Double Patterning Layout Decomposition for Simultaneous Conflict and Stitch Minimization

Kun Yuan, Jae-Seok Yang, and David Z. Pan, *Senior Member, IEEE*

***Abstract*—Double patterning lithography (DPL) is considered as a most likely solution for 32 nm/22 nm technology. In DPL, the layout patterns are decomposed into two masks (colors), and manufactured through two exposures and etch steps. If the spacing between two features (polygons) is less than certain minimum coloring distance, they have to be assigned opposite colors. However, a proper coloring is not always feasible because two neighboring patterns within the minimum distance may be in the same mask due to complex pattern configurations. In that case, a feature may need to be split into two parts to resolve the conflict, resulting in stitch insertion which causes yield loss due to overlay and line-end effect. While previous layout decomposition approaches perform coloring and splitting separately, in this paper, we propose a simultaneous conflict and stitch minimization algorithm with an integer linear programming (ILP) formulation. Since ILP is in class NP-hard, the algorithm includes three speed-up techniques: 1) grid merging; 2) independent component computation; and 3) layout partition. In addition, our algorithm can be extended to handle design rules such as overlap margin and minimum width for practical use as well as off-grid layout. Our approach can reduce 33% of stitches and remove conflicts by 87.6% compared with two phase greedy decomposition.**

***Index Terms*—Double patterning lithography, integer linear programming, layout decomposition.**

I. Introduction

As the minimum feature size decreases, semiconductor industry is facing the limitation of patterning sub-32 nm due to the delay of the next generation lithography equipment such as extreme ultraviolet [1]. Double patterning lithography (DPL) [2]–[5] emerges almost the only alternative for 32 nm/22 nm nodes and it is already used for NAND-flash production. In DPL, a single layout is decomposed into two masks and manufactured through two exposure/etching steps. As a benefit, the pitch size is doubled, which enhances the resolution as illustrated in Fig. 1. Although DPL requires two masks and increases the design cost, it is widely considered as a most likely solution for 32 nm, 22 nm, and even 16 nm.

Double patterning layout decomposition [6]–[8] is a process that assigns two features within the given minimum space to different masks. A layout may contain a pattern which is unable to assign a color. In this case, a feature may be split into two parts and colored differently to resolve the conflict, which generates stitches. Stitches will cause yield loss and increase manufacturing cost due to overlay errors, which is 5 nm or 6 nm under current 32 nm double patterning lithography. Some mask misalignment direction [4] could be actually beneficial for printability. However, on the presence of various process uncertainties, such as dose, focus, and mask errors, the printed stitch width could be easily smaller than 25 nm and result in design failure. Pushing overlay below 3 nm [9] is very challenging. Moreover, the additional line-ends may cause more pattern degradation and reduce yield in case of defocus and dose variation. After splitting, a few unresolved or even unresolvable conflicts may remain and will be corrected by time consuming layout redesign. Therefore, it is important to produce high quality decomposition solution with less conflicts and stitches.

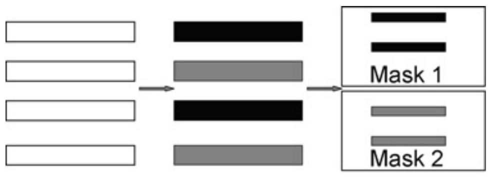


Fig. 1. One single design is decomposed into two masks and the pitch size is increased effectively in DPL.

There are a few works focusing on stand-alone layout decomposition. A heuristic approach is proposed in [7] to cut troublesome patterns after finding the coloring conflicts. The patterns are prefragmented into smaller pieces in [8] to perform coloring. All these works do not have a systematical way to minimize the number of conflicts and stitches. Coloring and splitting are considered in separate steps while they are highly correlated tasks. Pattern matching technique is proposed in [10] to decompose the layout. However, it might not be able to work on large scale problem, hence limits the solution quality. Recently, a practical layout decomposition flow is proposed in [11] to address design needs for double patterning. They first detect the features associated with unresolvable conflict cycles for layout modification. The remaining design is then decomposed to minimize the number of stitches based on an ILP formulation. However, in their work, the number of unresolvable conflict cycles and splitting stitches are not optimized together, and conflict elimination technique is quite greedy.

Manuscript received June 9, 2009; revised August 17, 2009. Current version published January 22, 2010. This paper was supported in part by the National Science Foundation, Semiconductor Research Corporation, Sun, Qualcomm, and equipment donations from Intel. This paper was recommended by Associate Editor P. Saxena.

The authors are with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78731 USA (e-mail: kyuan@cerc.utexas.edu; jsyang@cerc.utexas.edu; dpan@cerc.utexas.edu).

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

Digital Object Identifier 10.1109/TCAD.2009.2035577

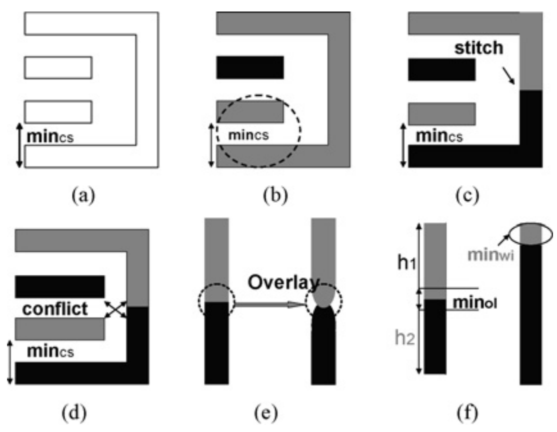


Fig. 2. Concept of conflict and stitch.

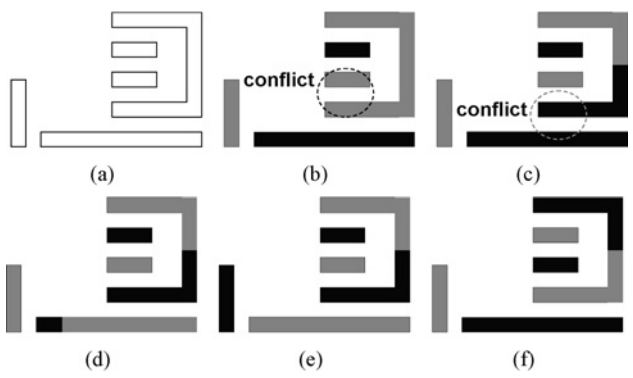


Fig. 3. Shortcoming of two phase layout decomposition flow in previous works [7], [8]. An unplanned coloring will need much extra effort during splitting.

In this paper, we propose an algorithm to decompose layout for minimizing conflicts and stitches simultaneously. The proposed approach reduces the conflicts by 87.6% with 33% less stitches compared to a greedy two phase decomposition flow. When compared to a methodology based on [11], we are also able to achieve averagely 87.2% and 10% reduction on conflicts and stitches, respectively. Although our approach is comparatively slower, we can obtain coloring solutions for all the test cases within a few minutes. The runtime shows linear complexity with respect to problem size.

Our main contributions are as follows.

1) We propose a new grid model to enable bigger solution space than previous works [7], [8] and perform simultaneous conflict and stitch optimization.
2) We develop an ILP algorithm to minimize the number of conflicts and stitches for a high quality solution.
3) We propose three speed-up techniques (grid merging, independent component computing and layout partition) to improve the runtime and scalability of our algorithm. For layout partition, we identify and solve a coloring flip optimization problem to minimize the conflicts and stitches across the boundary of different partitions.
4) We discuss how to extend our proposed grid model to handle various splitting rules and design patterns in practice.

The rest of the paper is organized as follows. Section II provides the preliminaries and motivates. In Section III, we discuss the problem formulation with related model and definitions. The basic ILP formulation is described in Section IV with three speed-up techniques. The extensive discussion on grid model for practical design issues is presented in Section V. Section VI presents the experiment results and Section VII concludes this paper.

II. Preliminaries and Motivation

A. Double Patterning Layout Decomposition Considerations

As explained in Section I, in DPL, the original design will be assigned into two masks. There are two critical issues with this layout decomposition: coloring conflict and splitting stitch.

