Native-Conflict and Stitch-Aware Wire Perturbation for Double Patterning Technology

Shao-Yun Fang, Szu-Yu Chen, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—Double patterning technology (DPT), in which a dense layout pattern is decomposed into two separate masks to relax its pitch, is the most popular lithography solution for the sub-22 nm node to enhance pattern printability. Previous work focused on stitch insertion to improve the decomposition success rate. However, there exist native conflicts (NCs) which cannot be resolved by any kind of stitch insertion. A design with NCs is not DPT-compliant and may fail the decomposition, resulting in design for manufacturability redesign and longer design cycles. In this paper, we give a sufficient condition for the NC existence and propose a geometry-based method for NC prediction to develop an early-stage analyzer for DPT decomposability checking. Then, a wire perturbation algorithm is presented to fix as many NCs in the layout as possible. The algorithm is based on iterative 1-D compaction and can easily be embedded into existing industrial compaction systems. The algorithm is then further applied to further reduce the number of stitches required for the decomposition process. Experimental results show that the proposed algorithm can significantly reduce the number of NCs by an average of 85% and reduce the number of stitches by an average of 39%, which may effectively increase the decomposition success rate for the next stage.**

***Index Terms*—Compaction, double patterning technology, lithography, native conflict, stitch minimization, wire perturbation.**

I. Introduction

FOUNDRIES have been systematically reducing the printed feature size (half pitch) for years. The main challenges for this reduction can be seen from the Rayleigh criterion [20]

$$\frac{P_{\min}}{2} = k_1 \frac{\lambda}{\mathrm{NA}}$$

where $P_{\min}$ denotes the minimum pitch, k_1 denotes the process difficulty factor, λ denotes the wavelength of the light, and NA is the numerical aperture. Currently, the 193 nm wavelength is still the smallest wavelength used in semiconductor production since the technology for extreme ultraviolet is continually delayed. Also, with the highest feasible NA at 1.35 for water immersion lithography, the theoretical limit of the k_1 value is 0.25. Therefore, the half pitch is limited to 36 nm, which cannot satisfy the desired sub-22 nm technology nodes.

The most popular solution for sub-22 nm node is double patterning technology (DPT) [1], [9], [17]. One of many double patterning processes is called the litho-etch-litho-etch process. The two exposures are executed on the same layer and regarded as completely independent. Such a process enables pitch relaxation and hence reduces the effective k_1 factor.

To achieve pitch relaxation, DPT requires decomposition of a dense layout pattern into two sparse patterns; i.e., two features have to be assigned to different masks if the minimum Euclidean distance between them is less than the minimum double patterning spacing (DP-spacing). Note that the pitch size is defined as the distance between the center axes of two features; thus, the minimum pitch size is equivalent to the following:

$$P_{\min} = W_{\min} + S_{\min}$$

where $W_{\min}$ denotes the minimum width and $S_{\min}$ denotes the minimum spacing in a layout. Therefore, we define the DP-spacing as $2P_{\min} - W_{\min}$ in this paper, as indicated in Fig. 1(a). In addition, the DP-spacing can be regarded as a new spacing rule for these two masks.

The DPT decomposition problem can be modeled as a two-coloring problem. First, we construct a conflict graph, where each vertex represents a feature in the layout and an edge exists if the distance between these two corresponding features is less than the DP-spacing. Then, we find a two-coloring solution on the conflict graph, such that no edge connects to two vertices of the same color. Finally, the features are assigned to two masks according to their colors, i.e., each mask contains all the features of the same color. Fig. 1 illustrates a decomposition process. Five features with pitch distance α are shown in Fig. 1(a), and the distance of each pair of $\{(A, B), (B, C), (C, D), (A, E), (C, E)\}$ is less than the DP-spacing. Therefore, in the corresponding conflict graph, the edges are introduced between the two vertices of the pairs, as shown in Fig. 1(b). On the other hand, the distance between A and C is not less than the DP-spacing, indicating these two features can be assigned to the same mask. By finding a two-coloring solution on the conflict graph, we can decompose the

Manuscript received March 13, 2011; revised June 12, 2011, August 22, 2011 and October 9, 2011; accepted November 6, 2011. Date of current version April 20, 2012. This work was supported in part by ITRI, Springsoft, Synopsys, TSMC, and the National Science Council of Taiwan, under Grants NSC 100-2221-E-002-088-MY3, NSC 99-2221-E-002-207-MY3, NSC 99-2221-E-002-210-MY3, and NSC 98-2221-E-002-119-MY3. A preliminary version of this paper was presented at the 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10), San Jose, CA, November 2010 [3]. This paper was recommended by Associate Editor J. Hu.

S.-Y. Fang and S.-Y. Chen are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: yuko703@eda.ee.ntu.edu.tw; aknow@eda.ee.ntu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

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

Digital Object Identifier 10.1109/TCAD.2011.2179039

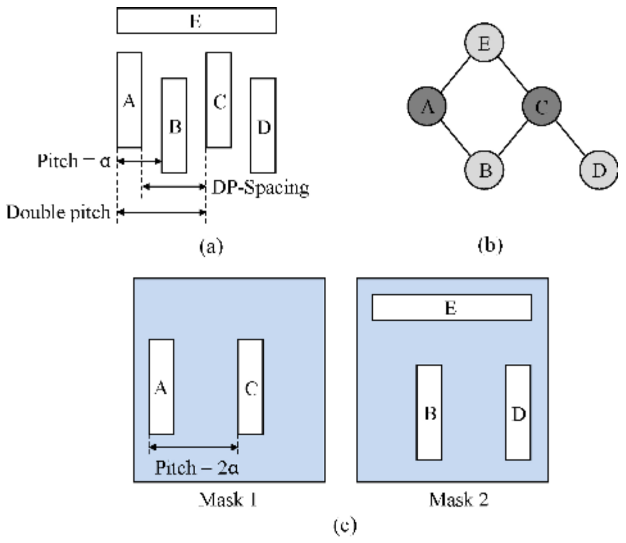


Fig. 1. Decomposition problem is modeled as a two-coloring problem. (a) Five features with pitch distance of α. (b) Conflict graph and its two-coloring solution. (c) By assigning A and C to Mask 1, and B, D, and E to Mask 2, the pitch size is relaxed to 2α.

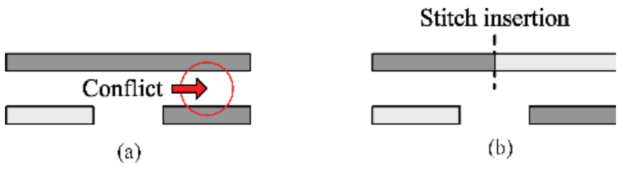


Fig. 2. Stitch insertion. (a) Conflict resulted from two neighboring features being assigned to the same mask. (b) Conflict can be resolved by inserting a stitch that splits the top feature into two parts.

layout into two partitions $\{A, C\}$ and $\{B, D, E\}$, as illustrated in Fig. 1(c).

However, not all the patterns can be directly decomposed into two masks. In some cases, two neighboring features within the DP-spacing might be in the same mask, which is regarded as a conflict in the layout. In order to resolve the conflict, stitch insertion which splits a feature pattern into two parts to be manufactured with different masks can resolve the conflict and thus further improve the decomposition success rate. Fig. 2(a) illustrates a conflict resulted from two neighboring features being assigned to the same mask. The conflict can be resolved by inserting a stitch that splits the top feature into two parts as shown in Fig. 2(b). Nevertheless, stitch insertion may cause yield loss due to overlay errors [16]. Stitch minimization thus becomes an important issue in DPT.

Much previous work focuses on the layout decomposition problem. Chiou *et al.* [4] proposed a rule-based approach for pattern decomposition, which first pre-fragments the patterns and then iteratively colors the polygons in a greedy manner. Kahng *et al.* [15] presented a decomposition flow with conflict graph construction, node splitting, and integer linear programming (ILP) optimization. Also, due to the yield loss causing from stitches, Yuan *et al.* [23] addressed on simultaneous minimization of conflicts and stitches. Xu and Chu [21] proposed an efficient matching-based decomposer that addresses the same objective as [23]. Hsu *et al.* [12], [13] proposed a simultaneous layout migration and decomposition algorithm based on an ILP formulation. Recently, Yang *et al.* [22] proposed an efficient layout decomposition framework by using a graph-theoretical approach. They also extended their work to contact layer decomposition and timing-driven decomposition to reduce the timing variation due to overlay. Yuan *et al.* [24] presented a wire spreading technique to enhance layout decomposition. The work first identifies wire-spreading candidates and then simultaneously minimizes the number of conflicts and stitches with an ILP formulation.

At first glance, the DPT layout decomposition looks similar to the phase assignment problem of alternating phase-shift mask (Alt-PSM). The problem asks to find a set of phase conflicts so that by removing them, we can ensure two-colorable for its conflict graph. However, there are some different characteristics between these two topics, so that the method of Alt-PSM cannot be directly extended to the DPT domain. We observe that the conflict graph used in DPT has a different nature from that in Alt-PSM. The algorithms for finding a set of phase conflicts that need to be removed heavily rely on the dual graph of a planar embedding [2]. The work [21] claims that the DPT conflict graph for a 2X (1.4X) pitch reduction is planar based on the Manhattan (Euclidean) distance. However, the DPT problem should be handled based on the Euclidean distance due to the lithographic nature. Since DPT is a very costly technology, a 1.4X pitch reduction is not economical. (We observe that a conflict graph used in DPT based on the Euclidean distance is not necessarily planar for 2X pitch reduction.) Cho *et al.* also showed a limitation to extend the method of Alt-PSM to the DPT domain: with the stitch insertion, the vertices can be split and hence the corresponding conflict graph might be changed dynamically [5], i.e., stitch insertion provides more flexibility for handling the conflicts.

