Double Patterning Layout Decomposition for Simultaneous Conflict and Stitch Minimization

Kun Yuan, Jae-Seok Yang, David Z. Pan
ECE Dept. Univ. of Texas at Austin, Austin, TX 78712
{kyuan, jsyang}@cerc.utexas.edu, dpan@ece.utexas.edu

ABSTRACT

Double patterning lithography (DPL) is considered as a most likely solution for 32nm/22nm technology. In DPL, the layout patterns are decomposed into two masks (colors). Two features (polygons) have to be assigned opposite colors if their spacing is less than certain minimum coloring distance. 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 be split into two parts to resolve the conflict but the resulting stitch causes yield loss due to overlay error and increases manufacturing cost. While previous layout decomposition approaches perform coloring and splitting separately, in this paper, we propose an algorithm to minimize the number of conflicts and stitches simultaneously. Our algorithm is based on grid layout model and integer linear programming. Two techniques, independent component computation and layout partition, are proposed to reduce runtime of the algorithm. The experimental results show that, compared with the two phase decomposition flow, the proposed algorithm reduces the conflicts significantly using less stitches under reasonable runtime.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuit**]: Design Aids

General Terms

Algorithms, Design

Keywords

Layout Decomposition, Double Patterning Lithography, Integer Linear Programming

1. INTRODUCTION

As the minimum feature size decreases, semiconductor industry is facing the limitation of patterning sub-32nm due to the delay of the next generation lithography equipment such as Extreme Ultra Violet (EUV) [1]. Double patterning lithography (DPL) [2, 3] emerges almost the only alternative for 32nm/22nm nodes and it is already used for memory products. 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.

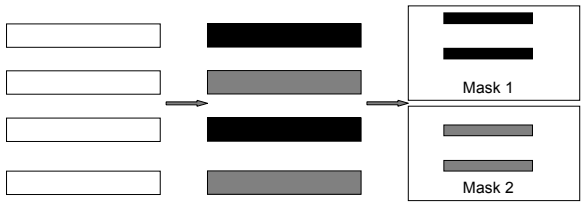


Figure 1: One single design is decomposed into two masks and the pitch size is increased effectively in DPL.

Decomposition [2, 4, 5] is a process that assigns opposite colors if the distance between two features is less than the minimum coloring spacing. A layout in modern design may contain patterns which is unable to color. In this case, a feature may be split into two parts and colored differently to resolve the conflicts, which generates stitches. Stitches cause yield loss due to overlay error and they also increase manufacturing cost. 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 unresolved conflicts and stitches to save design and manufacturing effort.

There are a few layout decomposition works for DPL technology. A heuristic approach is suggested in [5] to cut troublesome patterns after finding the coloring conflicts. The patterns are pre-fragmented into smaller pieces in [6] 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. Recently, a practical layout decomposition flow is proposed in [7] 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 ILP formulation. However, in their work, the number of unresolvable conflict cycles and splitting stitches are not optimized together.

In this paper, we propose an algorithm to decompose layout for minimizing conflicts and stitches simultaneously. A good coloring solution would help reduce the number of required stitches, and considering potential stitch locations during coloring also enables better coloring scheme. This

ISPD'09, March 29–April 1, 2009, San Diego, California, USA.

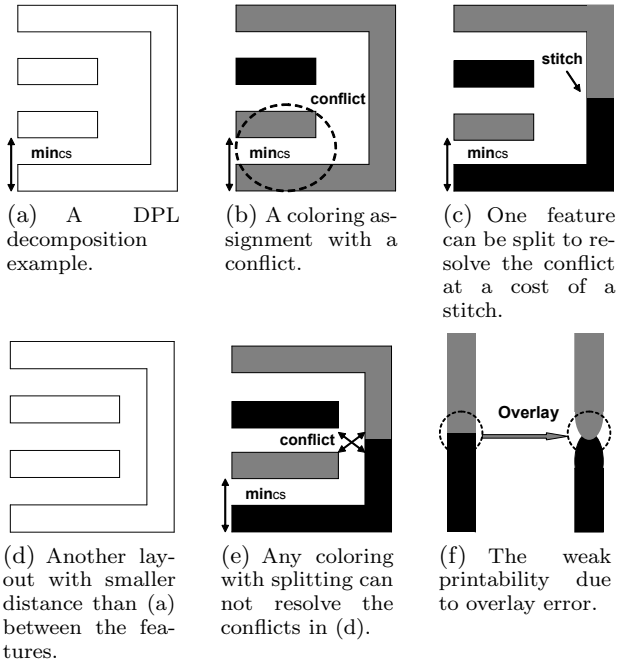
co-optimization can help improve the coloring assignment greatly. Our optimization is based on grid layout model and integer linear programming. Compared to a conventional two phase decomposition flow, the proposed algorithm reduces the conflicts by 8.1x with 33% less stitches. Although our approach is comparatively much 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 contribution is as follows:

1. We propose a new grid model to enable bigger solution space than previous works [5, 6] and perform simultaneous conflict and stitch optimization.
2. We develop an integer linear programming algorithm to minimize the number of conflicts and stitches for high quality solution.
3. We propose two speed-up techniques (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.

The rest of the paper is organized as follows. Section 2 provides the preliminaries on double patterning decomposition and motivation of simultaneous coloring and splitting optimization. In Section 3, we discuss the problem formulation with related model and definitions. The basic ILP formulation is described in Section 4 with two speed-up techniques. Section 5 presents the experiment results and Section 6 concludes this paper.

2. PRELIMINARIES AND MOTIVATION



(a) A DPL decomposition example.
(b) A coloring assignment with a conflict.
(c) One feature can be split to resolve the conflict at a cost of a stitch.
(d) Another layout with smaller distance than (a) between the features.
(e) Any coloring with splitting can not resolve the conflicts in (d).
(f) The weak printability due to overlay error.

Figure 2: This example explains the concept of conflict and stitch.

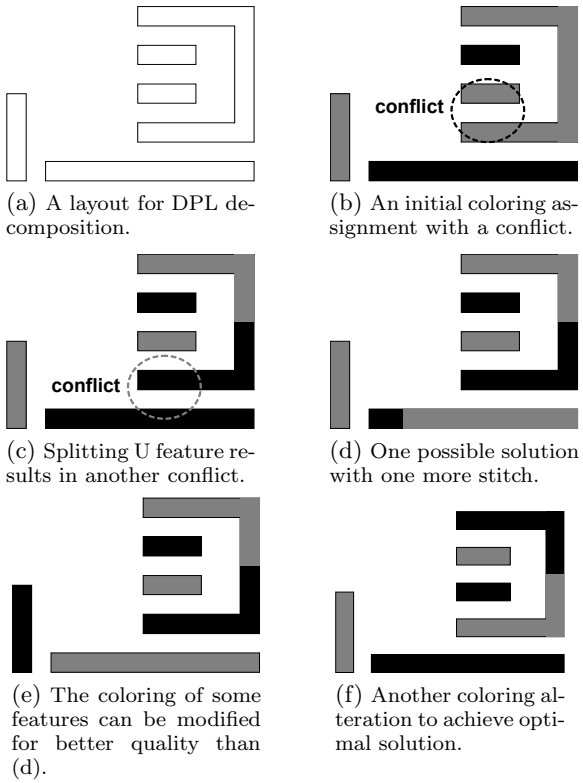
2.1 Double Patterning Lithography

The lithography technology has been facing its limit in printing smaller feature size 32nm/22nm. In single exposure infrastructure, as Rayleigh criterion describes, $hp = k1 * \lambda/NA$, the minimum printable half pitch hp depends on three parameters. $k1$ is the process difficulty factor, NA is numerical aperture and λ is the light wavelength. Currently, keeping pitch scaled becomes extremely difficult. $k1$ has been pushed into its lower bound, and the immaturity of EUV makes the 193nm wavelength remaining the main stream lithography in the product line. The high NA (1.35) is also challenged by the practical processing. One emerging feasible solution is double patterning lithography. The basic idea is to decompose the original layout into two masks as shown in Fig. 1 and print the design by two exposure/etching steps. The pitch size for each mask is effectively increased without changing the minimum feature size of the original design.

2.2 Layout Decomposition Considerations

There are two critical issues with DPL layout decomposition: coloring conflict and splitting stitch.

Coloring Conflict: If the distance between two separate features in the same mask is less than minimum coloring spacing min_{cs}, they should be assigned to different masks (colors). Otherwise, there will be a coloring conflict. This is common in complex design patterns. 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



(a) A layout for DPL decomposition.
(b) An initial coloring assignment with a conflict.
(c) Splitting U feature results in another conflict.
(d) One possible solution with one more stitch.
(e) The coloring of some features can be modified for better quality than (d).
(f) Another coloring alteration to achieve optimal solution.