1) *Coloring Conflict:* If the distance between two separate features is less than minimum coloring spacing $\min_{cs}$, they should be assigned to different masks (colors). Otherwise, there will be a coloring conflict.

Fig. 2(a) shows a layout with three features, and any two of them are required to have different colors because of the insufficient spacing. A coloring conflict will be unavoidable as in Fig. 2(b). Sometimes, such a violation can be eliminated by appropriately splitting the features like Fig. 2(c). There are also unresolvable conflicts, as Fig. 2(d) indicates, which requires modifying the design.

2) *Splitting Stitch:* The stitch exists when two *touched* features are assigned to different masks. The stitch can be inserted to split some features to resolve the conflict as shown in Fig. 2(c). However, stitch insertion can have negative effects on yield due to overlay error between the two masks as Fig. 2(e) illustrates. In addition, the line-end will cause pattern degradation.

There are several practical guidelines for splitting. As Fig. 2(f) shows, in order to control the overlay, there is a minimum overlap length, $\min_{ol}$, requirement for stitch insertion. The segments h_1 and h_2 on different masks should be overlapped to certain amount ensuring better manufacturability. Moreover, we do not want to have any minimum width, $\min_{wi}$, rule violation during splitting, as marked by the circle in Fig. 2(f).

Without altering layout in the scope, the general objective of layout decomposition can be stated as minimizing the unresolved conflicts by introducing as few as possible stitches.

B. Simultaneous Optimization

The previous works insert stitches after coloring to resolve conflicts. Without planning possible splitting during coloring, it is hard to eliminate the conflict. Considering a layout in Fig. 3(a), we have a coloring solution in Fig. 3(b). During the splitting, the U feature should be cut into two parts to remove the conflict but we have to further check whether the splitting will result in another conflict like Fig. 3(c). In such case, the coloring of the neighborhood features needs to be reconsidered to avoid unnecessary stitches like Fig. 3(d) and enable optimal solution in Fig. 3(e) or (f). This is a simple example, but as we can see, given the complexity of modern design, the two-phase approach will have extreme difficulty handling the exploding consideration and producing

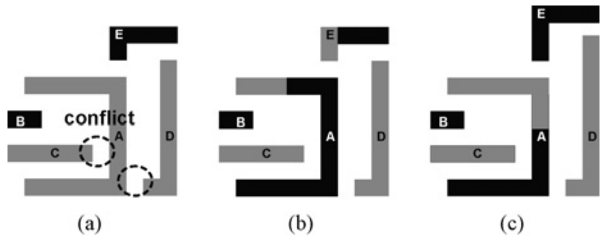


Fig. 4. Different stitch candidates can lead to different solution qualities.

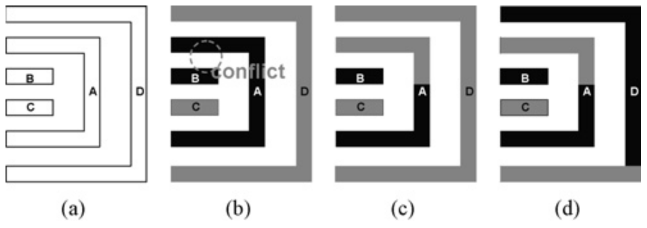


Fig. 5. Difficulty of predicting where the splitting is needed.

high quality solution. This motivates us simultaneous conflict and stitch minimization during layout decomposition.

III. Problem Formulation

In this section, we will first motivate and introduce our grid model in Section III-A. The basic terms will be defined in the following Section III-B. The formal problem definition will be described in the end.

A. Grid Layout Model

Considering splitting during coloring is a challenging problem. First of all, the stitch configurations are highly correlated and all the potential locations need to be considered for global optimality. Fig. 4(a) is a case with two conflicts. As we can see, two possible splitting choices on feature A lead to two different solutions, Fig. 4(b) and (c). The first one has two stitches, where the latter one associates with only one. Moreover, we can even hardly predict where we could have a splitting due to some *chain* effect. For example, the right most feature D is not expected to be cut in Fig. 5(a) because it is only adjacent to one single feature A. However, given a coloring assignment as shown in Fig. 5(b), feature A will be split to resolve the conflict between A and B like Fig. 5(c). As a result, feature D also needs to be broken into two segments as shown in Fig. 5(d).

In order to overcome these issues, we will map the whole layout into grids with its size to be half the pitch of the original design. Each grid is either empty or fully occupied by the pattern, and each occupied grid will be assigned one color. Therefore, any boundary between grids is a potential splitting location. This is shown in the Fig. 6. Essentially, we provide fine resolution for splitting options. This model is able to offer sufficient stitch candidates for all the features across the design in practice and the solution space is much bigger than previous works [7], [8]. The discretization is reasonable because a design usually follows underlying regular pitches

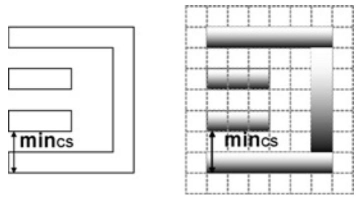


Fig. 6. Proposed grid layout model.

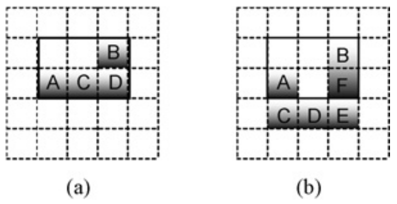


Fig. 7. Concept of blocking path. The solid rectangle marks the bounding box.

in modern layout. Minimum coloring spacing $\min_{cs}$ is taken as two-grid size to double the spacing for each mask in this paper and also subject to change according to given $\min_{cs}$.

B. Terms and Problem Formulation

Before formulating our problem, we will first define the terms in the grid layout model.

Definition 1 (Occupied Grid, OG): The grid filled by the layout.

The OG must be assigned one of the two colors: gray and black.

Definition 2 (Blocking Path, BP): Given two occupied grids OG_1 and OG_2, a blocking path is a path when:

1) it is fully composed of OGs and connects OG_1 and OG_2;
2) OG_1 and OG_2 are touching its two ending grids, respectively, but not belonging to this path;
3) this path is within the bounding box of OG_1 and OG_2.

The main usage of blocking path is to identify neighboring but locally isolated layout grids. These grids, even belonging to the same connection, need to be considered as different features, and could form a coloring conflict.

As shown in Fig. 7(a), C–D is a blocking path for grid A and B. In another example Fig. 7(b), C–F is not a BP for A–B, because not all of them are in the bounding box of A–B as the third rule defines. Some part of it (C–E) is beyond the box, and hence locally A–B can be considered as isolated.

Definition 3 (Potential Conflict Grid Pair, PCGP, and Potential Stitch Grid Pair, PSGP): Given two occupied grids OG_1 and OG_2.

1) If the distance between OG_1 and OG_2 is less than $\min_{cs}$ and the two grids are not touching, they form a potential conflict grid pair.
2) If OG_1 and OG_2 are touching, they form a potential stitch grid pair.

The distance between a pair of OGs is the minimum distance between any two points from the OGs. For example

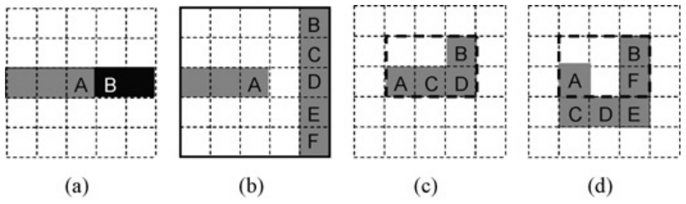


Fig. 8. Stitch grid pair and conflict grid pair. Dashed box in (c) and (d) is the bounding box of A and B.

in Fig. 7(b), the distance for untouched B and C is $\sqrt{2}$ grid size due to two closest corners, which is smaller than $\min_{cs}$. Therefore, they form a PCGP.

Definition 4 (Stitch Grid Pair, SGP): If the grids of a PSGP are assigned different colors, it is a stitch grid pair.