Nevertheless, the stitch insertion might not be powerful enough for resolving all the conflicts. A conflict that cannot be resolved by inserting stitches anywhere is called a native conflict (NC). Some previous work pointed out the issue of NCs. For example, from the results of decomposition, Oosten *et al.* [18] found that NCs may occur, which prohibits the existence of a DPT solution by any approach, even manually. Chiou *et al.* [4] also stated that the unresolvable conflicts require modifications in the design to achieve a DPT-compliance design. Therefore, dealing with the NC problem is a crucial part of implementing the DPT process. The work [12], [13] first proposed a simultaneous layout migration and decomposition algorithm based on an ILP formulation. However, the ILP formulation is time-consuming, and thus the algorithm can only handle circuits like standard cells and cannot scale to very large-scale designs. In addition, the algorithm used in [24] performs wire spreading to enhance layout decomposability with an ILP formulation. However, the method which only locally spreads conflict patterns might not be effective in resolving NCs for a compact layout. Therefore, a method to predict the occurrences of NCs and a more flexible wire perturbation algorithm is desired.

In this paper, we deal with the NC issue by two stages: 1) NC prediction, which locates the positions of NCs, and 2) NC removal, which perturbs the layout to remove the NCs found in the previous stage. In detail, we give a sufficient condition for the NC existence and propose a geometry-based method for NC prediction to develop an early-stage analyzer for DPT decomposability checking. Then, a wire perturbation algorithm based on iterative 1-D compaction is presented to fix as many NCs in the layout as possible. In addition to NC minimization, we further extend our algorithm flow to minimize required stitches for layout decomposition by reusing the wire perturbation technique. We summarize the major features of the proposed prediction method and wire perturbation algorithm as follows.

1) Our NC prediction method is based on the geometry relation of patterns. It does not rely on an explicit coloring; hence, the running time of prediction can be improved.
2) The proposed NC-aware wire perturbation algorithm based on an iterative 1-D compaction method could efficiently enhance the decomposability and effectively resolve NCs of a layout, which remedies the deficiencies of the previous ILP-based algorithms.
3) The proposed algorithm flow can be extended to further reduce the number of stitches required for layout decomposition, which shows the flexibility of the use of the compaction method.
4) Experimental results show that the proposed wire perturbation algorithm significantly reduces the number of NCs by an average of 85% and the number of stitches by an average of 39%, which may increase the decomposition success rate for the next stage.

The remainder of this paper is organized as follows. Section II provides preliminaries on DPT. Section III proposes an NC-prediction method. Sections IV and V present wire perturbation algorithms to minimize the number of NCs and the number of stitches in the layout, respectively. Experimental results are shown in Section VI, and conclusions are given in Section VII.

II. Preliminaries

Some preliminaries on DPT are provided in this section. Section II-A gives an example of the occurrence of NCs. Then, a DPT-compliance redesign that can remove NCs is introduced in Section II-B. Finally, the overall problem formulation is stated in Section II-C.

A. Native Conflict

Some layout patterns cannot be directly decomposed into two masks, as the layout shown in Fig. 3(a). To prove it, we first give some definitions and then show that there is no two-coloring solution for the conflict graph of this layout.

Definition 1: An *odd cycle* is a cycle with odd number of edges.

It is well-known that a graph with an odd cycle does not have a two-coloring solution, and thus a layout cannot be decomposed into two masks if there is an odd cycle in its corresponding conflict graph [15].

To resolve the odd cycles and improve the decomposition success rate, we can apply stitch insertion. A stitch can split a

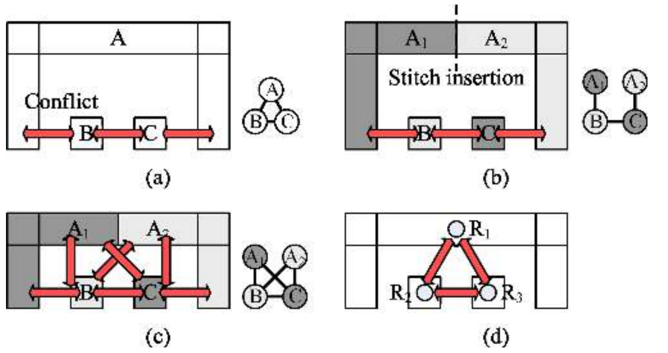


Fig. 3. Challenges in decomposition. (a) Layout cannot be directly decomposed into two masks since there is an odd cycle in its conflict graph. (b) Resolve the odd cycle by stitch insertion, which is regarded as a vertex splitting operation on the conflict graph. (c) Moving the wire A downward causes the stitch insertion failing to resolve the odd cycle in this layout. (d) Odd cycle from arbitrarily small point regions, R_1, R_2, and R_3, cannot be resolved by any kind of stitch insertion, indicating the existence of NCs.

feature into two parts, and the corresponding vertex in the conflict graph is also split into two vertices. As shown in Fig. 3(b), a stitch is inserted on A and split the feature into A_1 and A_2; hence, the odd cycle in the conflict graph is eliminated, and the layout can be decomposed into two masks.

Assume that the wire A in Fig. 3(a) is moved downward as illustrated in Fig. 3(c), and the same stitch is also inserted on wire A to attempt to break the odd cycle. However, after splitting A into A_1 and A_2, the distance between features A_1 and C is still less than the DP-spacing. Thus, the new odd cycle $< A_1, B, C >$ forms in the conflict graph. Also, the same situation occurs for features A_2 and B. The combination of these two situations indicates that the original odd cycle $< A, B, C >$ is unresolvable.

The reason for the existence of unresolvable odd cycle $< A, B, C >$ is explained in Fig. 3(d). Considering three point regions R_1, R_2, and R_3 on wires A, B, and C, respectively, and a point region is regarded as an arbitrarily small region. We found that any position in R_1 is within the DP-spacing from R_2 or R_3. Therefore, even we split R_1 into two parts, each of these two parts still forms an odd cycle with R_2 and R_3. As a result, it is impossible to break this odd cycle by stitch insertion, and the conflict is "native" in the layout. The NC is defined as follows:

Definition 2: A NC is a conflict that must exist in the layout, and the conflict cannot be resolved by any kinds of decomposition methods even with stitches.

From the discussion above, we know that the cause of a NC is an odd cycle forming by the neighboring point regions within the DP-spacing on different wires. However, predicting the odd cycle is not easy since there could be many combinations for selecting those point regions. Therefore, developing a systematic prediction method is desirable. We show that it can be achieved by a pattern projection and an odd cycle detection approach.

B. DPT-Compliant Redesign

A NC can only be resolved by DPT-compliant redesign. Fig. 4 illustrates a wire perturbation solution for NCs. The

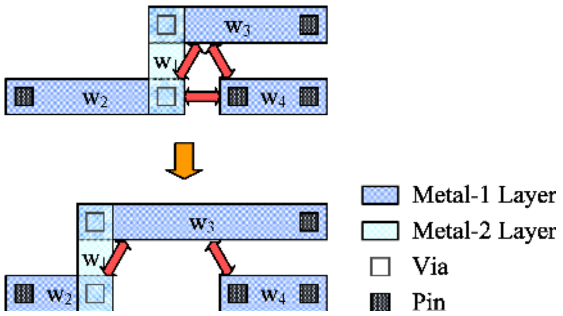


Fig. 4. NC can only be resolved by DPT-compliant redesign. By moving the wire w_1 leftward, and separating wires w_2 and w_4, the odd cycle is eliminated.

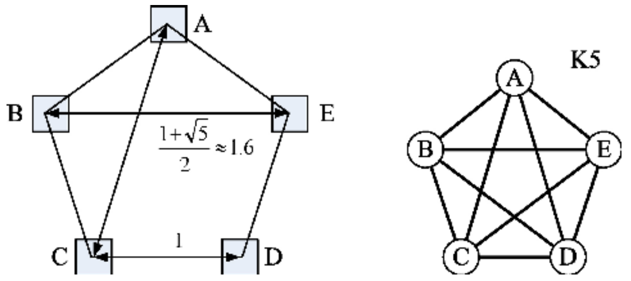


Fig. 5. Example showing a conflict graph used in DPT is not necessarily a planar graph.

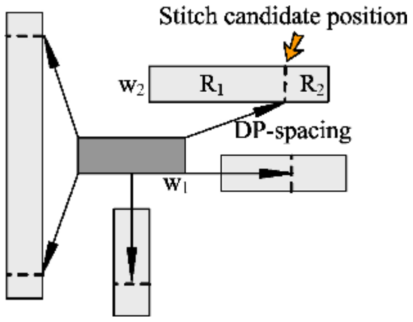


Fig. 6. Stitch candidate positions can be determined by projection from wire w_1 to its neighboring wires. For example, the intersection point on w_2 is a candidate position since it splits the wires into two regions, R_1 and R_2, where one region is within DP-spacing from w_1, and another is not.

wires w_2, w_3, and w_4 are on the metal-1 layer and form a NC, due to their neighboring endpoints. To overcome this NC, the wire w_1 on the metal-2 layer is moved leftward. Thus, the odd cycle is eliminated because the right end of w_2 is shortened and separated from w_4.

One of the most important stages for DPT-compliant redesign is identifying real NCs from all the odd cycles since the others can be removed in the decomposition process. That is, DPT-compliant redesign should first predict the occurrence and the locations of NCs, and then further correct them. Therefore, an analyzer for NC prediction is desirable and crucial for developing a true DPT-friendly design.

As mentioned in Section I, there are some limitations to extend the methods of Alt-PSM to the DPT domain. One is the flexibility provided by stitch insertion for DPT. Another is that the conflict graph used in DPT may not be planar (based on the Euclidean distance), different from that in Alt-PSM. We show the fact with the following example shown in Fig. 5. In this example, we assume that the DP-spacing is $2P_{\min} - W_{\min}$ as defined in Section I. Fig. 5 shows that five features lie on the corners of a pentagon with the minimum pitch size being 1. There are totally two kinds of pitch sizes in the layout, 1 and $\frac{1+\sqrt{5}}{2}$. Both of them are smaller than 2 (twice of the minimum pitch size). Therefore, if we construct the conflict graph, it will form a $K5$ (the complete graph with five vertices). A graph with a $K5$ is known not planar.