Figure 3: This example shows the shortcoming of two phase layout decomposition flow in previous works [5, 6]. An unplanned coloring will need much extra effort during splitting.

coloring conflict will be unavoidable as in Fig. 2 (b). Sometime, violations can be eliminated by appropriately splitting the features like Fig. 2 (c). There are also unresolvable conflicts, as Fig. 2 (d) and (e) indicate, which require modifying the design.

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 indicated in Fig 2 (c). However, stitch can have negative effects such as yield loss due to overlay between the two masks Fig. 2 (f) and increase of manufacturing cost.

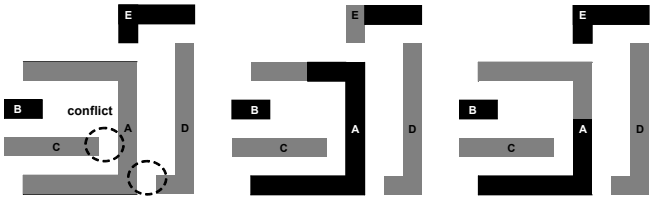
Without altering layout in the scope, the general objective of layout decomposition can be stated as minimizing the unresolved conflicts by introducing as few as possible stitches, because the main goal of DPL is to separate the features within min_{cs} to enable the printability despite the side effects of the stitches.

2.3 Simultaneous Optimization

The existing works insert stitches after coloring to resolve conflicts. Without planning possible splitting during coloring, it is not an easy task 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 need 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, it takes much extra effort to eliminate the conflict after coloring and has potential risk producing non optimal result. Given the complexity of modern design, the two phase approach will have extreme difficulty handling the exploding consideration. If the initial coloring configuration is poor, it may hardly produce high quality solution even after tremendous effort. This motivates us simultaneous conflict and stitch minimization during layout decomposition.

3. PROBLEM FORMULATION

3.1 Grid Layout Model



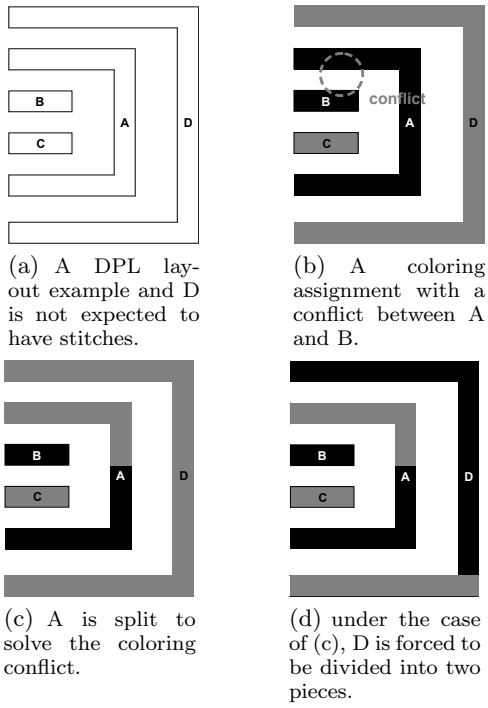
(a) A DPL layout example for conflict elimination.

(b) A possible splitting on A with one more cut on E.

(c) An alternative cut on A with no other stitches.

Figure 4: This example shows that different stitch candidates can lead to different solution qualities.

Considering splitting during coloring is a challenging problem. First of all, the stitch configurations are highly correlated and all the potential locations need be considered for global optimality. Fig. 4 (a) is a case with two conflicts. As we can see, two possible splitting choices on 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.



(a) A DPL layout example and D is not expected to have stitches.

(b) A coloring assignment with a conflict between A and B.

(c) A is split to solve the coloring conflict.

(d) under the case of (c), D is forced to be divided into two pieces.

Figure 5: This example shows the difficulty of predicting where the splitting is needed.

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), A will be split to resolve the conflict between A and B like Fig. 5 (c). As a result, D also needs to be broken into two segments as shown in Fig. 5 (d). From these examples, we can see that it is necessary to consider but difficult to determine all the stitch candidates.

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 patten, 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 [5, 6]. The discretization is reasonable because a design follows underlying regular pitches 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}.