Definition 5 (Conflict Grid Pair, CGP): If a PCGP is in the identical color, and there is no blocking path connecting them in the same mask, it is a conflict grid pair.

The definition of SGP is straightforward as grids A and B shown in Fig. 8(a). Fig. 8(b) shows the normal CGP cases, where a PCGP is colored identically and unconnected. B–F and A are within the minimum coloring spacing. There are even no paths connecting them, not to mention blocking path. The rule one of Definition 2 is violated. As a result, any of B–F and A are a CGP.

There are also some special CGP cases that we need to further consider blocking path in order to avoid false recognition of lithography friendly pattern. If two nontouching grids are electrically connected through a blocking path, we should not consider them belonging to different features. The printability will not be an issue. As shown in Fig. 8(c), grid A and B have a BP C–D in the same mask between them, so they do not form a CGP. It is indeed a normal jog, and can be printed well. In contrast, although there is a path C–F connecting A and B in Fig. 8(d), C–E is out of their bounding box. In consequence, the path is not a blocking path. This violates the third rule of Definition 2, so grid A and B form a CGP. In this case, A and B are in fact locally isolated but neighboring within the bounding box. This configuration is a typical U shape pattern, and would have weak printability.

C. Problem Description

In our work, we use the number of SGPs and CGPs as the cost, which assigns higher weight to the grids that are associated with more conflicts/stitches. Formally, we formulate the layout decomposition optimization problem as follows:

Problem Formulation: Given a grid layout, color it into two parts (gray and black). The primary objective is to minimize the number of CGPs and the second objective is to minimize the number of SGPs.

We prefer a solution with less CGPs than one with smaller number of SGPs but more CGPs, because a layout with nonzero CGPs is essentially not manufacturable and a solution with less CGPs reduces expensive redesign effort.

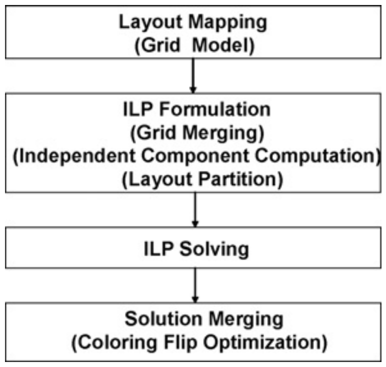


Fig. 9. Overall layout decomposition flow.

TABLE I
NOTATION FOR BASIC ILP FORMULATION

$og_{i,j}$	Occupied grid of which i and j are coordinates.
$x_{i,j}$	Binary variable that denotes the color of $og_{i,j}$. $x_{i,j}=1$ if the color is gray, otherwise, it is black.
$s_{ij,mn}$	Binary variable $s_{ij,mn}=1$ if $og_{i,j}$ and $og_{m,n}$ is a SGP.
$c_{pq,uv}$	Binary variable $c_{pq,uv}=1$ if $og_{p,q}$ and $og_{u,v}$ is a CGP.
SP	Set of PSGPs.
CP	Set of PCGPs.
$P_{pq,uv}$	Set of BPs connecting $og_{p,q}$ and $og_{u,v}$.
$p^k_{pq,uv}$	k_{th} BP connecting $og_{p,q}$ and $og_{u,v}$.
$n^k_{pq,uv}$	Number of grids in $p^k_{pq,uv}$.
$g^k_{pq,uv}$	Binary variable $g_{pq,uv}=1$ if $p^k_{pq,uv}$ is a gray BP.
$b^k_{pq,uv}$	Binary variable $b_{pq,uv}=1$ if $p^k_{pq,uv}$ is a black BP.

IV. ALGORITHM

In this section, we will present our ILP-based layout decomposition algorithm. The entire flow is shown in Fig. 9. After mapping the design to grid model, we will process the grids and formulate the basic ILP formulation. Since the timing complexity for ILP is very high, we will then propose three speed-up techniques by either eliminating unnecessary variables or dividing the whole problem into several smaller ones. Finally, the layout decomposition for the entire design can be obtained by merging the subproblem solutions. For better solution reunion, we formulate a problem of coloring flipping optimization through ILP.

A. Basic ILP Formulation

To better present our method, we first describe the notation in Table I. The simultaneous coloring and splitting optimization can be formulated as follows:

$$\min \left(\sum_{s_{ij,mn} \in SP} s_{ij,mn} + \alpha \sum_{c_{pq,uv} \in CP} c_{pq,uv} \right) \tag{1}$$

subject to

$$x_{i,j} + (1 - x_{m,n}) \leq 1 + s_{ij,mn} \quad \forall s_{ij,mn} \in SP \tag{2}$$

$$(1 - x_{i,j}) + x_{m,n} \leq 1 + s_{ij,mn} \quad \forall s_{ij,mn} \in SP \tag{3}$$

$$\sum_{x_{e,f} \in p^k_{pq,uv}} x_{e,f} \le (n^k_{pq,uv} - 1) + g^k_{pq,uv} \quad \forall p^k_{pq,uv} \in P_{pq,uv} \tag{4}$$

$$\sum_{x_{e,f} \in p^k_{pq,uv}} (1 - x_{e,f}) \le n^k_{pq,uv}(1 - g^k_{pq,uv}) \quad \forall p^k_{pq,uv} \in P_{pq,uv} \tag{5}$$

$$\sum_{x_{e,f} \in p^k_{pq,uv}} (1 - x_{e,f}) \le (n^k_{pq,uv} - 1) + b^k_{pq,uv} \quad \forall p^k_{pq,uv} \in P_{pq,uv} \tag{6}$$

$$\sum_{x_{e,f} \in p^k_{pq,uv}} x_{e,f} \le n^k_{pq,uv}(1 - b^k_{pq,uv}) \quad \forall p^k_{pq,uv} \in P_{pq,uv} \tag{7}$$

$$x_{p,q} + x_{u,v} \le 1 + c_{pq,uv} + \sum_k g^k_{pq,uv} \quad \forall c_{pq,uv} \in CP \tag{8}$$

$$(1 - x_{p,q}) + (1 - x_{u,v}) \le 1 + c_{pq,uv} + \sum_k b^k_{pq,uv} \quad \forall c_{pq,uv} \in CP. \tag{9}$$

The objective function (1) is to minimize the weighted summation of SGPs and CGPs. Parameter α is used to tune the relative importance between SGP and CGP, and can be set to ensure the priority of CGP elimination. All the PCGPs and PSGPs are predetermined by examining the neighboring grids for each OG.

Constraints (2) and (3) are used to identify SGP from PSGP. According to the definition of SGP, we need to know whether the PSGP grids have opposite colors. Whenever $x_{i,j}$ and $x_{m,n}$ have opposite values, the left hand side of one of the constraints will be two. As a result, $s_{ij,mn}$ must be assigned one to satisfy the constraints, which detects a SGP.

The usage of Constraints (4)–(9) is to determine whether a PCGP forms a CGP. Identifying CGP takes more effort. Besides checking the colors of PCGP, we need to know whether there is a blocking path in the same mask. All the possible BPs $P_{pq,uv}$ can be easily enumerated by depth first search on the occupied grids within the bounding box. We can investigate their coloring using Constraints (4)–(7). The corresponding binary variable $g^k_{pq,uv}/b^k_{pq,uv}$ will be true only if the grids of some blocking path are in the same mask. Constraints (8) and (9) evaluate the conditions for CGP. A conflict will be reported only if PCGP grids are assigned same color and the possible BPs $g^k_{pq,uv}/b^k_{pq,uv}$ do not exist.

Let n_{og} be the number of occupied grids, the basic formulation contains at most $O(n_{og})$ variables. The constraints are specified for detecting either PSGPs or PCGPs. Suppose there are n_{sp} PSGPs and n_{cp} PCGPs, the complexity of n_{sp} is $O(n_{og})$. n_{cp} is linearly related to n_{og}, but quadratically proportional to $\min_{cs}$. The complexity of constraints due to PSGPs is $O(n_{sp})$. The constraint number for PCGPs is linear proportional to n_{cp}. It is also exponentially related to $\min_{cs}$, which results from the enumeration of blocking paths. Although this formulation shows exponential complexity in terms of $\min_{cs}$, when we fix the value of $\min_{cs}$ as the presetting for layout decomposition, the number of variables and constraints is quadratic with respect to n_{og}.

