WISDOM: Wire Spreading Enhanced Decomposition of Masks in Double Patterning Lithography *

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

ABSTRACT

In Double Patterning Lithography (DPL), conflict and stitch minimization are two main challenges. Post-routing mask decomposition algorithms [1–4] may not be enough to achieve high quality solution for DPL-unfriendly designs, due to complex metal patterns. In this paper, we propose an efficient framework of WISDOM to perform wire spreading and mask assignment simultaneously for enhanced decomposability. A set of Wire Spreading Candidates (WSC) are identified to eliminate coloring constraints or create additional splitting locations. Based on these candidates, an Integer Linear Programming (ILP) formulation is proposed to simultaneously minimize the number of conflicts and stitches, while introducing as less layout perturbation as possible. To improve scalability, we further propose three acceleration techniques without loss of solution quality: odd-cycle union optimization, coloring-independent group computing, and suboptimal solution pruning. The experimental results show that, compared to a post-routing mask decomposition method [2], we are able to reduce the number of conflicts and stitches by 41% and 23% respectively, with only 0.43% wire length increase. Moreover, with proposed acceleration methods, we achieve 9x speed-up.

1. INTRODUCTION

As 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) [5]. Double patterning lithography is currently the forerunner for 32nm, 22nm, and even 16nm technology [6]. In DPL, the original layout will be decomposed into two masks, e.g., BLACK and GRAY, and manufactured through two exposure/etching steps. As the benefit, the effective pitch can be doubled, which improves lithography resolution.

There are two critical issues with decomposition of masks in DPL [7,8]: coloring conflict and splitting stitch. If the distance between two polygons is less than minimum coloring spacing min_{cs}, they should be assigned different masks. Otherwise, there will be a conflict. Sometimes, a feature may be split into two touching parts and colored differently to resolve conflict. However, this introduces stitches, which cause yield loss due to overlay error and increase manufacturing cost. Therefore, conflict and stitch minimization are two of the main challenges in DPL.

Many researches focus on post-routing mask decomposition. A novel flow is proposed in [1] to optimize splitting locations with ILP. Xu et al. [4] present an efficient graph reduction based algorithm for stitch minimization, and Yang et al. [9] propose a fast partition-based approach. In these works, conflicts are eliminated in a greedy way. To enable simultaneous conflict and stitch minimization, ILP is adopted in [2,3] with different feature pre-slicing techniques. Xu et al. [10] propose a matching based decomposer that handles the same optimization problem as [2,3].

*This work is supported in part by NSF, SRC, and equipment donations from Intel.

There are also several DPL-aware optimization works from design side. Cho et al. [11] propose a correct-by-construction DPL-friendly routing with built-in layout decomposer. The idea is extended by [12] with enhancement of lazy evaluation and with-in net optimization. In [13], the DPL awareness and redundant via insertion are considered together during routing. Hsu at al. [14] propose a simultaneous layout migration and decomposition for standard cell design, which aims to minimize stitch number and layout area together. Because the spacing between the features are considered dynamically during coloring, their approach suffers from run time overhead, which is not suitable for large-scale layout modification.

In this paper, we present WIre Spreading enhanced Decomposition of Masks in double patterning lithography (WISDOM). The chip area is fixed in our work. After initial Decomposition Graph (DG) construction, we create a set of wire spreading candidates. The DG is updated then to model layout decomposition problem together with these potential WSCs. Our main technical contributions are two-fold: first, we develop an integer linear programming formulation from DG to simultaneously minimize the number of conflicts, stitches and amount of layout perturbation; three acceleration methods are further proposed without losing solution quality. The experimental results are promising and show the effectiveness and efficiency of our WISDOM methodology.

2. PRELIMINARY AND FORMULATION

2.1 Wire Spreading for Decomposability

Our key idea of performing wire spreading for decomposition of masks is to push layout segments away for more flexible coloring. This helps reducing the number of conflicts and stitches.

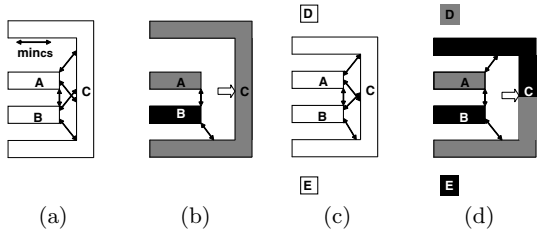
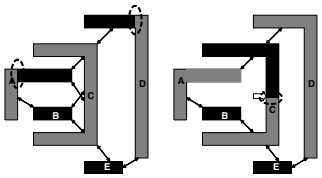


Figure 1: Wire spreading to eliminate conflict. Assume only feature C can be moved.