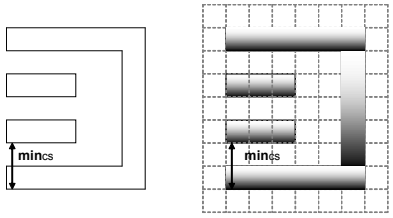
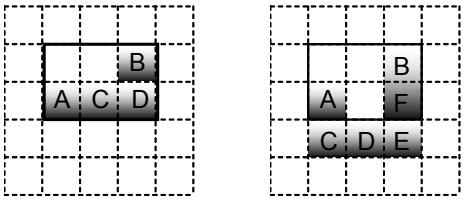


Figure 6: The proposed grid layout model.



(a) C-D is a blocking path

(b) C-F is not a blocking path.

Figure 7: The examples illustrate the blocking path for A and B, and the solid rectangle marks the bounding box.

3.2 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 $OG1$ and $OG2$, a blocking path is a path when

1. It is fully composed of the OGs and connects $OG1$ and $OG2$.
2. $OG1$ and $OG2$ are touching its two ending grids, respectively.
3. This path is within the bounding box of $OG1$ and $OG2$.

As shown in Fig. 7 (a), C-D is a blocking path for A and B. In another example Fig. 7 (b), C-F is not a BP because a part of it is beyond the bounding box.

Definition 3 *potential conflict grid pair* **PCGP** and *potential stitch grid pair* **PSGP**: Given two occupied grids OG1 and OG2,

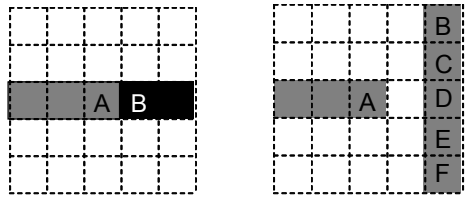
1. If the distance between OG1 and OG2 is less than min_{cs} and the two grids are not touching, they form a potential conflict grid pair.
2. If OG1 and OG2 are touching, they form a potential stitch grid pair.

The distance between a pair of OGs is the minimum distance between any two points from the OGs. Take Fig. 7 (b) as an example, 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.

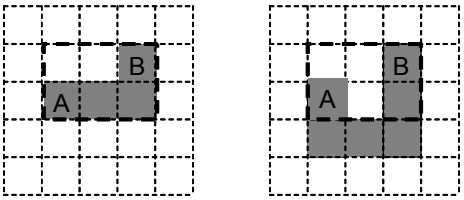
Definition 5 *conflict grid pair* **CGP**: If a PCGP is in the identical color, and there is no blocking path connecting them in the same mask, it is a conflict grid pair.

The definition of SGP is straightforward as 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 and not connected in the same mask, so any of them and A is 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 untouched grids are electrically connected through a blocking path, we should not consider it as a coloring conflict. As in Fig. 8 (c), A and B has a BP in the same mask between them, and it is indeed a normal jog. In contrast, although there is a path (not BP) connecting A and B in Fig. 8 (d), they are in fact isolated locally within the bounding box and would have weak printability.



(a) A and B is a SGP. They are touching and in different colors.

(b) A and any of B-F is a CGP because they are in the same mask and unconnected within minimum coloring spacing.



(c) A and B is not a CGP because there is a blocking path connecting them in the same mask.

(d) A and B is a CGP since the connection is not a blocking path.

Figure 8: The examples illustrate stitch grid pair and conflict grid pair for PSGP/PCGP A and B. The solid rectangle in (b) encloses all the grids whose distance to A is less than the minimum coloring distance, and the dash box in (c) and (d) is the bounding box of A and B.

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 non zero CGPs is essentially not manufacturable and a solution with less CGPs reduces expensive redesign effort.

4. ALGORITHM

In this section, we will present our integer linear programming based layout decomposition algorithm. The entire flow is shown in Figure 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, two speed up techniques, independent component computation and layout partition, are applied to divide the

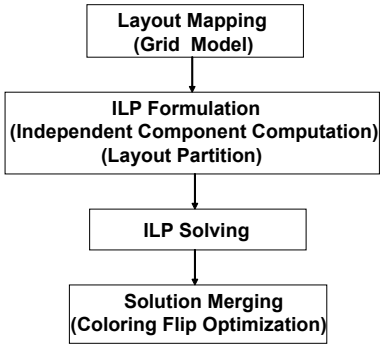


Figure 9: The overall layout decomposition flow.