C. Problem Formulation

In this paper, we develop a DPT-compliant redesign system by wire perturbation. The problem can be defined as follows. Given a postrouting layout, the minimum spacing, and the DP-spacing, finds a perturbed layout so that the number of NCs and the number of stitches required for layout decomposition are minimized, subject to the minimum-spacing rule and the double-patterning constraint. In addition, the locations of pins should not be changed and the layout boundary should not be enlarged after the perturbation.

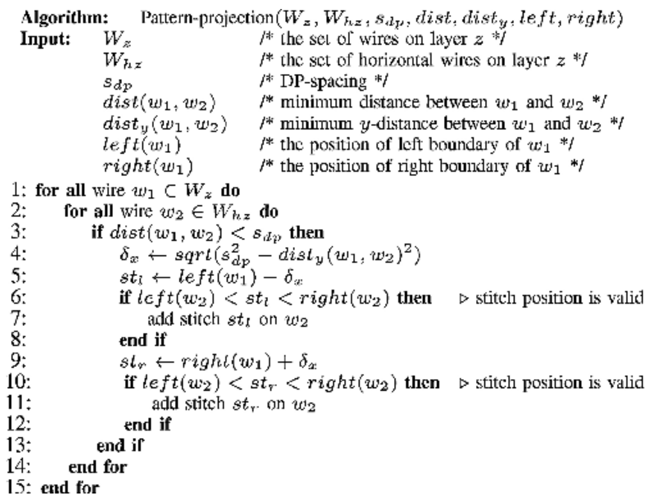
In order to minimize the number of NCs in the layout, two steps are necessary. One is a method for prediction and measurement of NCs, and another is perturbation algorithm that can modify the layout to remove them. These two steps are described in Sections III and IV, respectively. Also, the extended wire perturbation technique for stitch minimization is presented in Section V.

III. NC Prediction

In this section, we propose a geometry-based method for NC prediction, which assists the DPT-compliant redesign in the later stage. As discussed in Section II, an odd cycle in the conflict graph is not necessarily a NC because of the possible solution by stitch insertion. Therefore, to utilize the capability of stitch insertion, the prediction method first exploits the set of stitch candidate positions by pattern projection. It guarantees that the optimal stitch combination with the fewest conflicts is within this set. That implies that the results found by our prediction method are indeed NCs since we have already considered the possibility of resolving them. Then, based on the projection results, a segmentation stage is performed to split a feature into several tiles, and a sufficient condition for the NC existence is stated that the NCs can be examined by finding the odd cycles on the tile conflict graph, which is different from the previous feature conflict graph. Finally, an odd cycle detection algorithm is given to locate the NCs.

A. Pattern Projection

The set of stitch candidate positions is found by pattern projection, which is a similar process used in some previous works [9], [15]. Fig. 6 illustrates the projection process. Considering the wire w_1, its neighboring wires can be split into several parts according to the conflict relation with w_1. For example, the wire w_2 is split into two regions, R_1 and R_2, where R_1 conflicts with w_1 and cannot be assigned to the



```
Algorithm:  Pattern-projection($W_z$, $W_{hz}$, $s_{dp}$, $dist$, $dist_y$, $left$, $right$)
Input:  $W_z$                  /* the set of wires on layer $z$ */
        $W_{hz}$               /* the set of horizontal wires on layer $z$ */
        $s_{dp}$               /* DP-spacing */
        $dist(w_1, w_2)$       /* minimum distance between $w_1$ and $w_2$ */
        $dist_y(w_1, w_2)$     /* minimum $y$-distance between $w_1$ and $w_2$ */
        $left(w_1)$            /* the position of left boundary of $w_1$ */
        $right(w_1)$           /* the position of right boundary of $w_1$ */
1:  for all wire $w_1 \subset W_z$ do
2:      for all wire $w_2 \in W_{hz}$ do
3:          if $dist(w_1, w_2) < s_{dp}$ then
4:              $\delta_x \leftarrow sqrt(s_{dp}^2 - dist_y(w_1, w_2)^2)$
5:              $st_l \leftarrow left(w_1) - \delta_x$
6:              if $left(w_2) < st_l < right(w_2)$ then    ▷ stitch position is valid
7:                  add stitch $st_l$ on $w_2$
8:              end if
9:              $st_r \leftarrow right(w_1) + \delta_x$
10:             if $left(w_2) < st_r < right(w_2)$ then    ▷ stitch position is valid
11:                 add stitch $st_r$ on $w_2$
12:             end if
13:         end if
14:     end for
15: end for
```

Fig. 7. Algorithm of pattern projection to horizontal wires.

same color, while R_2 is not. Accordingly, the position that splits the wire w_2 is considered as a stitch candidate position. The method is detailed as follows. We draw a line with its length equal to the DP-spacing from the corners and edges of w_1. The intersections of the lines and the neighboring wires are the candidate positions. Also, each split part belongs to one of these two categories: 1) within the DP-spacing from w_1, and 2) out of the DP-spacing from w_1. Notice that the projection is performed only once since the positions obtained by the projection from the first-time stitches are not needed for odd cycle minimization. (The first-time stitches are the positions determined by our projection method, and the positions that can be projected from the first-time stitches are referred to as the second-time stitches.)

The algorithm of pattern projection from wires to their neighboring horizontal wires is summarized in Fig. 7. The algorithm for the projection to vertical wires is similar. In the algorithm, a wire w_1 is first selected. Then, we consider all the other wires w_2 within the DP-spacing from w_1. Finally, two stitch positions are computed and added if they are valid for w_2. Note that for bent polygons, we first segment them into horizontal and vertical wires, and then perform pattern projection.

We have the following theorem for the set of stitch candidate positions found by the pattern projection.

Theorem 1: There exists an optimal stitch combination within the set of stitch candidate positions generated from the algorithm in Fig. 7 that introduces the fewest conflicts.

Proof: As shown in Fig. 8(a), the set of stitch candidate positions is $\{A, B\}$. Assume that in the optimal solution, there is a stitch C between A and B, and C is not contained in the set. There must be one of the following two cases.

1) *Case A:* If the color of the region R_3 can be the same as one of R_4, C can be replaced by A or B without causing any conflicts. Therefore, an optimal stitch combination can be found within the set $\{A, B\}$, and thus adding C is redundant.
2) *Case B:* If the color of the region R_3 cannot be the same as one of R_4, it implies that there must be a region R_1 which conflicts with R_3, but not R_4, or vice versa. Therefore, C must lie in the position that can be projected from the edge of a region R_1, and that edge must be a wire boundary or a first-time stitch.
   a) *Case B1:* If the edge is a wire boundary, as in Fig. 8(a), the position C will be marked when the projection is applied from this wire boundary to its neighboring wires. Thus, this case violates our assumption that C is not contained in the set of stitch candidate positions.
   b) *Case B2:* If the edge is a first-time stitch D, as in Fig. 8(b), the colors R_1 and R_2 are different. Therefore, R_3 is both within the DP-spacing from R_1 and R_2, and it must cause one conflict. Thus, it means that C is not helpful in resolving the conflicts and odd cycles, and we can remove C and keep the same conflict configuration. ■

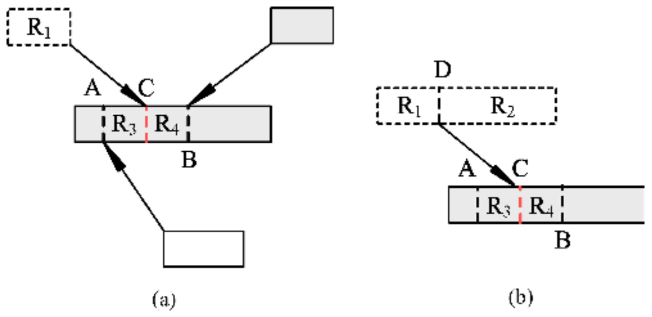


Fig. 8. Set of stitch candidate positions computed by pattern projection is optimal for conflict minimization. (a) Additional stitch C between A and B must be projected from a region R_1. Therefore, C is in the candidate set. (b) Position projected from a stitch is not helpful for resolving the conflict. Even with stitches C, R_3 still conflicts with R_1 and R_2. Therefore, removing the stitch C also keeps the same conflict configuration.

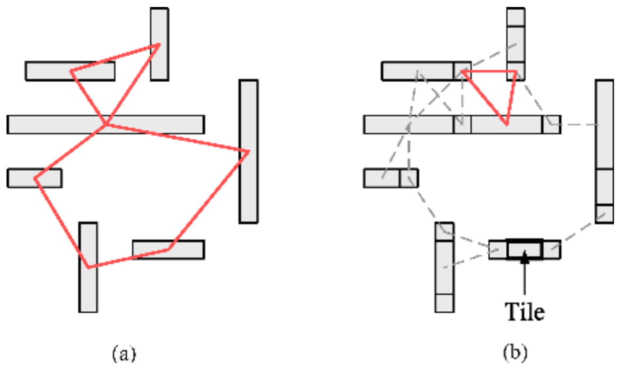


Fig. 9. Differences between feature conflict graph and tile conflict graph. (a) Feature conflict graph with two odd cycles. (b) After projection, wires are split into several tiles and form a tile conflict graph. Only the top odd cycle incurs a NC.

B. Segmentation

After the pattern projection, a wire is split into several parts. We refer to the part on a wire as a tile in the remainder of this paper.

Definition 3: A *tile* is a region on a wire enclosed by two neighboring stitches or a boundary edge of a wire with its neighboring stitch.

Then, a tile conflict graph is constructed. Different from the feature conflict graph, each tile is represented by a vertex in the graph, and an edge exists if two tiles from two different features are within the DP-spacing. Fig. 9(a) shows a feature conflict graph. After segmentation, the results and its corresponding tile conflict graph are shown in Fig. 9(b).

C. Condition of NC Existence

Based on the tile conflict graph, a sufficient condition of NC existence is stated as follows.

Theorem 2: Every odd cycle on the tile conflict graph is a NC.