The proposed integer linear formulation can minimize the number of conflicts and stitches simultaneously. However, because ILP is nondeterministic polynomial time-complete, it is not affordable to directly apply a basic ILP formulation for large modern designs.

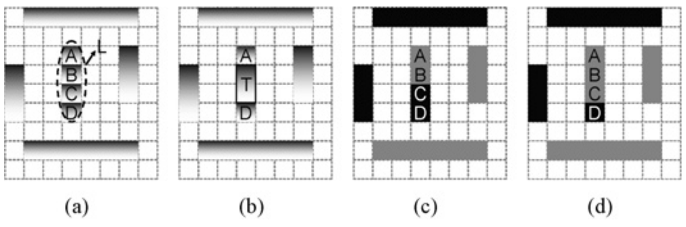


Fig. 10. Main idea of grid merging.

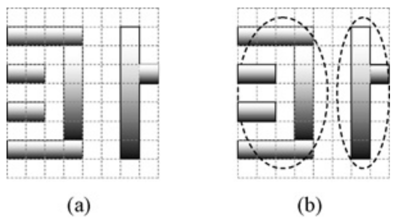


Fig. 11. Example of breaking big layout into two independent components, having no interacted PSGPs/PCGPs and marked by the dashed circle.

B. Speed-Up Techniques

In this section, we will discuss three speed-up techniques. The clustering methodology is applied in grid merging to reduce the number of variables and constraints. In contrast, the key idea of the other two techniques is to use a divide and conquer algorithm to convert the problem into smaller subproblems.

1) *Grid Merging:* In the proposed grid model, we aim to provide very fine resolution for stitch candidates. This may be over skilled under certain situations.

Consider the layout segment L in Fig. 10(a) with unit grids A–B–C–D. Only the two ending grids A and D may have coloring interaction with other layout objects besides L. B and C can be considered as *isolated* to some extent, because there are no occupied grids outside A–B–C–D which are touching them or within $\min_{cs}$ of their boundary. Therefore, it is not possible for B or C to form a stitch or conflict with other layout apart from the grids of segment L.

We can make advantage of above property to reduce problem size by combining this type of connected grids into a big super grid. As graphically shown in Fig. 10(b), B and C can be treated as a united grid T. This is equivalent to enforce B and C the same color. It will not deteriorate the conflict and stitch optimization. For this super grid, it does not have any chance to form a conflict or stitch with surrounding grids other than its two adjacent grids A and D.

Generally speaking, the elimination of internal splitting candidates is not a problem for solution quality. For any optimized solution obtained under original grid model with internal stitches, it can be mapped to one solution in the merged model with the stitch propagated to its ending grids, such as from (c) to (d) in Fig. 10.

2) *Independent Component Computation:* We propose independent component computation for reducing the ILP problem size without losing optimality. In real layout, we observe many *isolated* occupied grid clusters, i.e., there are no PSGPs or PCGPs formed between them. Therefore, we can

Algorithm 1 Independent Components Finding

```
Require: The grid layout
Ensure: The independent components, having no PSGPs/PCGPs
    between any pair of components
1: Build a graph G(V, E), $V \in \phi$, $E \in \phi$.
2: for each OG $og_{i,j}$ do
3:     Create one graph node $v_{i,j}$.
4: end for
5: for each PSGP/PCGP ($og_{i,j}$, $og_{m,n}$) do
6:     Create one edge between $v_{i,j}$ and $v_{m,n}$.
7: end for
8: Perform the depth first search on the graph G to find the
    independent components.
9: Map the graph nodes in each component back to OGs $og_{i,j}$ and
    return.
```

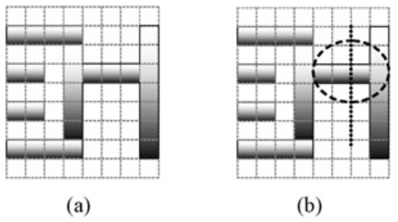


Fig. 12. Example of layout partition. The dotted line cuts the layout into two parts while the dashed circle marks PCGP and PSGP locations across the boundary of the two partitions.

break down the whole design into several independent components as shown in Fig. 11, and apply a basic ILP formulation for each one. The overall solution can be taken as the union of all the components without affecting the global optimality. The runtime of ILP formulation scales down dramatically with the reduction of the variables and constraints.

Our independent component finding algorithm is given in Algorithm 1. The timing complexity of this algorithm is $O(V+E)$, which V is the total number of the OGs and E is the total number of PSGPs and PCGPs.

3) *Layout Partition:* Some component may still have prohibitive problem size even after independent component computation. Our heuristic is to divide a big component into several small connected partitions and perform an ILP approach for each one, indicated in Fig. 12. Different from the independent component computation, there will be some PSGPs/PCGPs between different partitions. Although we solve each partition by ILP, the united solution does not guarantee to be optimal for the whole component in terms of ILP objective since the partition boundaries are not considered in the optimization.

In order to minimize the loss of global optimality, we need to partition the circuit with as few as possible cuts while ensuring that each partition can be efficiently solved by ILP. Balanced min-cut partition method is applied in our work. We first construct a graph G which is the same as in independent component computing. For each vertex (OG), we assign a weight as its edge degree plus one, taking into account the number of both variables and constraints it associates with. A threshold W_t is predefined for the maximum node weight summation we allow for each partition. The number of partitions can be calculated as $\lceil \frac{W}{W_t} \rceil$, where W is the total

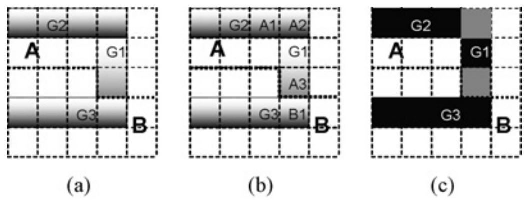


Fig. 13. "Internal" and "external" concepts. The wide solid line is the boundary of different partitions.

vertex weight of G. Suppose W is 10 000 and W_t is 3000, the component will be partitioned into four parts.

C. Solution Merging

After solving the solution for each component/partition, we need to merge the coloring assignment as a whole. While it is trivial to combine the solutions for smaller independent components, there comes a *coloring flip optimization* problem when we try to merge the solutions of all the partitions for the bigger components with partitioning applied.

In layout partition, the PSGPs and PCGPs for each partition can be divided into two disjoint subsets: the internal stitch/conflict grid pairs $PSGP^i$s/$PCGP^i$s, and external ones $PSGP^e$s/$PCGP^e$s. If the associated grids, which are needed for identifying whether a $PSGP/PCGP$ is a SGP/CGP, are all within the same partition, this $PSGP/PCGP$ belongs to $PSGP^i$s/$PCGP^i$s, otherwise, it is considered as a $PSGP^e/PCGP^e$. Similarly, during the unitization, the SGPs and CGPs for each partition can be categorized as SGP^is/CGP^is and SGP^es/CGP^es. SGP^is/CGP^is are from $PSGP^i$s/$PCGP^i$s, and SGP^es/CGP^es are from $PSGP^e$s/$PCGP^e$s.

As illustrated in Fig. 13(a), there are two partitions A and B. Suppose we are considering two $PCGP$s, (G_1, G_2) and (G_1, G_3), (A_1, A_2) and (A_3, B_1) are their additional associated grids, respectively, for correctly identifying a CGP, indicated by Fig. 13(b). (G_1, G_2) is a $PCGP^i$ because the grids which are related to (G_1, G_2, A_1, A_2) are all in partition A. In contrast, (G_1, G_3) is a $PCGP^e$ while (G_1, A_3) belongs to partition A and (G_3, B_1) is in partition B. Similarly, in one possible coloring configuration in Fig. 13(c), (G_1, G_2) is a CGP^i and (G_1, G_3) is a CGP^e.