Table 1: Notation for basic ILP formulation

$og_{i,j}$	occupied grid which i and j are its 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	the set of PSGPs.
CP	the set of PCGPs.
$P_{pq,uv}$	the set of BPs connecting $og_{p,q}$ and $og_{u,v}$.
$p^k_{pq,uv}$	the k_{th} BP connecting $og_{p,q}$ and $og_{u,v}$.
$n^k_{pq,uv}$	the 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.

whole problem into several smaller problems. 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.

4.1 Basic ILP Formulation

In this subsection, we will show how to formulate double pattern layout decomposition into ILP. To better present our method, we first describe the notation in Table 1.

The simultaneous coloring and splitting optimization can be formulated as follows:

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

subject to

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

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

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

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

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

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

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

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

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 CP and PSGPs SP are pre determined 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) evaluates the conditions for CGP. A conflict will be reported only if a PCGP grids are assigned same color and the possible BPs $g^k_{pq,uv}/b^k_{pq,uv}$ do not exist.

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

4.2 Speed-Up Techniques

In this section, we will discuss two techniques, independent component computation and layout partition to accelerate the ILP formulation in Section. 4. The key idea is to use divide and conquer heuristic to convert the problem into smaller subproblems.

4.2.1 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 break down the whole design into several independent components as shown in Fig. 10, and apply basic ILP formulation for each one. The overall solution can be taken as the union of all the components without effecting

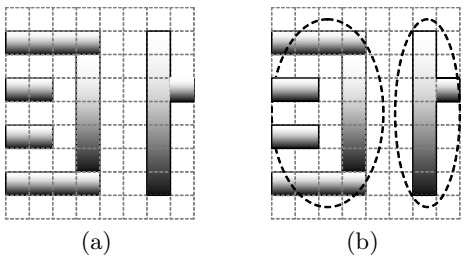


Figure 10: An example of breaking big layout into two independent components, which have no interacted PSGPs/PCGPs and are marked by the dash circle.

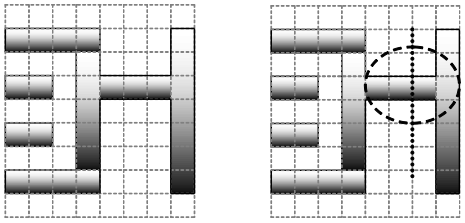
the global optimality. The runtime of ILP formulation scales down dramatically with the reduction of the variables and constraints, and the coloring assignment can be effectively accelerated.

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.

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.

4.2.2 Layout Partition

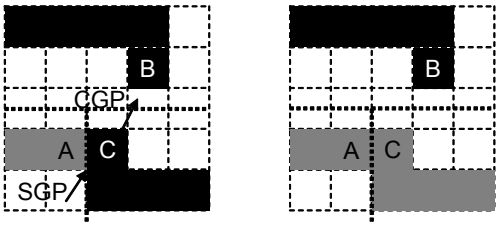


(a) (b)

Figure 11: An example of layout partition. The dotted line cuts the layout into two parts while the dash circle marks PCGP and PSGP locations across the boundary of the two partitions.

Some components 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 ILP approach for each one, indicated in Fig. 11. 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 pre defined 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 vertex weight of G. Suppose W is 10000 and W_t is 3000, the component will be partitioned into 4 parts.



(a) one possible mergence with one CGP and one SGP.

(b) another union with no CGPs/SGPs by flipping partition C coloring in (a).

Figure 12: Different coloring flips have distinct numbers of SGPs/CGPs across the boundaries, marked by the dot lines.

4.3 Solution Mergence

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 independent components, it comes a *coloring flip optimization* problem when we try to merge the solutions for all the partitions. After the unitization, the possible SGPs and CGPs for each partition can be divided into two disjoint subsets, the internal conflicts/stitches SGP^i/CGP^i which are inside the same partition and external conflicts/stitches SGP^e/CGP^e crossing the boundary of different partitions. Although we are not able to change SGP^i/CGP^i in the mergence, it is possible to reduce the number of SGP^e/CGP^e by flipping the coloring of some partitions. This is illustrated in Fig. 12, which has three partitions. The coloring mergence in Fig. 12(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. To optimize 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 are usually not large, we also use ILP formulation to solve this problem. The relevant notation can be found in Table 2.