Proof: From Theorem 1, the set of stitch positions obtained by pattern projection is optimal for minimizing the number of conflicts. It means that inserting any stitches to split tiles will not further reduce the number of odd cycles in its conflict graph. In other words, the odd cycles on the tile conflict graph cannot be resolved by stitch insertion, and they are indeed NCs. ■

Fig. 9(a) shows a feature conflict graph with two odd cycles. However, after segmentation and the construction of the tile conflict graph as shown in Fig. 9(b), only the top odd cycle is detected as a NC, which indeed cannot be resolved by stitch insertion.

D. Odd Cycle Detection

An odd cycle in the tile conflict graph indicates that there is a NC in the layout. The existence of odd cycles can be determined by a DFS algorithm. However, we still need to know how many odd cycles are in the layout to measure the level of DPT-friendly for a design. Unfortunately, the total number of odd cycles in a graph could be exponential. Thus, finding all the odd cycles is time-consuming. Therefore, an easy-to-compute metric is desired. We show that the number of odd cycles in a *cycle basis*, which is a well-known concept [7], is good enough for guiding our perturbation system since we can achieve odd cycle free by making the number of odd cycles in cycle basis zero.

Definition 4: A *basis* is a set of vectors that, in a linear combination, can represent every vector in a given vector space. In addition, no element of the set can be represented as a linear combination of the others. In other words, a basis is a linearly independent spanning set.

Definition 5: A *cycle basis* is a basis if we treat every cycle as a vector represented by a series of edges, and the linear combination of cycles is a ring sum operation of cycles. Then the basis form a linearly independent spanning set for all cycles.

Definition 6: Let the edges in the cycles C_1 and C_2 be sets E_1 and E_2, respectively. A *ring sum operation* on C_1 and C_2 is denoted as $C_1 \oplus C_2$, which is a cycle with its edge being a set, $(\bar{E}_1 \cap E_2) \cup (E_1 \cap \bar{E}_2)$ (symmetric difference).

Definition 7: A ring sum operation on cycles $C_1, C_2, \ldots, C_n$ is denoted as $\bigoplus\{C_1, C_2, \ldots, C_n\}$, which equals $((C_1 \oplus C_2) \oplus \ldots) \oplus C_n$. The operation is commutative and associative.

Fig. 10 illustrates the cycle basis and the ring sum operation. There are totally three cycles in the graph, $< A, B, C >$, $< A, C, D >$, and $< A, B, C, D >$. Any two cycles consist of a cycle basis, since the remaining one cycle can be obtained by performing the ring sum operation on those two cycles in the basis.

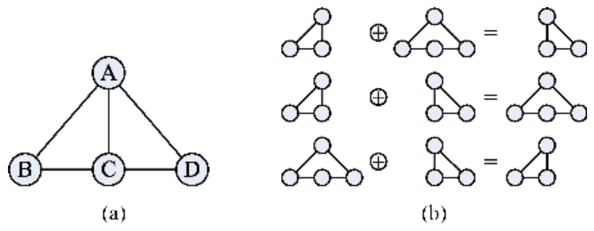
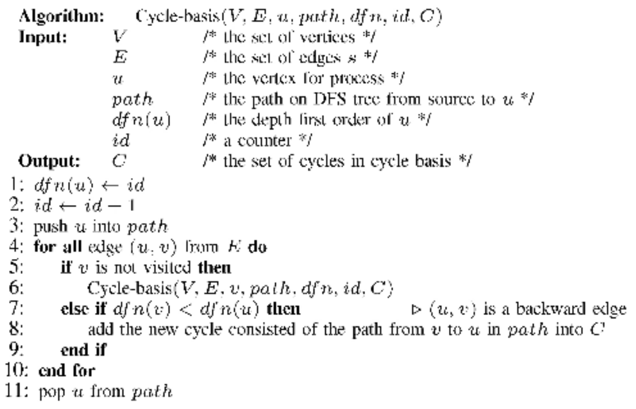


Fig. 10. Cycle basis and ring sum operation. (a) There are totally three cycles in the graph, $< A, B, C >$, $< A, C, D >$, and $< A, B, C, D >$. (b) Any two cycles consist of a cycle basis, since the remaining one cycle can be obtained by performing the ring sum operation on those two cycles in the basis.



```
Algorithm:  Cycle-basis($V, E, u, path, dfn, id, C$)
Input:      $V$         /* the set of vertices */
            $E$         /* the set of edges $s$ */
            $u$         /* the vertex for process */
            $path$      /* the path on DFS tree from source to $u$ */
            $dfn(u)$    /* the depth first order of $u$ */
            $id$        /* a counter */
Output:     $C$         /* the set of cycles in cycle basis */
 1: $dfn(u) \leftarrow id$
 2: $id \leftarrow id - 1$
 3: push $u$ into $path$
 4: for all edge $(u, v)$ from $E$ do
 5:     if $v$ is not visited then
 6:         Cycle-basis($V, E, v, path, dfn, id, C$)
 7:     else if $dfn(v) < dfn(u)$ then            ▷ $(u, v)$ is a backward edge
 8:         add the new cycle consisted of the path from $v$ to $u$ in $path$ into $C$
 9:     end if
10: end for
11: pop $u$ from $path$
```

Fig. 11. Algorithm of finding the cycle basis.

Cycle basis of a graph can be found by applying a modified DFS algorithm [6], [19]. For a connected graph $G = (V, E)$, the result of DFS traversal forms a spanning tree T on V. Inserting any other edge $e \in (E - T)$ causes a cycle. Therefore, there exists a cycle basis consisting of those cycles, since every cycle has an edge (the edge we inserted) that is not contained in other cycles, and hence the cycle cannot be generated by a linear combination of others. Fig. 11 shows an algorithm for finding the cycle basis.

The number of odd cycles and that of odd cycles in cycle basis are in a positive correlation. Furthermore, the following theorem states that we can achieve odd cycle free by making the number of odd cycles in cycle basis zero.

Theorem 3: If there is no odd cycle in the cycle basis of a graph, then the graph contains no odd cycle.

Proof: Assume that there are two cycles, A and B, with the lengths of l_a and l_b, respectively. Then, the new cycle $C = A \oplus B$ must be with the length of $l_a + l_b - 2k$, where k is an integer. Therefore, if there is no odd cycle in the cycle basis, all the cycles spanned by the cycle basis must be with the length of an even number. ■

With Theorem 3 and the fact that the number of odd cycles and that of odd cycles in cycle basis are in a positive correlation, the number of odd cycles is approximated by that of odd cycles in cycle basis. Note that the computation can be

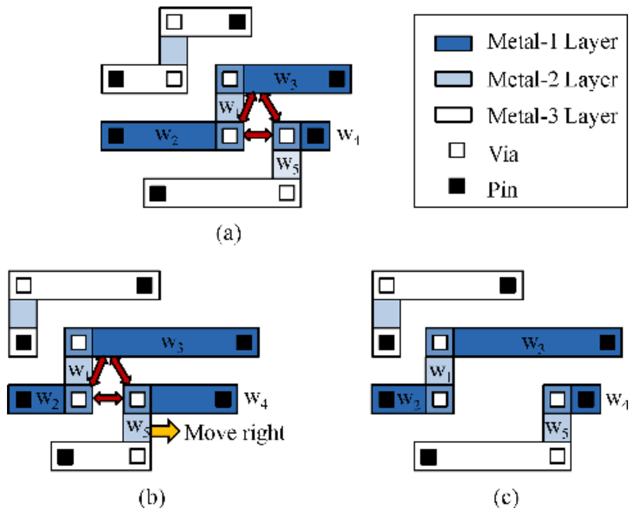


Fig. 12. Compaction approach is suitable for NC-aware wire perturbation problem since it can globally redistribute the spacing between patterns. (a) NC caused by the end points of wires w_2, w_3, and w_4. However, it is impossible to resolve this conflict by moving the wire w_1 or w_5. (b) Performing the x-directional compaction releases the resources that make the wire w_5 could be moved right more. (c) After moving wire w_5 rightward, the NC is resolved.

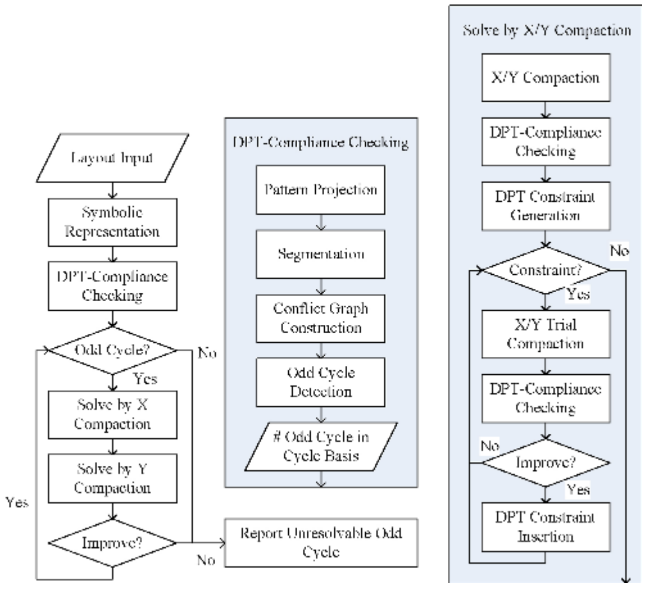


Fig. 13. Overall flow for our NC-aware wire perturbation algorithm.

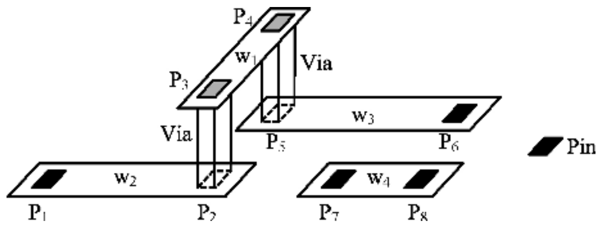


Fig. 14. Symbolic layout representation is used in our compaction system. Each wire is represented by two points. The wire $w_1(P_3, P_4)$ is on the metal-2 layer, while $w_2(P_1, P_2)$, $w_3(P_5, P_6)$, and $w_4(P_7, P_8)$ are on the metal-1 layer. In addition, there are two vias, (P_2, P_3) and (P_5, P_4).

done by a modified DFS algorithm; thus it is both effective and efficient.