During the solution union, it is possible to reduce the number of SGP^es/CGP^es by flipping the coloring of some partition. More importantly, such flipping will not change the status of SGP^is/CGP^is. In detail, it will not introduce new SGP^is/CGP^is, and any existing SGP^i/CGP^i will not go away as well. Based on the above definition, the related grids for identifying a SGP^i/CGP^i are in a single partition. Their coloring will be either flipped or not simultaneously. The conclusion of whether the respective $PSGP^i/PCGP^i$ is a SGP^i/CGP^i will not be changed.

The effect of coloring optimization is illustrated in Fig. 14, which has three partitions. The coloring merging in Fig. 14(a) produces one SGP and one CGP across the boundaries. If we flip the coloring of partition C from the black to gray, it becomes a SGP/CGP free assignment in Fig. 14(b). To optimize

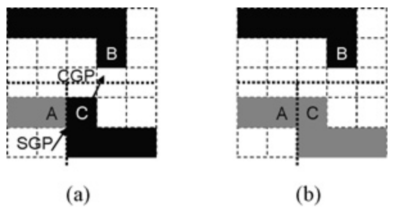


Fig. 14. Different coloring flips have distinct numbers of SGPs/CGPs across the boundaries, marked by the dotted lines.

TABLE II
NOTATION FOR COLORING FLIPPING PROBLEM

f_i	Binary variable $f_i = 1$ if partition i flips its coloring.
$f^0_{i,j}$	Binary variable $f^0_{i,j} = 1$ if both partitions flip or do not flip the coloring.
$f^1_{i,j}$	Binary variable $f^1_{i,j} = 1$ if only one partition between i and j flips its coloring.
$s^{e0}_{i,j}$	Number of stitches between partition i and j if both flip or do not flip the coloring.
$c^{e0}_{i,j}$	Number of conflicts between partition i and j if both flip or do not flip the coloring.
$s^{e1}_{i,j}$	Number of stitches between partition i and j if only one partition flips its coloring.
$c^{e1}_{i,j}$	Number of conflicts between partition i and j if only one partition flips its coloring.

the flipping scheme, we define *coloring flip optimization* as follows.

Coloring Flip Optimization: Given a number of partitions and their coloring solutions for one independent component, choose the best flipping scheme to minimize total cost of SGP^e and CGP^e, which cross the boundaries among all the partitions.

Because the number of partitions is usually not large, we also use an ILP formulation to solve this problem. The relevant notation can be found in Table II.

The formulation is as follows:

$$\min \sum (f^0_{i,j}(s^{e0}_{i,j} + \alpha c^{e0}_{i,j}) + f^1_{i,j}(s^{e1}_{i,j} + \alpha c^{e1}_{i,j})) \quad \forall i, j \tag{10}$$

subject to

$$f_i + f_j \leq 1 + f^0_{i,j} \tag{11}$$

$$(1 - f_i) + (1 - f_j) \leq 1 + f^0_{i,j} \tag{12}$$

$$f_i + (1 - f_j) \leq 1 + f^1_{i,j} \tag{13}$$

$$f_j + (1 - f_i) \leq 1 + f^1_{i,j} \tag{14}$$

$$f^0_{i,j} + f^1_{i,j} = 1. \tag{15}$$

Our objective function (10) is to minimize the number of SGP^e and CGP^e. The same α as basic ILP formulation in Section IV-A is used for balancing the cost. For each pair of partitions, there are two cases: 1) only one of them is flipped; and 2) flipping both or none of them. We can easily precompute the cost for each case, stored as $(s^{e0}_{i,j} + \alpha c^{e0}_{i,j})$ or $(s^{e1}_{i,j} + \alpha c^{e1}_{i,j})$.

Constraints (11) and (12) specify the case if both or neither of the partitions flip their coloring. Constraints (13) and (14) specify the case if only one of two partitions flips the coloring. Only one case can happen and this is formulated as Constraint (15).

It should be noted that, in our implementation, we do not explicitly impose Constraint (15). Instead, we substitute f^1_{ij} by $(1 - f^0_{ij})$ in (10)–(14) based on (15). This helps further reduce the number of variables and constraints.

V. GRID MODEL FOR PRACTICAL DESIGN ISSUES

In this section, we will present how our proposed grid model can handle various splitting rules and design patterns in Section V-A and Section V-B, respectively.

A. *Practical Splitting Rules*

Various manufacturability issues could impose many practical constraints on the locations of the stitches. Our grid model can be extended to satisfy these requirements. In the following, we will mainly focus on two major DPL-related guidelines, minimum width and minimum overlapping requirements.

1) *Minimum Width:* The $\min_{wi}$ violation can result from careless splittings as Fig. 15 shows. It could be located in the ending parts of polygons like Fig. 15(a) and (d), or created by two close stitches as Fig. 15(b) and (e) show.

In most process technology, $\min_{wi}$ is smaller than or equal to the half pitch, $0.5 \min_{cs}$, which is illustrated in Fig. 15(a) and (b). Our grid model can successfully avoid these extra constraints implicitly. By only allowing splitting on the boundary of the grids as shown in Fig. 15(c), the resulting small layout segments from splitting will be bounded from lower side by one grid size, i.e., $0.5 \min_{cs}$. The minimum width rule will be automatically satisfied, and there will be no pitfalls when we work on grids.

For the technology which has a $\min_{wi}$ larger than one grid width additional constraints can be augmented into our ILP formulation to ensure minimum width rule. We assume $\min_{wi}$ is still less than two-grid width here just for illustration purpose, and similar ideas can be applied for even larger $\min_{wi}$ requirement. For the example in Fig. 15(d), we can enforce the coloring of grid A and B identical to avoid minimum width violation. We are also able to specify constraints to eliminate the situation resulting from adjacent stitches as Fig. 15(e) indicates. A pair of stitches, S_1 and S_2, within one grid distance will not allowed to be selected simultaneously.

2) *Minimum Overlapping Margin:* The possible $\min_{ol}$ violation comes from the extra extension over the splitting locations, which may result in additional coloring conflict, such as the case from (a) to (b) in Fig. 16. A and B initially do not form a PCGP based on the definition in Section III-B, although they are in the same color. On the existence of some possible splitting, the extended metal could bring them into a distance smaller than $\min_{cs}$, which causes a coloring conflict.

This issue does exist in the process with 5–6 nm overlay error. To encounter this problem, when we are extracting PCGPs, the extra extension needs to be included for calculating distance between two grids. As the example in Fig. 16, when A and C or B and D have different colors, the overlapping error should be considered.

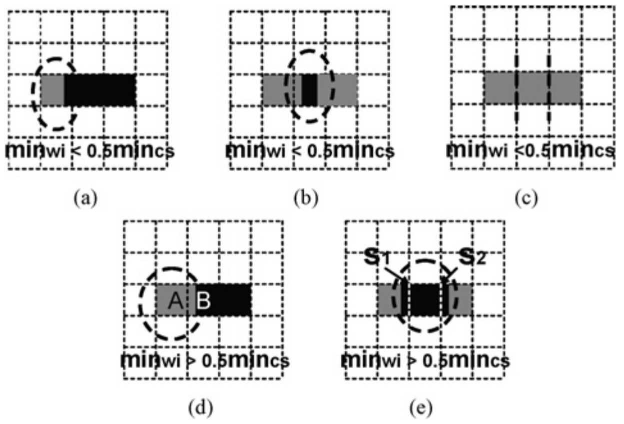


Fig. 15. Grid model can handle minimum width requirement.

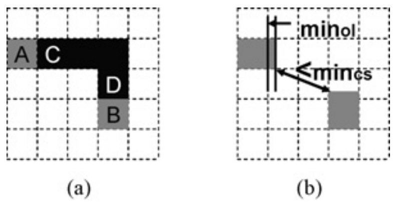


Fig. 16. Grid model can handle minimum overlap requirement.

