - y^*_{1,i,j} + \varepsilon_{1,i,j} \geq 0, \forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i \quad (17)$$

$$x_{0,i,j}, y_{0,i,j}, x_{1,i,j}, y_{1,i,j}, \delta_{0,i,j}, \delta_{1,i,j}, \varepsilon_{0,i,j}, \varepsilon_{0,i,j} \in R, \quad (18)$$
$$\forall i,j : 1 \leq i \leq n, 1 \leq j \leq m_i$$

$$e_{i,j,k} \in \{0,1\}, \forall i,j,k : 1 \leq i \leq n, 1 \leq j \leq m_i,\ j < k \leq m_i, \quad (19)$$
$$E_{i,j,k} \in G_i$$

$$f_{i,l} \in \{0,1\},\ \forall i,l : 1 \leq i \leq n, 1 \leq l \leq o_i \quad (20)$$

The objective function can be decomposed into two terms. The first term is the gain function which is equal to the reduction of number of odd faces; i.e. twice the cardinality of a candidate removal set. The second term is the penalty function which is equal to the deviation induced by revising the layout. A constant β is introduced to provide a tradeoff between these two terms. Constraints (1) and (2) require wire segments to locate within a fixed die area. Constraints (3) and (4) bind the ends of wire segments with fixed pins. Constraints (5) and (6) confine wire segments to move within the given allowable range. Constraint (7) maintains the connectivity among wire segments. To satisfy the minimum spacing rule of altPSM and to keep the relative vertical order of wire segments, we have constraint (8). To maintain the orientations of horizontal wire segments, we have constraint (9). To preserve the graph simplicity constraint, we have constraint (10). In constraint (11), we use a Boolean variable $e_{i,j,k}$ to determine whether edge $E_{i,j,k}$ is removed or not. If $e_{i,j,k}$ is assigned to 1, $E_{i,j,k}$ is removed from the corresponding conflict graph. As a consequence, the spacing between wire segments $N_{i,j}$ and $N_{i,k}$ must be greater than B. To satisfy the constraints imposed on a candidate removal set, we have constraints (12) and (13). To calculate the deviation of both ends of a wire segment, we apply a relaxation technique. With constraints (14) and (15), the sum of $\delta_{0,i,j}$ and $\varepsilon_{0,i,j}$ is exactly the absolute value of difference between $y_{0,i,j}$ and $y^*_{0,i,j}$, when we minimize $\delta_{0,i,j}$ and $\varepsilon_{0,i,j}$. Similar principles are applied to calculate the deviation of $T_{1,i,j}$. Thus we have constraints (16) and (17). Finally, the value ranges of all variables are given by constraints (18), (19) and (20).

In this MILP model, we introduce six real variables for each node, one binary variable for each edge and one binary variable for each face. Assume that the number of nodes, edges and faces are v, e and f, respectively. The total number of variables is $6v+e+f$. In the given planar graphs, both e and f are of the same order as v. Thus, the number of variables is $O(v)$. Obviously we can see that, except constraint (8) whose number grows with v^2, the number of the rest of the constraints linearly grows with v, e or f. Therefore, the number of constraints is $O(v^2)$.

Table 1: Variables and notations used in our formulation.

H	The height of the die region
B	The minimum spacing without applying altPSM
b	The achievable minimum spacing of altPSM
movement	Maximum allowable vertical movement of wire segments
n	The number of layers
G_i	The conflict graph of the ith layer
m_i	The number of nodes in G_i
o_i	The number of faces in G_i
$N_{i,j}$	The jth node in the conflict graph G_i
$E_{i,j,k}$	The edge between $N_{i,j}$ and $N_{i,k}$
$T_{0,i,j}$	If $N_{i,j}$ is a horizontal wire segment, $T_{0,i,j}$ is the left end of the center line of $N_{i,j}$. Otherwise, $T_{0,i,j}$ is the lower end of the center line of $N_{i,j}$.
$T_{1,i,j}$	If $N_{i,j}$ is a horizontal wire segment, $T_{0,i,j}$ is the right end of the center line of $N_{i,j}$. Otherwise, $T_{0,i,j}$ is the upper end of the center line of $N_{i,j}$.
$e_{i,j,k}$	A Boolean variable to control the deletion of edge $E_{i,j,k}$. If $e_{i,j,k}$ is 1, edge $E_{i,j,k}$ is removed from G_i.
$f_{i,l}$	A Boolean variable to determine the number of deleted edge(s) of face $F_{i,l}$. If $F_{i,l}$ is an odd face, we remove one edge from $F_{i,l}$ when $f_{i,l}$ is 1. If $F_{i,l}$ is an even face, we remove two edges from $F_{i,l}$ when $f_{i,l}$ is 1. If $f_{i,l}$ is 0, no edge is removed from from $F_{i,l}$.
$(x_{0,i,j}, y_{0,i,j})$	The coordinate of $T_{0,i,j}$ after wire spreading
$(x_{1,i,j}, y_{1,i,j})$	The coordinate of $T_{1,i,j}$ after wire spreading
$(x^*_{0,i,j}, y^*_{0,i,j})$	The original coordinate of $T_{0,i,j}$
$(x^*_{1,i,j}, y^*_{1,i,j})$	The original coordinate of $T_{1,i,j}$
$w_{i,j}$	The half width of wire segment $N_{i,j}$
$\delta_{0,i,j}$, $\delta_{1,i,j}$, $\varepsilon_{0,i,j}$, $\varepsilon_{1,i,j}$	Four relaxation real variables to calculate the deviation of both ends of wire segment $N_{i,j}$

IV. Experimental Results

Our algorithm was implemented in C++ and run on an Intel 1.6GHz Linux machine. We used CPLEX [9] to solve the MILP problems. The values of B and b are 160nm and 80nm, respectively. We adopted randomly generated layouts with the same minimum spacing and wire width to test the robustness and stability of our algorithm. All the layouts use 4 metal layers. *Table 2* shows the statistics of each layout. The names of layouts are listed in the first column. The *#node*, *#edge* and *#odd_face* respectively show the numbers of nodes, edges and odd faces of the conflict graphs before wire spreading. The number of nets for each layout is indicated in the column *#net*. For general consideration, we assume that some pins are immovable.