IV. NC-Aware Wire Perturbation

Pattern decomposition for the metal layer is one of the most challenging problems, due to the complex 2-D metal design. The complex layout patterns might cause a large number of NCs that cannot be resolved. Therefore, it is desirable to incorporate the DPT-compliant issue into the current design flow. In this section, we propose a wire perturbation algorithm on metal layers to convert a layout into a decomposable one. The perturbation algorithm is based on iterative 1-D compaction and can easily be embedded into an existing industrial compaction system.

The compaction approach is suitable for this problem since it can globally redistribute the spacing between patterns to intelligently handle the minimum-spacing constraint and double patterning constraint. Fig. 12 illustrates an example of its power. In Fig. 12(a), there is a NC caused by the end points of wires w_2, w_3, and w_4. To resolve this conflict, the wire w_1 should be separated from the wire w_5. However, the wire w_1 cannot be moved leftward or rightward; otherwise, it violates the spacing rule. For another choice, w_5, although it can be moved rightward, the spacing is still not enough for separating w_2 and w_4. Considering performing the x-directional compaction as Fig. 12(b), all the vertical wires on metal-2 layer are moved left as much as possible. The resources are now released, and w_5 has the capability to move right. Then, we separate w_2 and w_4 by moving w_5 rightward as shown in Fig. 12(c). Finally, the NC is resolved.

The objective of the proposed wire perturbation algorithm is to fix as many NCs in the layout as possible. The number of NCs is measured by the number of odd cycles in the cycle basis of the tile conflict graph of the layout. (We use the term odd cycle basis to indicate odd cycles in the cycle basis.) We have shown that this number is a significant metric and can be computed by a linear-time DFS algorithm; thus, it can effectively and efficiently guide our perturbation algorithm.

The overall flow of our perturbation algorithm is depicted in Fig. 13. Given a postrouting layout, we first construct its symbolic representation, which is the fundamental data structure in our system for representing a layout and is appropriate for the operations of perturbations. Then, a decomposability checking described in Section III is performed to detect the NCs in the layout. We then use a 1-D compaction-like approach to resolve the NCs we found. In each compaction process, we adjust the distances between wires on one dimension such that the number of odd cycles can be reduced as many as possible. Each step is detailed in the following sections.

A. Symbolic Layout Representation

In order to maintain the connections through vias, some wires should be shortened or stretched when others are moved. Therefore, the symbolic layout representation is used in our compaction system. As shown in Fig. 14, every wire (via) is represented by two points. (Note that in the routing result of DEF format, L, U, T, W shape wires are represented by a series of rectangular wires.) The operation of moving a wire is achieved by moving the points. For example, if we move four points (P_2, P_3, P_4, P_5) of two vias leftward, the wire w_1 is moved leftward automatically since its location is recomputed according to P_3 and P_4. Also, the left end of the wire w_3 is stretched and the right end of the wire w_2 is shortened. With the symbolic representation, we can perform the compaction across layers simultaneously.

B. DPT-Compliance Checking

We implement the NC prediction method described in Section III. The flow for DPT-compliance checking is depicted in Fig. 13. First, the pattern projection is performed to exploit the set of stitch candidate positions. After that the features are segmented to several tiles according to the stitch positions on each feature. Then, we construct a tile conflict graph, and detect the odd cycles on the graph. Finally, the number of odd cycle in cycle basis is reported to measure the level of DPT-compliance of the layout.

C. Wire Perturbation

Fig. 13 shows a flow for resolving the NCs by x/y-directional wire perturbation. We use x-direction here to describe the main idea of flow. First, we compact the layout along the x-direction, and then perform the DPT-compliance checking method to find the locations of NCs. Every NC consists of an odd cycle. By separating two corresponding wires for an edge, the odd cycle can be eliminated. Thus, for all the odd cycles, the DPT constraints that model these separating operations are generated. Then, we add the constraints into our compaction system, and perform a trial compaction to see whether the level of DPT-compliance is improved. The trial compaction is an incremental approach, i.e., we just consider the new constraint's effect on the current result. Therefore, by implementing an event-driven approach, and propagating from the new constraint, the running time of this trial compaction can be speeded up. After the trial compaction, the DPT constraint is permanently added into the system if it makes a better decomposability. Otherwise, all locations of wires are restored. Then, we move to the next DPT constraint, and consider its effect in the same manner.

The detailed method for the DPT constraint generation is described as follows. We enumerate the separating wire pairs that can resolve the odd cycles and perturb the layout by separating the most beneficial wire pair, which can resolve several odd cycles at a time. The separating operation is modeled as an edge in a constraint graph used in the compaction system; hence, it is easy to incorporate our NC-aware perturbation into an existing industrial compactor.

In order to perform this wire perturbation method, two core parts, compaction algorithm and DPT constraint generation, are needed. We describe them in Sections IV-D and IV-E, respectively.

D. Compaction

As in most existing work [10], the compaction problem is modeled as a longest-path problem on the constraint graph, which records the topology information of a layout and the minimum allowable distance between two wires. The longest-path distance from a super source to each wire is the new

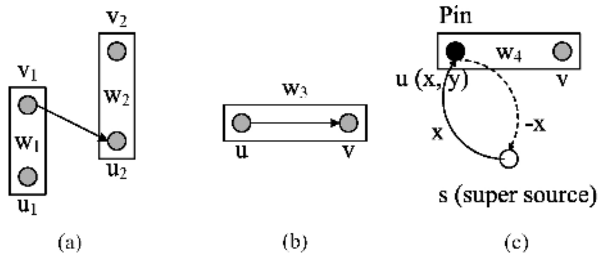


Fig. 15. Three types of constraints are modeled as an edge in the point constraint graph. (a) Design-rule constraint that ensures the minimum spacing between w_1 and w_2. (b) Wire-shape constraint that keeps the wire w_3 being horizontal. (c) Fixed-pin constraint that ensures the pin location not be changed during the perturbation.

location for that wire after compaction. In our compaction system, the positions of points are adjusted, and then the wires are reconstructed from the locations of points to which they connect. Therefore, a point constraint graph is needed. We first generate a wire constraint graph and use the graph and the nature of layout to help us build the desired point constraint graph. Note that to maintain the correct topology of a final layout in our compaction system, we need to simultaneously consider all patterns on the same layer instead of separately considering patterns on two masks after decomposition.

1) *Wire Constraint Graph Construction:* In this stage, a wire-to-wire constraint graph is constructed. We use the x-directional constraint graph to describe the main idea. (The construction for the y-directional constraint graph is similar.) The graph is a directed graph, where each vertex represents a wire in the layout, and an edge from u to v or (u, v) indicates that the wire u is on the left of v. Thus, the topology relation of wires can be recorded in the constraint graph. Note that the layout boundary is modeled as four wires, left-most, right-most, top-most, and bottom-most, to measure the area change after the layout compaction.

To reduce the size of the constraint graph (e.g., the number of edges) without loss of information, the transitive edges in the graph can be removed. In other words, if there are two edges (u, v) and (v, w), then the transitive edge (u, w) is redundant. Doenhardt and Lengauer [8] proposed a $O(n \lg n)$-time algorithm for generating nonredundant constraint graph by using perpendicular sweep lines.

2) *Point Constraint Graph Construction:* The topology relation of points is recorded in the point constraint graph. Similar to wire constraint graph, the edge (u, v) with its weight w indicates that the point v should be on the right of point u, and the distance between them should be at least w. (A negative weight means that v is on the right of u.)

According to the nature of layout, there are four types of constraints: 1) design-rule constraint; 2) wire-shape constraint; 3) fixed-pin constraint; and 4) layout-boundary constraint. Below, we describe how to embed these four constraints in the graph, see Fig. 15 for an illustration.

a) *Design-rule constraint:* The distance between any two nontouching wires in the same layer should be larger than a given value, the minimum spacing. The constraints are read from the wire constraint graph, i.e., any edge in the wire constraint graph is converted to edges in the point constraint graph. Although there are four points for two wires, we only need to add one edge in the point constraint graph. Fig. 15(a) shows a wire w_1 on the left of wire w_2. Then, we just add an edge from the top/right point of w_1 to the bottom/left point of w_2 and set the weight as the minimum spacing. Note that the values of the minimum spacing differ for side-to-side, side-to-line-end, and line-end to line-end spacing. In addition, if there are two or more constraints of different types (e.g., side-to-side and side-to-line-end) between two points, all constraints need to be satisfied simultaneously.

b) *Wire-shape constraint:* The wires should maintain their horizontal or vertical shapes and also the via property during the compaction. Fig. 15(b) shows a horizontal wire w_3 which connects to two points, u and v (u is the left one). This wire should always be horizontal, i.e., the x-coordinate of v must be larger than the one of u, and the y-coordinates of u and v should be the same. Therefore, we add the edge (u, v) and two edges (u, v), (v, u) in the x-directional and y-directional point constraint graphs, respectively (all with weight 0). Note that we treat each via as a wire, and convert it to the constraints. Both the x and y-coordinate of two end points of a via should be the same, so that the upper layer and lower layer could be aligned correctly during the compaction. That means our method can simultaneously perform the compaction on all layers, and maintain the connection through vias.

c) *Fixed-pin constraint:* The locations of pins should not be changed during the perturbation. Fig. 15(c) shows a pin point u located at (x, y). Then, to model the fixed-location property, the edge from the super source s to u with weight x, and a backward edge from u to s with weight $-x$ are added into our x-directional point constraint graph. The process for the y-directional point constraint graph is the same.

d) *Layout-boundary constraint:* The layout boundary should not be enlarged after compaction. Therefore, we modeled the layout boundary as four wires and maintain their topology relationship with other wires. Then, we just treat every endpoint of these four wires as a pin point and apply the fix-pin constraint. Note that the minimum distance between other wires and boundary wires is set to zero instead of the minimum design-rule spacing. By the above setting, it ensures that the actual layout boundary may only be only be smaller but not larger.