On the other side, benefiting from possible further improvement on optical engineering, the overlap margin may not be a problem for our grid model in most cases. Remind that we are performing optimization based on the grids. For any pair of grids which do not form a PCGP, at least one of x and y dimensional distance will be two grid size, $\min_{cs}$. As the research works show [9], $\min_{ol}$ will be possibly controlled below 3 nm. With such a small overlap margin, the diagonal distance between A and B will still be larger than $\min_{cs}$, when taken as 64 nm.

B. Non-grid-mappable Layout

The grid model not only works on regular designs, it can also be extended to handle non-grid-mappable layout.

1) *Off-Grid Layout:* In deep submicro technology, although on-grid patterns are commonly favorable, there still exist off-track wires on lower layer metals, as illustrated by Fig. 17(a). Pattern A is not aligned with the grid lines. Under such case, we are not able to apply our grid-based formulation directly.

To resolve this issue, if a grid has any layout object, we will assign a binary grid variable for it. This is denoted as "relaxed grid mapping". As the example Fig. 17(b) shows, grids A1 and A2 will both be considered occupied. Moreover, to exclude false detection, additional consideration will be required when we formulate our mathematical programming.

First of all, we need to pay extra effort to check whether a pair of grids are within $\min_{cs}$ or connected, which is the crucial factor for determining PCGP or PSGP. Instead of using the grid number-based measurement as in Section III-B, we have to work on the distance or connection information of the underlying physical layouts.

TABLE III
TEST CASES

ckt	Area	Grid Array Size	#OG	#PCGP	#PSGP
C1	89	294 × 294	6670	21 215	5926
C2	160	395 × 395	15 710	48 007	14 143
C3	207	450 × 450	20 496	63 403	18 461
C4	292	534 × 534	33 497	10 5641	30 314
C5	422	642 × 642	53 998	172 826	49 167
C6	540	726 × 726	68 820	214 527	62 387
C7	747	854 × 854	10 1431	323 890	92 493
C8	1028	1002 × 1002	142 535	447 441	129 172

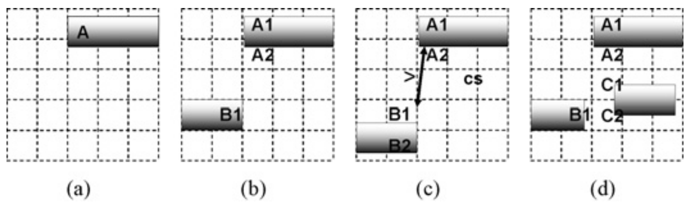


Fig. 17. Grid model can handle off-grid layout.

Fig. 17(b) and (c) show the need for checking $\min_{cs}$ condition for PCGP by distance, not the number of grids. In Fig. 17(b), the distance between grids A2 and B1 is one grid unit, smaller than two-grid unit threshold. This is consistent with the fact that the distance between related patterns is smaller than $\min_{cs}$. They indeed form a PCGP. On the other side, in Fig. 17(c), although grids A2 and B1 are away from each other by one grid, the distance of underlying design objects is no less than $\min_{cs}$. They are not a PCGP. If determining the distance only by grid-based unit, we will draw false conclusion. Similarly, it is the correct way to determine whether two grids are linked by checking the layouts instead of grid occupancy status. Fig. 17(d) shows an example where A1 and B1 are actually not connected. Grid-based judgment will falsely consider they are linked together because A2, B1, C1, and C2 are all occupied grids.

Moreover, we also need to exclude the unfeasible stitch locations resulting from "relaxed grid mapping". For the stitch candidate between A1 and A2 in Fig. 17(b), since the splitting will cause minimum width violation, it should be forbidden.

2) *Fat Wire:* When dealing with fat wires, we can map them into multiple grids. Although this will increase the complexity of the ILP formulation because of the dense clustering of occupied grids, we can apply previously proposed grid merging technique to reduce the problem size. Practically, a great portion of grids inside the wide wire can be merged.

VI. Experimental Results

In our benchmarks, eight industrial designs are scaled down to 32 nm. The metal1 for each test case is used for the experiments, which is one of the most troublesome layers in terms of double patterning lithography. The detailed information is shown in Table III. The first column "ckt" denotes the circuit name, "area" is the chip area in terms of um^2, "grid array size" shows the number of rows by the number of columns in our layout grid array. "#OG," "#PCGP," and "#PSGP" give the number of OGs, PCGPs and PSGPs, respectively.

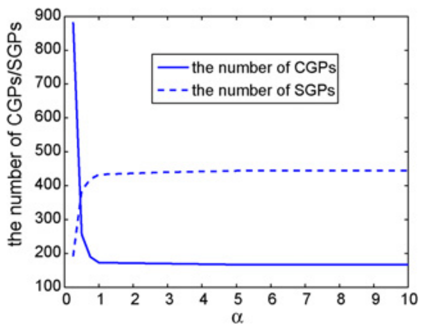


Fig. 18. Performance of our algorithm with different α values.

We implement our algorithm in C++ and test on Intel Core 3.0 GHz Linux machine with 32G RAM. Moreover, we use glpk [12] as our ILP solver and hMetis [13] for min-cut partition. The threshold W_t for each partition is 1500. We study different α settings in the ILP objective function. As shown in Fig. 18, when we start to increase α with higher penalty on conflict, the number of CGPs/SGPs drops/climbs obviously. After certain value, it has little effect, because the ILP formulation has reached its best point to reduce conflicts. In our work, we set α as 10 for all the benchmarks.

A. Result Comparison

We implement two different layout decomposition algorithms for comparison. To be fair, the same conflict interpretation defined in Section III-B will be applied in our experiments. Specially, although two features belong to the same net, as long as they are locally isolated, they could still result in a conflict.

We first prepare a greedy two-phase layout decomposition flow for comparative study, which adopts construct-and-fix methodology as in the previous works [7], [8]. We first color all the layout features sequentially. Each feature will be assigned to a color which can minimize the current number of conflicts. In the second phase, the violations are detected and corrected by inserting stitches. This is done by flipping the coloring of conflict segments, which basically splits certain features. Finally, the decomposition solution is mapped back to grids for comparison.

As the second comparative method, we also implement a design methodology based on the previous ILP-based work [11]. The conflict cycle will be removed iteratively first, followed by an ILP formulation to minimize the number of stitches. We are not able to compare with [11] directly because some of our main objectives are different. In their work, the unresolved conflict cycle is used for judging the indecomposable patterns, while we apply much finer metric, conflict pair grid. To resolve the issue, as the last step, we perform an additional grid-based greedy coloring run for the detected unresolved conflict cycles. The decomposition results will be mapped into grids in the end.

The detailed comparison is shown in Table IV. Under "two-phase approach", "1CGP" is the number of CGPs after the first step coloring and "uCGP" is the number of unresolved CGPs after inserting stitches. "CGP" under "previous ILP-based work" and "our algorithm" shows the final unresolved CGPs. We also list the results of "previous ILP-based work" when the conflict cycle removal iteration is set as 1 and 5, which are reported in columns with postfix name "Ite. 1" and "Ite. 5," respectively. For all the three approaches, "SGP" is the final number of stitch grid pairs and "central processing unit (CPU)" is the runtime by second. "Total" is the total number of all the test cases, and "ratio" is the average of their individual ratios.

Although "two-phase approach" is much faster, our algorithm significantly outperforms its results in terms of quality. "two-phase approach" can indeed eliminate the number of CGPs by averagely 52% after inserting stitches. However, lack of the careful planning, their coloring in the first step produces very poor starting solution, and there are a big amount of unresolved conflicts left after possible splitting. In contrast, our simultaneous method can averagely reduce the number of unresolved conflict grid pairs by about 87.6% with 33% less stitch grid pairs.