Table 2: The 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.
$s^{e0}_{i,j}$	the number of stitches between partition i and j if both flip or do not flip the coloring.
$c^{e0}_{i,j}$	the number of conflicts between partition i and j if both flip or do not flip the coloring.
$s^{e1}_{i,j}$	the number of stitches between partition i and j if only one partition flips its coloring.
$c^{e1}_{i,j}$	the number of conflicts between partition i and j if only one partition flips its coloring.

The formulation is as follows:

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

subject to

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

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

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

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

$$f^0_{i,j} + f^1_{i,j} = 1 \quad (15)$$

Our objective function (10) is to minimize the number of SGP^e and CGP^e. The same α as basic ILP formulation in Section. 4.1 is used for balancing the cost. For each pair of partitions, there are two cases: (1) only one of them is flipped; (2) flipping both or none of them. We can easily pre compute 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 flip 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 Constraints (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.

5. EXPERIMENTAL RESULT

In our benchmarks, eight 180nm technology designs are scaled down to 32nm. The metal1 for each testcase is used for the experiments, because it is one of the most troublesome layers in terms of double patterning lithography. The detailed information is shown in Table 3. The first column 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.

We implement our algorithm in C++ and test on Intel Core 3.0GHz Linux machine with 32G RAM. Moreover, we use glpk [8] as our solver for integer linear programming and hMetis [9] for min cut partition. The threshold W_g for each partition is 1500. We study the different α settings in the ILP objective function. As shown in Fig. 13, 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. The reason is that the ILP formulation has reached its best point to eliminate the conflicts. In our work, we set α as 10 to ensure the priority of the conflict elimination for all the benchmarks.

Table 3: The test cases.

circuit	area	grid array size	#OG	#PCGP	#PSGP
C1	89	294x294	6670	21215	5926
C2	160	395x395	15710	48007	14143
C3	207	450x450	20496	63403	18461
C4	292	534x534	33497	105641	30314
C5	422	642x642	53998	172826	49167
C6	540	726x726	68820	214527	62387
C7	747	854x854	101431	323890	92493
C8	1028	1002x1002	142535	447441	129172

5.1 Comparison with Two Phase Approach

We implement a two-phase layout decomposition flow for comparative study, which adopts construct-and-fix methodology as in the previous works [5, 6]. We first color all the layout features sequentially. Each feature will be assigned 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. We are not able to compare with previous work [7] 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 grid pair.

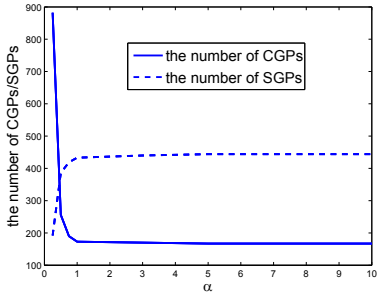


Figure 13: The performance of our algorithm with different α values.

The comparison between the two phase approach and our algorithm is shown in Table. 4. 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 "Our algorithm" shows the final unresolved CGPs. For both approaches, "SGP" is the number of stitch grid pairs and "CPU" is the runtime by second. "total" is the total number of all the testcases, and "avg 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 8.1x with 33% less stitch grid pairs.

Table 4: Result comparison

	Two phase approach				Our algorithm		
circuit	1CGP	uCGP	SGP	CPU(s)	CGP	SGP	CPU(s)
C1	401	272	70	0.2	110	88	7
C2	1765	939	389	0.4	160	220	8
C3	1799	779	416	0.5	129	175	11
C4	4232	2084	620	0.6	171	452	16
C5	8125	4408	1325	1.0	367	1001	38
C6	9052	4625	1621	1.2	607	1112	43
C7	13607	5551	2753	1.8	606	1651	57
C8	18975	9223	3038	2.4	949	2271	70
total	57956	27881	10232	8.1	3099	6970	250
avg ratio	16.6	8.1	1.5	0.04	1	1	1

In double patterning lithography layout decomposition, quality of results is much more important than computational time. Essentially, 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 to *clean* the layout. Our simultaneous flow with much higher quality solution can reduce expensive

redesign effort as well as the number of iterations, which may eventually converges to a *clean* design much more quickly. Runtime for layout decomposition is not an issue as long as it is affordable.