Fig. 16 illustrates the final point constraint graph for Fig. 14. Note that for ease of implementing the longest-path algorithm to compute the new locations after compaction, the edges are classified into two groups: 1) forward edges, and 2) backward edges. The backward edges are marked with dashed lines in the figure.

3) *Longest-Path Algorithm:* After the construction of the point constraint graph, the longest-path algorithm is performed to compute the location of each point. The translations of wire-shape constraints and fixed-pin constraints may introduce some backward edges, which incurs cycles in the graph. Therefore, we perform the algorithm which can handle cycles to compute the location of each point (Bellman–Ford or Liao–Wong [10]). If there is no feasible solution, the set of constraints is mutually conflicting, and thus it is impossible to obtain a desired layout.

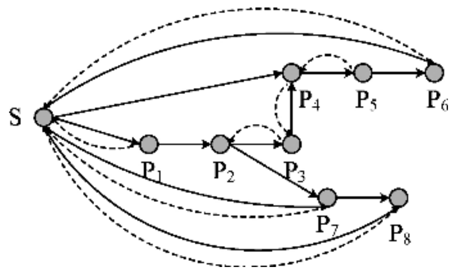
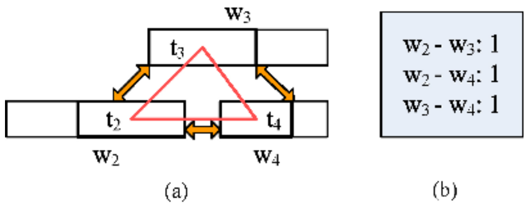


Fig. 16. Overall point constraint graph for Fig. 14. The edges with dashed lines are backward edges.



w_2 - w_3: 1
w_2 - w_4: 1
w_3 - w_4: 1

Fig. 17. Separating wire pairs generation. (a) Three tiles t_2, t_3, and t_4 on wires w_2, w_3, and w_4, respectively, forms an odd cycle. The cycle can be broken by separating (w_2, w_3), (w_2, w_4), or (w_3, w_4). (b) Wire pairs solution table. The number denotes the occurrence of the pair since one pair may be added several times by different odd cycles. The higher the value indicates the higher potential to resolve the odd cycle by breaking that edge.

E. DPT Constraint Generation

DPT constraints are the additional constraint edges in the point constraint graph that can help the compaction system further relax the DPT hotspots. We first generate a set of candidate separating wire pairs from the tile conflict graph. Each wire pair corresponds to one edge in the odd cycle. By separating these two wires, the edge in the conflict graph can be broken. Then, the wire pairs are translated into edges in the point constraint graph so that they can be inserted into the graph, hence guiding the compaction system.

1) *Separating Wire Pairs Generation:* After compaction and DPT-compliance checking, the odd cycles on the tile conflict graph are reported. To relax the conflict between two tiles, the wire pairs formed by the edges corresponding to one of the edges in the odd cycle should be separated. Notice that for a corner tile, it corresponds to two wires, the horizontal one and the vertical one. Both of them can be adjusted to resolve the conflict between tiles.

Fig. 17(a) illustrates how the candidate separating wire pairs are generated. There is an odd cycle formed by the tiles t_2, t_3, and t_4 on the tile conflict graph. The cycle can be eliminated by breaking any one of its edges. Thus, according to the corresponding wires for these tiles, we have three possible wire pairs that can be separated to break the conflict edges, (w_2, w_3), (w_2, w_4), and (w_3, w_4). Then, we enumerate all these

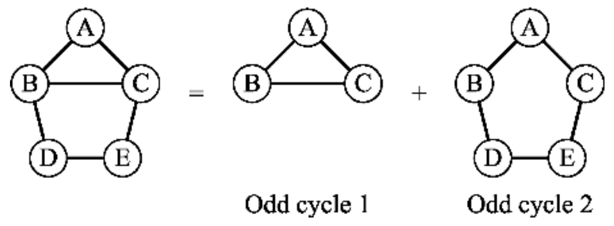


Fig. 18. Breaking edge (B, C) can only resolve odd cycle 1. Choosing the common pairs (A, B) or (A, C) can resolve two cycles at a time.

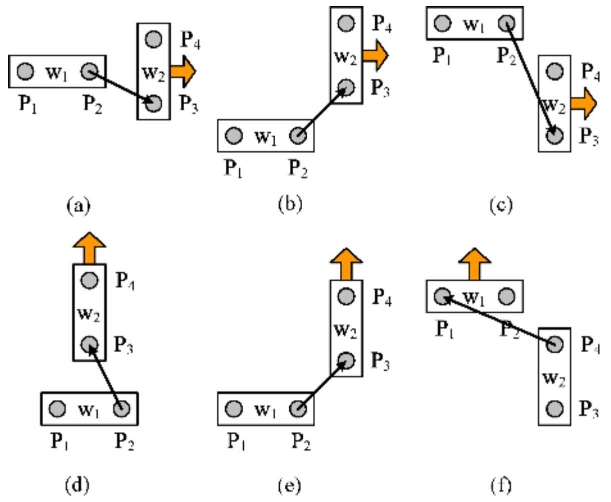


Fig. 19. Separating the wires w_1 and w_2 can be achieved by adding several kinds of point constraint edges according to their topologies. (a)–(c) Topologies that can be resolved by x-directional compaction. (d)–(f) Topologies that can be resolved by the y-directional compaction.

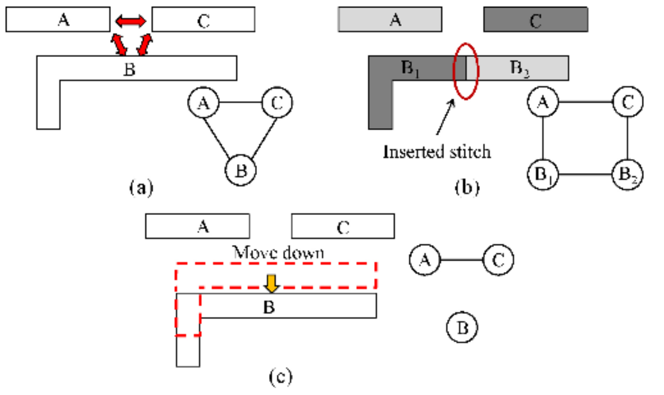


Fig. 20. Methods to resolve odd cycles formed by features. (a) Three patterns conflicting with each other and the corresponding feature conflict graph. (b) Inserting a stitch can resolve the conflict. (c) By perturbing the feature B, the conflict can be resolved without inserting a stitch.

kinds of separating wire pairs from the odd cycles in the graph, and record them in a wire pair solution table. The occurrence of each wire pair is also recorded since it may be added several times by different cycles. Fig. 17(b) shows a solution table for the wire pairs generated from Fig. 17(a).

Wire pairs in the solution table are sorted by their occurrence, since it is a significant index for their potential to resolve the odd cycles. For example, in Fig. 18, there are totally two odd cycles in the graph. To resolve the odd cycle 1, $< A, B, C >$, we can break any one of three edges. However, if the edge (B, C) is selected by us, even the odd cycle 1 is eliminated, the odd cycle 2, $< A, B, D, E, C >$ still exists. It needs to break one more edge to make the layout decomposable. With the information of occurrence, we found that the pair (A, B) and pair (A, C) are added twice, and they are the common separating wire pairs for odd cycles 1 and 2. Therefore, if we separate wire A from B, two odd cycles can be resolved at a time. In this stage, the wire pair with high occurrence is chosen first, i.e., the most beneficial pair is considered.

2) *DPT Point Constraint Edge:* In this stage, the selected separating wire pair is translated into an edge in the point constraint graph so that it can be inserted into our compaction system. Based on the topology relation for the two wires, there are several kinds of solutions to separate them. Fig. 19 illustrates the topologies and their corresponding point constraint edges. Without loss of generality, we assume that w_1 is a horizontal wire and w_2 is a vertical wire, and we are going to separate them. Fig. 19(a)–(c) shows three kinds of topologies that can be resolved by the x-directional compaction, and the cases that can be resolved by the y-direction is shown in Fig. 19(d)–(f). In detail, w_1 and w_2 in Fig. 19(a) are in a left-right relation; hence in order to separate them, we should move w_2 rightward by adding the edge (P_2, P_3) in the x-directional point constraint graph. For the cases that w_2 is on the top-right of w_1 shown in Fig. 19(b) and (e), we have two solutions to separate them, moving w_2 rightward or upward. The operation can be achieved by adding an edge from P_2 to P_3 in the x-directional or y-directional point constraint graphs. Fig. 19(d) and Fig. 19(c) and (f) show the respective remaining cases for w_1 on the bottom of w_2 and w_1 on the top-left of w_2. For both cases, we can separate two wires by adding the edge into the appropriate point constraint graphs.

V. Stitch Minimization with Wire Perturbation

To reduce the yield loss due to overlay errors, the second objective of DPT-compliant redesign is minimizing the number of inserted stitches. Therefore, after optimizing the number of NCs in a layout, we utilize the wire perturbation technique to further minimize the number of stitches required for layout decomposition.

Fig. 20(a) shows three features conflicting with each other and the corresponding feature conflict graph. Traditionally, the conflict can be solved by traversing the conflict graph, finding the odd cycle indicating a conflict, and then breaking the odd cycle by splitting an appropriate node, which is equivalent to inserting a stitch on the corresponding feature, as illustrated in Fig. 20(b). However, the wire perturbation technique provides an opportunity to resolve conflicts without inserting stitches. As shown in Fig. 20(c), the conflict formed by the three features is resolved by simply moving down the feature B. Thus, we reuse the wire perturbation framework proposed for NC minimization to reduce as many required stitches for layout decomposition as possible.

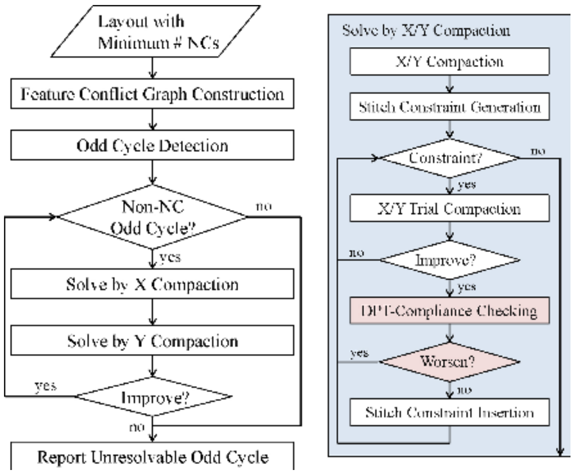


Fig. 21. Flow of stitch minimization with the wire perturbation technique.