The experimental results are summarized in *Table 3*. Besides our algorithm, we also present the results of two other algorithms (the adjacent pairing algorithm and the aggressive algorithm) for comparison. The adjacent pairing algorithm removes the odd faces by deleting edges which are shared by two abutted odd faces, and the aggressive algorithm eliminates odd faces as many as possible without considering the deviation. Both of these two algorithms are derived from our algorithm by either modifying some constraints or the objective function. For each layout, each algorithm performed one run of vertical wire spreading followed by horizontal wire spreading. The columns *#rof*, *deviation*, *%comp* and *runtime* in *Table 3* show the number of remaining odd faces, the induced deviation (nm), the reduction rate of odd faces, and the run time (sec), respectively. To make comparisons among these algorithms, we normalize the value of each column with respect to the results of our algorithm. The adjacent pairing algorithm intuitively solves the phase conflict problem using the observation of Lemma 1. Although it is the fastest algorithm among the three algorithms, the solution space is only a subset of the other two algorithms. Many odd faces which are surrounded by even faces have no chance to merge with other odd faces. Therefore, there are still many odd faces left after wire spreading. On the contrary, the aggressive algorithm completely eliminates all odd faces and the runtime is shorter than our algorithm. However, it induces a large amount of deviation, which may impact the circuit timing. In order to emphasize the importance of deviation, we assigned β a relatively small value in our algorithm. The results indicate that our algorithm shows a good tradeoff among runtime, deviation, and completion rate of odd face elimination. Our completion rate is only slightly lower than the aggressive algorithm, and the deviation is significantly smaller than the aggressive algorithm. In addition, the experimental results demonstrate that the runtime of our MILP algorithm is reasonable.

Table 2: Benchmark layout parameters.

	#node	#edge	#odd_face	#net
C1	1009	436	14	254
C2	4984	2088	58	1702
C3	10974	6668	330	3514
C4	19642	12578	668	6175
C5	57182	36373	1908	17787

V. Conclusion

In this paper, we present an MILP-based wire spreading algorithm to solve the phase conflict problem with minimal perturbation to the given layout. Different from the previous works which ask for finding a minimum weighted edge set or a minimal edge set whose removal makes the conflict graph 2-colorable, our algorithm use the exact deviation of wire segments as cost. The experimental results show the effectiveness of our algorithm; less than 2% of odd faces are left in the modified layout and no area increase is needed for the modification. In addition, the modification will not alter the lengths of the critical nets, if we add linear constraints to control the critical wire lengths. The flexible MILP model also allows integrating with a variety of practical issues such as redundant via insertion [10].

Table 3: The experimental results of layout modification.

	Our algorithm				Adjacent pairing algorithm				Aggressive algorithm			
	#rof	deviation	%comp	runtime	#rof	deviation	%comp	runtime	#rof	deviation	%comp	runtime
C1	0	629	100%	0.2	0	629	100%	0.2	0	677424	100%	0.2
C2	0	3667	100%	1.8	0	3503	100%	2.9	0	3504300	100%	1.3
C3	2	18691	99.39%	9.1	48	13599	85.45%	8.2	0	13756230	100%	8.6
C4	8	33967	98.88%	24.0	102	23339	84.73%	9.9	0	23933200	100%	22.1
C5	30	96535	98.43%	297.7	324	70711	83.02%	131.4	0	69980000	100%	140.8
	normalized				normalized				normalized			
	#rof	deviation	%comp	runtime	#rof	deviation	%comp	runtime	#rof	deviation	%comp	runtime
	1	1	1	1	11.85	0.73	0.85	0.46	0	728.7	1.01	0.52

Reference

[1] http://www.reed-electronics.com/semiconductor/article/CA6356254

[2] Kevin McCullen, "Phase Correct Routing for Alternating Phase Shift Masks", in *Proc. DAC*, pp 317-320, 2004.

[3] P. Berman, A. B. Kahng, S. Mantik, I. L. Markov, and A. Zelikovsky, "Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks", in *IEEE TCAD* (9), pp 1265-1278, 1999.

[4] K. Cao, J. Hu, and M. Cheng, "Layout Modification for Library Cell Alt-PSM Composablility" in *SPIE*, 2004.

[5] C. Chiang, A. B. Kahng S. Sinha and X. Xu, "Fast and Efficient Phase Conflict Detection and Correction in Standard-Cell Layouts", in *Proc. ICCAD*, pp. 149-155, 2005.

[6] M. Chi and D.Z. Pan, "BoxRouter: A new Global Router Based on Box Expansion and Progressive ILP", in *Proc. DAC*, pp. 373-378, 2006.

[7] http://www.silvaco.com

[8] D. Kral and H.-J. Vosss, "Edge-Disjoint Odd Cycle in Planar Graphs", in *Journal of Combinatorial Theory Series B*, Vol 90, Issue 1, pp. 107-120, 2007.

[9] http://www.ilog.com

[10] K. McCullen, "Redundant Via Insertion in Restricted Topology Layouts", in *Proc. ISQED*, pp. 821-828, 2007