5.2 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 4, 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. 14 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. Table 5 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.

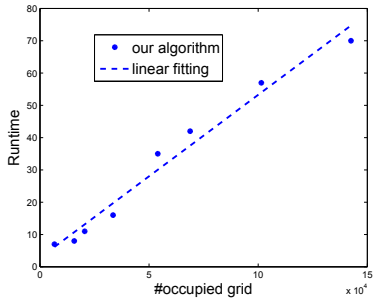


Figure 14: The runtime of our algorithm vs. number of occupied grids.

Table 5: Statistics on the independent components

name	#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%

5.3 Coloring Flip Optimization

Table 6 shows the improvement when coloring flip optimization is applied to merge solution. The CPU time difference between "Without coloring flip" and "With coloring flip" is very tiny and not listed. "CGP" and "SGP" denote the total number of CGPs and SGPs for the independent components requiring layout partition. "CGP^e" and "SGP^e" are the number of corresponding external conflict and stitch grid pairs. The results show that there are outstanding "CGP^e" and "SGP^e" for further optimization. "with coloring flip" can reduce CGP^e and SGP^e by 70% and 40%, about 25% and 8% for total CGPs and SGPs. This experiment demonstrates the necessary of coloring flip optimization and the effectiveness of our ILP based approach.

Table 6: Results on coloring flip optimization

	Without coloring flip				With coloring flip			
circuit	CGP	SGP	CGP^e	SGP^e	CGP	SGP	CGP^e	SGP^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
avg ratio	1	1	1	1	0.75	0.92	0.30	0.60

Overall, our ILP based layout decomposition is a very efficient and effective approach for double patterning lithography. As we observe, a solution with non-zero CGPs can hardly be achieved in one time and many conflicts are due to lithography unfriendly design. It implies that earlier stage cell layout and placement/routing can not be arbitrary and have to be more aware of double patterning lithography. We would like to pursue our research along this direction.

6. CONCLUSION

In this paper, we have developed a double patterning aware layout decomposition flow for simultaneous conflict and stitch minimization. Our approach is featured by grid layout model and integer linear programming. We also propose two speed-up techniques: independent component computation and layout partition. Experimental results show that our approach can achieve significantly better conflict reduction with less stitches within reasonable runtime.

For future work, we plan to keep improving the efficiency of our approach, although the runtime complexity shows linearity. The adjacent grids with similar neighboring conditions can be possibly merged. Moreover, we would like to explore more comprehensive metrics for helping double patterning lithography, and study earlier stage placement/routing, and standard cell designs to produce DPL-friendly layout.

7. ACKNOWLEDGMENT

This work is supported in part by NSF, SRC, Sun, Qualcomm, and equipment donations from Intel. The authors would also like to thank Dr. Minsik Cho at IBM research for helpful discussions on this problem.

8. REFERENCES

[1] Micrea Dusa, Jo Finders, and Stephen Hsu. Double patterning lithography: The bridge between low k1 arf and euv. In *Microlithography World*, Feb 2008.

[2] George E. Bailey, Alexander Tritchkov, Jea-Woo Park, Le Hong, Vincent Wiaux, Eric Hendrickx, Staf Verhaegen, Peng Xie, and Janko Versluijs. Double pattern EDA solutions for 32nm HP and beyond. In *Proc. SPIE 6521*, 2007.

[3] 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. of SPIE*, volume 6924, 2008.

[4] A. B. Kahng. Key Directions and a Roadmap for Electrical Design for Manufacturability. In *Proc. European Solid-State Circuits Conf*, 2007.

[5] M. Drapeau, V. Wiaux, E .Hendrickx, S. Verhaegen, and T. Machida. Double patterning design split implementation and validation for the 32nm node. In *Proc. of SPIE*, volume 6521, 2007.

[6] Tsann-Bim Chiou, Robert Socha, Hong Chen, Luoqi Chen, Stephen Hsu, Peter Nikolsky, Anton van Oosten, and Alek C. Chen. Development of layout split algorithms and printability evaluation for double patterning technology. In *Proc. of SPIE*, March 2008.

[7] Andrew B. Kahng, Chul-Hong Park, Xu Xu, and Hailong Yao. Layout decomposition for double patterning lithography. In *Proc. Int. Conf. on Computer Aided Design*, November 2008.

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

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