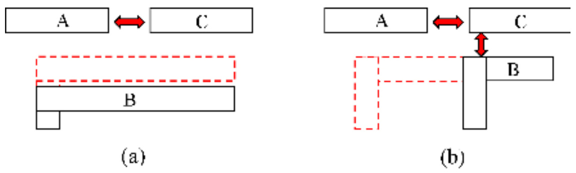


Fig. 22. Conflict elimination with the wire perturbation technique. (a) Resolving the conflict between A and B by moving the horizontal wire of B downward. (b) Resolving the conflict between A and B by moving the vertical wire of B rightward.

A. Stitch Minimization Flow

Fig. 21 shows the algorithm flow of stitch minimization with the wire perturbation technique, which is modified from the NC-aware wire perturbation algorithm flow proposed in Section IV. Instead of performing pattern projection and segmentation beforehand, we directly construct the feature conflict graph for a layout with the minimum number of NCs. After that, odd cycle detection is performed to find the locations where stitch insertion is needed. Note that we only target the odd cycles that are not NCs since the odd cycles of NCs have been shown that they cannot be resolved by this iterative compaction approach. For a non-NC odd cycle, we perturb its corresponding features by X/Y compaction. DPT-compliance checking is required for each successful odd cycle elimination because it is undesired that the number of stitches is reduced but the number of NCs increases.

B. Stitch Constraint Generation

For a detected odd cycle in a feature conflict graph, the feature pairs corresponding to the edges of the odd cycle are examined as described in Section IV-E1. We also keep a feature pair solution table to record the occurrences of feature pairs, and try to separate the features of a feature pair with the highest frequency of its occurrences. Instead of moving a whole feature as shown in Fig. 20(c), we may just adjust a part of a feature to resolve conflicts, which may make the wire perturbation technique more flexible. As shown in Fig. 22, to resolve the odd cycle formed by the features A, B, and C, we try to separate the feature B from the feature A. This can be achieved by moving the horizontal wire of the feature B downward or by shifting the vertical wire rightward, as illustrated in Fig. 22(a) and (b). In addition, successive movements of wires of a feature may increase the success rate of conflict elimination.

TABLE I
STATISTICS OF THE ISCAS-89 CIRCUITS

Design	Cells	Layers	Nets	Wires	Tiles	Conflict Edges	Points	Constraint Edges
s27	19	3	26	174	116	52	200	1158
s208_1	73	5	86	851	586	227	946	5946
s298	141	4	147	1623	1075	385	1783	11 555
s344	132	4	144	1695	1189	470	1849	12 069
s349	123	4	135	1466	937	269	1620	10 450
s382	159	6	165	1956	1273	375	2143	13 987
s386	167	4	177	1908	1398	523	2105	13 886
s400	163	5	169	1985	1520	669	2182	14 489
s420_1	187	5	208	2360	1493	461	2587	16 771
s444	162	5	168	1897	1376	579	2087	13 867
s510	256	5	278	3132	2563	1270	3437	23 494
s526	237	4	243	2729	2261	1065	2994	20 782
s526n	228	4	234	2822	2183	1026	3080	20 681
s641	193	6	231	2450	1621	449	2693	17 700
s713	192	6	230	2262	1546	527	2502	16 383
s820	351	6	372	4513	3393	1410	4919	33 497
s832	333	6	354	4512	3583	1669	4913	33 408
s838_1	367	5	404	4554	3431	1626	4994	33 888
s1196	483	5	500	7051	5473	2504	7608	52 357
s1238	586	5	603	7866	6745	3829	8528	59 982
s1423	607	4	627	7837	6031	2359	8549	61 190
s1488	636	6	647	9287	7847	4155	10 012	69 851
s1494	643	5	654	9240	7607	3786	9970	70 163
s5378	1294	6	1336	17361	12 455	4272	18 841	133 850
s9234_1	974	6	1016	12 414	8043	2109	13 539	94 713
s13207	1219	6	1313	16 214	10 196	2717	17 556	122 742
s15850	685	6	766	8277	4963	848	9057	62 671

Cells: number of standard cells. Layers: number of routing layers. Nets: number of nets. Wires: number of wires (each net consists of several wire segments). Tiles: number of tiles after segmentation. Conflict Edges: number of edges in the tile conflict graph. Points: number of points in our symbolic layout representation. Constraint Edges: number of edges in the point constraint graphs.

VI. Experimental Results

The proposed wire perturbation algorithm was implemented in the C++ programming language on a 2 GHz Linux machine with 8 GB memory. The ISCAS-89 circuits from the IWLS benchmark [14] are used to evaluate our algorithm. To obtain the layout, the circuits were scaled down for 45 nm technology, and were placed and routed by Cadence SoC Encounter. Then, the wire information was stored in a DEF file as the input to our perturbation system. In addition, for the circuits, the minimum width was equal to the minimum spacing. Thus, the DP-spacing was set accordingly in the layout. The overlap margin was set to be 20% of the minimum width.

Table I lists the statistics of the ISCAS-89 circuits. In this table, "Design" gives the names of the circuits, "Cells" denotes the number of standard cells, "Layers" denotes the number of routing layers used, and "Nets" denotes the total number of the nets. In the circuit, each net consists of several wire

TABLE II
RESULTS OF NC-AWARE WIRE PERTURBATION FOR THE ISCAS-89 CIRCUITS

Design	NC_o	NC_p	NC_%	S_o	S_p	S_%	CPU
s27	0	0	–	0	0	–	0
s208_1	0	0	–	1	0	100	1
s298	21	0	100	1	1	0	2
s344	10	3	70	1	1	0	11
s349	1	0	100	0	0	–	0
s382	23	0	100	2	1	50	2
s386	0	0	–	4	3	25	37
s400	3	0	100	2	1	50	8
s420_1	6	4	33	5	3	40	21
s444	9	0	100	1	0	100	2
s510	26	0	100	10	4	60	72
s526	9	2	78	11	7	36	71
s526n	25	0	100	14	7	50	65
s641	0	0	–	8	3	63	43
s713	2	0	100	1	1	0	1
s820	13	1	92	5	3	40	124
s832	65	5	92	19	10	43	493
s838_1	26	7	73	19	11	42	449
s1196	60	0	100	17	15	12	257
s1238	70	16	77	36	22	39	3912
s1423	56	3	95	25	18	28	898
s1488	88	45	49	41	33	20	4560
s1494	77	8	90	43	27	37	3812
s5378	69	14	80	35	30	14	17 782
s9234_1	24	2	92	18	12	33	5277
s13207	15	2	87	14	6	57	4933
s15850	4	2	50	8	5	38	1440
			85.51			39.24	

NC_o (NC_p): number of NCs in original (perturbed) layout. NC_%: reduction of NCs in terms of percentage. Iter.: number of iterations. S_o (S_p): number of stitches before (after) wire perturbation. S_%: reduction of stitches in terms of percentage. CPU: total running time (s).

segments, the total number is shown in "Wires." Note that in our perturbation system, each wire is regarded as the smallest unit to be moved. Therefore, the number of wires can be considered as the input size to our algorithm.

We also show the statistics about the problem size for our prediction method. "Tiles" and "Conflict Edges" give the total number of tiles after segmentation and the number of edges in the tile conflict graph. From the results, the number of tiles is linear to the number of wires, and thus the tile conflict graph is a sparse graph, respectively. Thus, finding the cycle basis can be done in $O(n)$ time, and hence the overall method for NC prediction is very efficient. For the problem size of our compaction system, "Points" and "Constraint Edges" in Table I are the number of points in our symbolic representation and the number of edges in the point constraint graph, respectively.

Table II lists the number of NCs and the number of stitches detected by our NC-prediction method for the original layout and the perturbed one. "NC" shows the number of NCs. "S" shows the number of stitches. "%" is the reduction in terms of percentage. The running time is shown in "CPU."

Experimental results have shown that the number of NCs and the number of required stitches can be significantly reduced by our algorithm. Although the order of x-directional and y-directional perturbation might cause the difference in solution quality, from the results, both the x–y iterative method and y–x iterative method significantly reduce the number of NCs by more than 85% and the number of stitches by more than 39%, on average. Therefore, with the perturbed layout, the decomposition success rate may be effectively increased for the next stage, and the yield loss caused by overlay errors may be reduced.

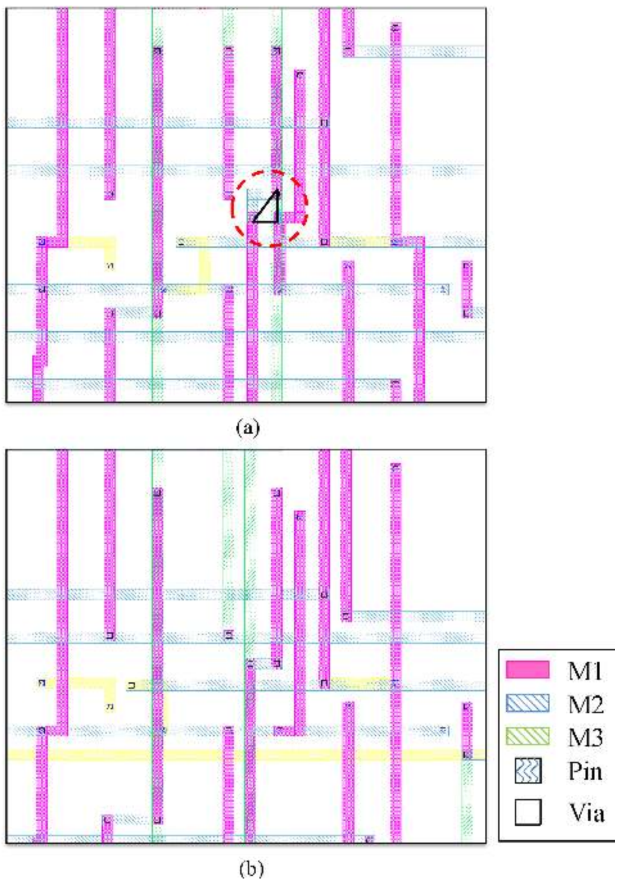


Fig. 23. Wire perturbation resolves the NC. (a) Native conflict in circuit s400 is detected and highlighted by our prediction method. (b) NC was resolved after wire perturbation.