When compared to the previous ILP-based work [11], we can also achieve averagely 87.2% conflict and 10% stitch reduction. "previous ILP-based work" only greedily eliminates the troublesome conflict cycle without global picture in mind. Although a little better than "two-phase approach," their approach generates much degraded results than our algorithm in terms of conflict. On the other side, because it applies ILP to optimize the stitch number, their splitting decision is close to our simultaneous optimization result. However, because "previous ILP-based work" also considers coloring and splitting separately, its stitch number is still 10% more than ours. Also, from the breakdown of the solutions by different number iterations, we can see that iterative conflict removal can help improve results but still not enough due to the lack of global view.

In DPL, zero CGPs is desired in final tape out but the high complexity of modern designs makes it almost a must to go though tedious *design–decomposition–redesign* iterations. Our simultaneous flow with much higher quality solution can reduce expensive redesign effort as well as the number of iterations, which may eventually converge to a *clean* design much more quickly. Runtime for layout decomposition is not an issue as long as it is affordable.

B. Efficiency

The naive implementation of basic ILP formulation has prohibitive problem size, and it is not able to finish any benchmark in one day. Comparatively, our algorithm effectively reduces the runtime. In Table IV, the column "CPU" under "our algorithm" shows that we can obtain the solution in a few seconds. For the biggest benchmark, it takes a little more than one minute. Fig. 19 also shows the scalability of our algorithm, and the runtime grows linearly with the number of occupied grids in the design. Moreover, our acceleration techniques sacrifice little optimality.

Next, we will show the effectiveness of our grid merging technique. We achieve the same number of conflict and stitch number for all the test cases with and without this option while independent component computing and layout partition are still applied. Fig. 20 also illustrates the runtime comparison.

TABLE IV
RESULT COMPARISON

	Two-Phase Approach				Previous ILP-Based Work [11]							Our Algorithm		
ckt	1CGP	uCGP	SGP	CPU(s)	CGP(Ite. 1)	SGP(Ite. 1)	CGP(Ite. 5)	SGP(Ite. 5)	CGP	SGP	CPU	CGP	SGP	CPU(s)
C1	401	272	70	0.2	413	98	412	98	412	98	0.8	110	88	5.6
C2	1765	939	389	0.4	1089	287	1029	283	1015	282	1.7	160	220	6.4
C3	1799	779	416	0.5	774	206	735	199	720	198	1.8	129	175	8.3
C4	4232	2084	620	0.6	2143	469	1998	463	1972	459	2.9	171	452	13.0
C5	8125	4408	1325	1.0	3886	1078	3503	1056	3478	1059	4.9	367	1001	30.9
C6	9052	4625	1621	1.2	5082	1297	4761	1282	4731	1291	5.2	607	1112	37.3
C7	13 607	5551	2753	1.8	6415	1831	5653	1811	5530	1817	7.6	606	1651	46.6
C8	18 975	9223	3038	2.4	9805	2599	9050	2503	8941	2510	11.8	949	2271	57.6
Total	57 956	27 881	10 232	8.1	29 607	7865	27 141	7695	26 799	7714	36.7	3099	6970	205.7
Ratio	16.6	8.1	1.5	0.043	9.55	1.13	8.76	1.10	7.9	1.12	0.18	1	1	1

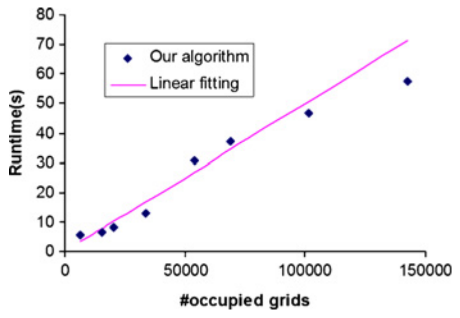


Fig. 19. Runtime of our algorithm versus number of occupied grids.

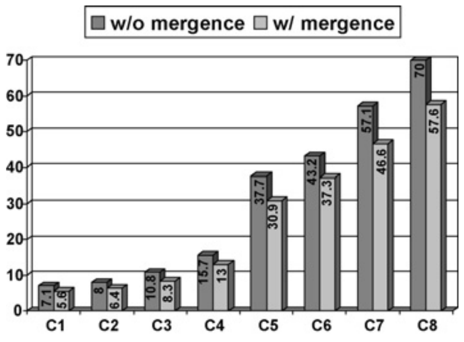


Fig. 20. CPU times of our algorithm with and without grid merging techniques.

TABLE V
STATISTICS ON THE INDEPENDENT COMPONENTS

ckt	#InComp	#w/o Partition	%w/o Partition
C1	181	178	98.3%
C2	362	357	98.6%
C3	688	681	99.0%
C4	838	824	98.3%
C5	1088	1070	98.3%
C6	1442	1420	98.5%
C7	1977	1951	98.7%
C8	3179	3147	99.0%

TABLE VI
ILP FORMULATION STATISTICS

	Reduced Problem Size		Coloring Flipping	
ckt	#max$_v$	#max$_c$	#max$_v^{cl}$	#max$_c^{cl}$
C1	804	1333	2	4
C2	867	1445	2	4
C3	873	1435	2	4
C4	904	1469	3	12
C5	911	1499	2	4
C6	902	1478	3	8
C7	921	1511	3	12
C8	923	1522	4	20

TABLE VII
RESULTS ON COLORING FLIP OPTIMIZATION

	Without Coloring Flip				With Coloring Flip			
ckt	CGP_{lp}	SGP_{lp}	CGP_{lp}^e	SGP_{lp}^e	CGP_{lp}	SGP_{lp}	CGP_{lp}^e	SGP_{lp}^e
C1	28	21	1	5	27	20	0	4
C2	18	22	9	4	12	20	3	2
C3	16	22	2	5	14	19	0	2
C4	37	70	10	11	31	66	4	7
C5	121	172	105	22	36	171	20	21
C6	65	98	13	20	55	90	3	12
C7	79	105	33	23	55	92	17	10
C8	108	142	79	28	88	127	59	13
Total	472	652	252	118	318	605	106	71
Ratio	1	1	1	1	0.75	0.92	0.30	0.60

The number on the bar is the exact CPU time in terms of second. As it is indicated, we can achieve approximately 19% speed-up. This mainly comes from the reduction of variables and constraints in the mathematical formulation.

Table V lists the statistics on the independent components. "#InComp" is the total number of independent components. "#w/o partition" and "%w/o partition," respectively, show the number and ratio of independent components, which are under partition threshold value W_t. As we can see, most components can be directly handled by ILP without performing layout partition and losing any optimality.

Table VI further shows the statistics on our ILP problem size. "#max$_v$" and "#max$_c$," respectively, give the maximum number of variables and constraints of the basic formulation with three proposed reduction techniques applied. Moreover, "#max$_v^{cl}$" and "#max$_c^{cl}$" list the maximum number of variables and constraints, respectively, of ILP formulation, which is applied in the coloring flip optimization.

As we can see from Table VI, the maximum ILP size is well controlled by the layout partition through the tuning threshold parameter W_t. W_t explicitly sets the upper bound for total number of grids, SGPs and CGPs within each sub problem. Therefore, the number of variables and constraints can be implicitly ensured in a reasonable range. Moreover, Table VI indicates the coloring flip optimization has relatively very small problem size, and hence can be handled with little effort.

C. Coloring Flip Optimization

Table VII shows the improvement when coloring flip is applied to merge solutions. This optimization will only be applied to relatively bigger independent components, which require proposed layout partition technique to further reduce problem size. Therefore, in Table VII, we only list the statistics for these bigger components in the respective benchmarks. The conflict and stitch number from smaller components without layout partitioning applied are not included.

In Table VII, "CGP_{lp}" and "SGP_{lp}" denote the total number of CGPs and SGPs for the independent components which adopt layout partition. The percentage of this type of components is very small, as shown in Table V. However, their conflict and stitch number have relatively much bigger portion when compared to the respective data under column "our algorithm" in Table IV.

"CGP^{e}_{lp}" and "SGP^{e}_{lp}" are the number of corresponding external conflict and stitch grid pairs. The results show that there are outstanding "CGP^{e}_{lp}" and "SGP^{e}_{lp}" for further optimization. "with coloring flip" can reduce CGP^{e}_{lp} and SGP^{e}_{lp} by 70% and 40%, about 25% and 8% for total CGPs and SGPs. This experiment demonstrates the necessity of coloring flip optimization and the effectiveness of our ILP-based approach. The CPU time difference between "without coloring flip" and "with coloring flip" is very tiny and not listed.

VII. Conclusion

In this paper, we have developed a double patterning aware layout decomposition flow for simultaneous conflict and stitch minimization. Experimental results are very promising. In future, we would like to study earlier stage placement/routing, and standard cell designs to produce DPL-friendly layout.

Acknowledgment

The authors would like to thank Dr. M. Cho at IBM Research for his helpful discussions on this problem.

References

[1] M. Dusa, J. Finders, and S. Hsu, "Double patterning lithography: The bridge between low k1 ArF and EUV," *Microlithography World*, Feb. 2008.

[2] M. Dusa, J. Quaedackers, O. F. A. Larsen, J. Meessen, E. van der Heijden, G. Dicker, O. Wismans, P. de Haas, K. van Ingen Schenau, J. Finders, B. Vleeming, G. Storms, P. Jaenen, S. Cheng, and M. Maenhoudt, "Pitch doubling through dual-patterning lithography challenges in integration and litho budgets," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6520. 2007, pp. 65200G.1–65200G.10.

[3] S.-M. Kim, S.-Y. Koo, J.-S. Choi, Y.-S. Hwang, J.-W. Park, E.-K. Kang, C.-M. Lim, S.-C. Moon, and J.-W. Kim, "Issues and challenges of double patterning lithography in DRAM," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6520. 2007, pp. 65200H.1–65200H.7.

[4] V. Wiaux, S. Verhaegen, S. Cheng, F. Iwamoto, P. Jaenen, M. Maenhoudt, T. Matsuda, S. Postnikov, and G. Vandenberghe. "Split and design guidelines for double patterning," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6924. 2008, pp. 692409.1–692409.11.

[5] Y. Borodovsky, "Lithography 2009 overview of opportunities," *Semicon West*, 2009.

[6] C. Bencher, Y. Chen, H. Dai, W. Montgomery, and L. Huli, "22 nm half-pitch patterning by CVD spacer self alignment double patterning (SADP)," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6924. 2008, pp. 69244E.1–69244E.7.

[7] M. Drapeau, V. Wiaux, E. Hendrickx, S. Verhaegen, and T. Machida, "Double patterning design split implementation and validation for the 32 nm node," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6521. 2007, pp. 652109.1–652109.15.

[8] 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," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6924. 2008, pp. 69243M.1–69243M.10.

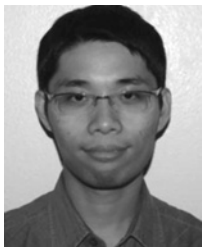
[9] W. Arnold, "Toward 3 nm overlay and critical dimension uniformity: An integrated error budget for double patterning lithography," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6924. 2008, pp. 692404.1–692404.9.

[10] J. Rubinstein and A. Neureuther, "Post-decomposition assessment of double patterning layout," in *Proc. Soc. Photo-Optic. Instrum. Eng.*, vol. 6924. 2008, pp. 69240O.1–69240O.12.

[11] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *Proc. Int. Conf. Comput. Aided Design*, Nov. 2008, pp. 465–472.

[12] http://www.gnu.org/software/glpk/glpk.html/

[13] http://glaros.dtc.umn.edu/gkhome/views/metis



Kun Yuan received the B.S. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004. He is currently working toward the Ph.D. degree in electrical and computer engineering at the University of Texas, Austin.

He was with TeraRoute, Austin, TX, during the summer of 2007 as a software engineering intern, and with NVIDIA, Santa Clara, CA, during the summer of 2009 as a hardware engineering intern. His current research interests include physical design automation for manufacturability and numerical optimization.

Mr. Yuan received the Microelectronic and Computer Development Fellowship for 2006–2008, the International Symposium on Physical Design Routing Contest Award in 2007, and the Best Paper Award Nomination at the Asian and South Pacific Design Automation Conference in 2010.

Jae-Seok Yang received the B.S. degree in electrical engineering from Sogang University, Seoul, Korea, in 1997, and the M.S. degree in electrical engineering and computer science from the University of California, Berkeley, in 2007. He is currently working toward the Ph.D. degree in electrical and computer engineering at the University of Texas, Austin.

From 1999 to 2005, he was with Samsung Semiconductor Research Center, Hwasung, Korea. His research interests include nanometer very large scale integration design for manufacturability and design automation for 3-D integrated circuit design.

Mr. Yang was the recipient of the Best Paper Award at the System-on-a-Chip Design Conference, Seoul, Korea, in 2002, the Samsung Scholarship in 2005, and the Best Paper Award Nomination at the Asian and South Pacific Design Automation Conference in 2010.

David Z. Pan (S'97–M'00–SM'06) received the Ph.D. degree in computer science from the University of California, Los Angeles, in 2000.

From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY. He is currently an Associate Professor and the Director with the UT Design Automation Laboratory, Department of Electrical and Computer Engineering, University of Texas, Austin. He has published over 100 refereed papers in international conferences and journals, and is the holder of six U.S. patents. His research interests include nanometer very large scale integration (VLSI) physical design, design for manufacturing, low-power vertical integration design and technology, and design/computer aided design (CAD) for emerging technologies.

Dr. Pan has served as an Associate Editor for the IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), the IEEE Transactions on VLSI Systems, the IEEE Transactions on

CIRCUITS AND SYSTEMS-PART I, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-PART II, and the IEEE CIRCUITS AND SYSTEMS SOCIETY NEWSLETTER. He was also a Guest Editor of the TCAD Special Section on "International Symposium on Physical Design (ISPD)" in 2007 and 2008. He serves as the Chair of the IEEE Comuter Aided Network Design Committee and the Association for the Computing Machinery (ACM)/Special Interest Group on Design Automation (SIGDA) Physical Design Technical Committee. He is in the Design Technology Working Group of the International Technology Roadmap for Semiconductor. He has served in the Technical Program Committees of major VLSI/CAD conferences, including the Asia and South Pacific Design Automation Conference (ASPDAC) (Topic Chair), the Design Automation Conference, the Design, Automation and Test in Europe, the International Conference on Computer Aided Design (ICCAD), the ISPD (Program Chair), the International Symposium on Quality Electronic Design (Topic Chair), the International Symposium on Circuits and Systems (CAD Track Chair), the Proceedings of the System-Level Interconnect Prediction (Publication Chair), the Great Lakes Symposium on VLSI, the Austic Conference on Integrated Systems and Circuits (Program Co-Chair), International Conference on Integrated Circuit Design and Technology, and the VLSI-Design, Automation, and Test (EDA Track Chair). He is the General Chair of the ISPD 2008 and the Steering Committee Chair of the ISPD 2009. He is a Member of the Technical Advisory Board of Pyxis Technology, Inc.

Dr. Pan has received a number of awards for his research contributions and professional services, including the ACM/SIGDA Outstanding New Faculty Award in 2005, the National Science Foundation CAREER Award in 2007, the Semiconductor Research Corporation Inventor Recognition Award thrice in 2004, 2005 and 2006, the IBM Faculty Award thrice from 2004 to 2006, the UCLA Engineering Distinguished Young Alumnus Award in 2009, the IBM Research Bravo Award in 2003, the SRC Techcon Best Paper in Session Award in 1998 and 2007, the Best Student Paper Award from the 2009 IEEE International Conference on Integrated Circuit Design and Technology, the Dimitris Chorafas Foundation Research Award in 2000, the ISPD Routing Contest Awards in 2007, the eASIC Placement Contest Grand Prize in 2009, five Best Paper Award Nominations at the Design Automation Conference/ICCAD/ASPDAC, and the ACM Recognition of Service Award in 2007 and 2008. He was a Cadence Distinguished Speaker in 2007 and an IEEE Circuits and Systems Society Distinguished Lecturer during 2008–2009.