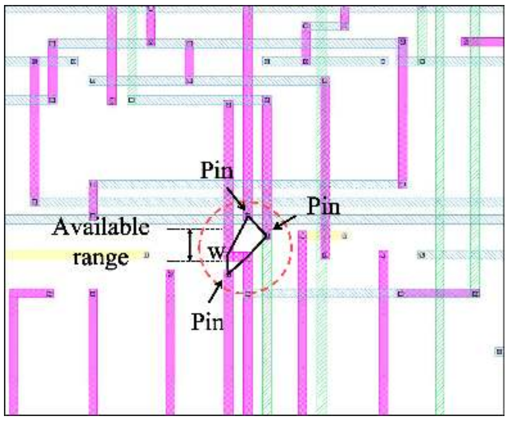


Fig. 24. Example of unresolvable NC from circuit s820 is highlighted in the center. Due to the limitations of neighboring pins, the wire w cannot have enough perturbation range. Thus, the conflict is unresolvable by wire perturbation.

Fig. 23 shows a local view for the perturbation result of circuit s1196. In the center, there is a NC caused by the line ends and the corner of wires. Obviously, the conflict cannot be resolved by any kind of stitch insertion. As shown in Fig. 23(b), after the perturbation, the corner part was moved downward by our algorithm and thus the odd cycle was resolved.

We found that some NCs are unresolvable by wire perturbation. For example, as shown in Fig. 24, the wire w in the center of figure conflicts with all the three neighboring wires at their endpoints. However, these three endpoints are all pins, which cannot be adjusted. Also, they limit the available perturbation range of w. Therefore, there is no solution to perturb the w or other three wires to resolve this NC. To conquer this problem, we need a more advanced technique, e.g., detour or rerouting.

VII. Conclusion

In this paper, we have presented an NC-prediction method based on geometry relation of features, and an NC and stitch-aware wire perturbation algorithm was proposed to minimize as many NCs and required stitches in a layout as possible. The proposed NC-aware wire perturbation algorithm could efficiently enhance the decomposability and effectively resolve NCs of a layout, which makes up the deficiencies of the algorithms based on ILP formulations proposed by some previous work. In addition, the extended wire perturbation technique can further reduce the number of stitches required for the decomposition process. Furthermore, since our wire perturbation system adopts an iterative 1-D compaction approach, it can easily be embedded into existing industrial compaction systems. Experimental results have shown that our algorithm can significantly reduce the number of NCs and the number of stitches required for layout decomposition.

Future work of NC removal and automatic DPT-compliant redesign includes the following.

1) Develop a wire perturbation algorithm with more advanced features, such as, layer assignment, detour, and simple rerouting. Thus, we can have more flexibility and capability to resolve the NCs which are originally unresolvable by moving wire segments alone.
2) Incorporate the DPT-compliant issue to other stages of the design flow, e.g., placement and routing. Thus, we can ensure the decomposability of a layout at an earlier stage.

References

[1] G. E. Bailey, A. Tritchkov, J.-W. Park, L. Hong, V. Wiaux, E. Hendrickx, S. Verhaegen, P. Xie, and J. Versluijs, "Double pattern EDA solutions for 32nm HP and beyond," *Proc. SPIE*, vol. 6521, p. 65211k, Mar. 2007.

[2] P. Berman, A. B. Kahng, D. Vidhani, H. Wang, and A. Zelikovsky, "Optimal phase conflict removal for layout of dark field alternating phase shifting masks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 2, pp. 175–187, Feb. 2000.

[3] S.-Y. Chen and Y.-W. Chang, "Native-conflict-aware wire perturbation for double patterning technology," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2010, pp. 556–561.

[4] T.-B. Chiou, R. Socha, H. Chen, L. Chen, S. Hsu, P. Nikolsky, A. van Oosten, and A. C. Chen, "Development of layout split algorithms and printability evaluation for double patterning technology," *Proc. SPIE*, vol. 6924, p. 69243M, Mar. 2008.

[5] M. Cho, Y. Ban, and D. Z. Pan, "Double patterning technology friendly detailed routing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2008, pp. 506–511.

[6] N. Deo, G. M. Prabhu, and M. S. Krishnamoorthy, "Algorithms for generating fundamental cycles in a graph," *ACM Trans. Math. Softw.*, vol. 8, no. 1, pp. 26–42, Mar. 1982.

[7] R. Diestel, *Graph Theory*, 2nd ed. Berlin, Germany: Springer-Verlag, 2000.

[8] J. Doenhardt and T. Lengauer, "Algorithmic aspects of one-dimensional layout compaction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 6, no. 5, pp. 863–878, Sep. 1987.

[9] M. Drapeau, V. Wiaux, E. Hendrickx, S. Verhaegen, and T. Machida, "Double patterning design split implementation and validation for the 32nm node," *Proc. SPIE*, vol. 6521, p. 652109, Feb. 2007.

[10] S. H. Gerez, *Algorithms for VLSI Design Automation*. New York: Wiley, 1999.

[11] J. D. Horton, "A polynomial-time algorithm to find the shortest cycle basis of a graph," *SIAM J. Comput.*, vol. 16, no. 2, pp. 358–366, Apr. 1987.

[12] C.-H. Hsu, Y.-W. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2009, pp. 595–600.

[13] C.-H. Hsu, Y.-W. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 9, pp. 284–294, Feb. 2011.

[14] *IWLS 2005 Benchmark* [Online]. Available: http://www.iwls.org/iwls2005/benchmarks.html

[15] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition approaches for double patterning lithography," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 6, pp. 939–952, Jun. 2010.

[16] D. Laidler, P. Leray, K. Dhave, and S. Cheng, "Sources of overlay error in double patterning integration schemes," *Proc. SPIE*, vol. 6922, p. 69221E, Feb. 2008.

[17] J. Park, S. Hsu, D. V. D. Broeke, J. F. Chen, M. Dusa, R. Socha, J. Finders, B. Vleeming, A. van Oosten, P. Nikolsky, V. Wiaux, E. Hendrickx, J. Bekaert, and G. Vandenberghe, "Application challenges with double patterning technology (DPT) beyond 45 nm," *Proc. SPIE*, vol. 6349, p. 634922, Sep. 2006.

[18] A. van Oosten, P. Nikolsky, J. Huckabay, R. Goossens, and R. Naber, "Pattern split rules! A feasibility study of rule based pitch decomposition for double patterning," *Proc. SPIE*, vol. 6730, p. 67301L, Sep. 2007.

[19] R. Tarjan, "Depth-first search and linear graph algorithm," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, Jun. 1972.

[20] A. K.-K. Wong, *Resolution Enhancement Techniques in Optical Lithography*. Bellingham, WA: SPIE, Mar. 2001.

[21] Y. Xu and C. Chu, "A matching based decomposer for double patterning lithography," in *Proc. ACM/IEEE Int. Symp. Phys. Des.*, Mar. 2010, pp. 121–126.

[22] J. S. Yang, K. Lu, M. Cho, K. Yuan, and D. Z. Pan, "A new graph-theoretic, multi-objective layout decomposition framework for double patterning lithography," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2010, pp. 637–644.

[23] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 2, pp. 185–196, Feb. 2010.

[24] K. Yuan and D. Z. Pan, "WISDOM: Wire spreading enhanced decomposition of masks in double patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2010, pp. 32–38.

Shao-Yun Fang received the B.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2008. She is currently pursuing the Ph.D. degree with the Graduate Institute of Electronics Engineering, National Taiwan University.

Her current research interests include physical design and design for manufacturability.

Szu-Yu Chen received the B.S. and M.S. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 2007 and 2009, respectively.

His current research interests include physical design and design for manufacturability.

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

He is currently the Director of the Graduate Institute of Electronics Engineering and a Distinguished Professor with the Department of Electrical Engineering, NTU. His current research interests include very large-scale integrated physical design and design for manufacturability. He has been working closely with the industry in these areas. He has co-edited one textbook on electronic design automation (EDA) and co-authored one book on routing, and published over 190 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a first-place winner of the 2011 PATMOS Timing Analysis Contest. He was also a four-time winner of the ACM ISPD Contests (placement in 2006, global routing in 2008, clock network synthesis in 2009 and 2010), a recipient of six Best Paper Awards (ICCD, etc.), and was nominated for 20 Best Paper Awards from DAC (five times), ICCAD (four times), etc., in the past ten years. He has received many research and teaching awards, such as the Distinguished Research Award from the National Science Council of Taiwan (twice), the IBM Faculty Award, the CIEE Distinguished EE Professor, the MXIC Young Chair Professorship, and the Excellent Teaching Award from NTU (seven times). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and an editor of the *Journal of Information Science and Engineering*. He has served as the General Chair and the Steering Committee Chair of ISPD, and the Program Chair of ASP-DAC, FPT, and ISPD. He was on the ICCAD Executive Committee (Vice Technical Program Committee Chair), the ASP-DAC Steering Committee, the IEEE CEDA Executive Committee, the ACM/SIGDA Physical Design Technical Committee, and the Technical Program Committee of ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SLIP, SOCC, and VLSI-DAT. He has served as an Independent Board Director of Genesys Logic, Inc., Taipei, a technical consultant of Faraday Technology Corporation, MediaTek, Inc., Hsinchu, Taiwan, and RealTek Semiconductor Corporation, Hsinchu, the Chair of the EDA Consortium of the Ministry of Education, Taiwan, and a member of the Board Governors of Taiwan IC Design Society.