Given a routed design, wire spreading can be used for eliminating unresolvable coloring conflicts. Fig. 1 (a) shows a three-way conflict cycle between features A-B-C, where any two of them are within minimum coloring space. Moreover, no stitches can be inserted while satisfying minimum overlapping margin requirement. In consequence, there is no way to produce conflict-free solution by mask decomposition algorithm alone. However, if part of feature C is spread to break the coloring constraint between A and C as Fig. 1 (b), we can easily resolve this problem. For more restricted situation, as Fig. 1 (c) illustrates, it is not possible to completely push C away from feature A or B beyond min_{cs}, due to surrounding fixed layout objects. In such case, we could still spread polygon C for creating a legal splitting location, as indicated by Fig. 1 (d).



(a) (b)

Figure 2: Wire spreading to reduce stitch. Assume only feature C can be moved.

It is also possible to spread wire to reduce the number of stitches. For the example in Fig. 2 (a), initially, feature C is not splittable, and two stitches are required on A and D for resolving conflicts. If we spread the route C as Fig. 2 (b), only one stitch is needed.

2.2 Problem Formulation

As it can be seen from Section 2.1, with help of wire spreading, the solution space of DPL decomposition can be extraordinarily high. On the other hand, as nanometer designs are mostly grid-based, that restricts wire spreading to its nearby discrete routing grids. Therefore, we will preprocess the initial decomposition graph and generate a library of Wire Spreading Candidates (WSC)s to improve decomposability (more detailed description to be presented in Section 3.2). Mask assignment is performed together with these candidates as options. As examples, Fig. 1 (b)/(d) and Fig. 2 (a) are simple WSCs for Fig. 1 (a)/(c) and Fig. 2 (f), respectively.

With this library of WSCs, our optimization problem is formally defined as follows:

Problem Formulation: Given a layout, perform mask decomposition with pre-computed WSCs as design modification options. The goal is to minimize the number of conflicts and stitches, while introducing as less layout perturbation as possible.

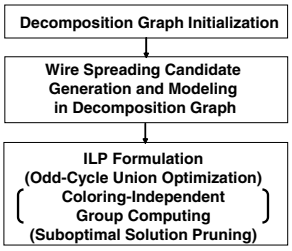


Figure 3: The overview of WISDOM

3. BASIC ALGORITHMS FOR WISDOM

In following two sections, we will present our WISDOM algorithm. The entire flow is shown in Figure 3.

3.1 Decomposition Graph Initialization

For simplification purpose, we adopt a flow similar to [2] to construct an initial DG for modeling mask decomposition problem. Other approaches as in [3, 4] are also flexible to apply. There are two kinds of edges: *Conflict Edge* (**CE**) and *Stitch Edge* (**SE**). If and only if two nodes(polygons) are connected by CE/SE and in same/different masks, it results in a conflict/stitch.

The key steps of our construction method are briefly reviewed with the help of Fig. 4. Fig. 4 (a) shows the irregular polygons for original layout. If two polygons are within min_{cs}, there is a CE between them, marked by a dash line in Fig. 4 (b). The node projection, proposed in [1], is then performed, where projected segments are highlighted by bold curves of Fig. 4 (c). Based on projection result, all the legal splitting locations are computed next. The corresponding rectangles are split with SE added, which updates DG as Fig. 4 (d).

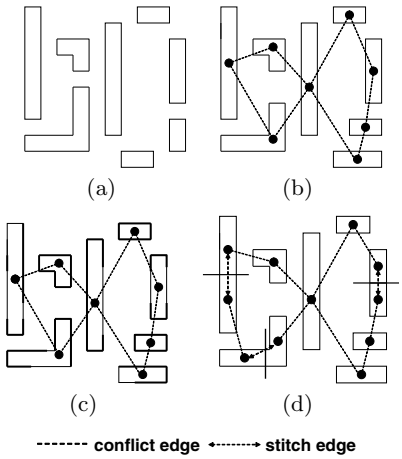


Figure 4: Initial decomposition graph construction

3.2 WSC Generation and Modeling

As the next step, we will generate a set of DPL-friendly WSCs, and model them in initial DG. There are two new types of edges introduced in the decomposition graph : *Conflict Elimination Edge* (**CEE**) and *Splitting Creation Edge* (**SCE**).

Each desired layout modification corresponds to a WSC, which will be constructed sequentially and independently based on original design. A *search region* is defined to avoid changing design too much. All the moved segments must be completely within their corresponding search regions, and the connectivity should be maintained. We also allow ripple movement of multiple wires. No design rules should be violated, and timing critical nets/vias are fixed. If the distance of two features in the original layout is larger than min_{cs}, this relationship should also hold after certain WSC is applied. This ensures no new coloring constraints are introduced in terms of double patterning lithography. It should be noted that, other user-defined conditions can be easily incorporated.

In the following, we present the key steps for WSC generation and modeling.

3.2.1 Spreading to Eliminate Conflict Edges

First, for each CE, we try to find a WSC, moving apart associated polygons beyond min_{cs}. This relaxes layout coloring constraints, since these two features can be assigned to the same mask consequently.

If such a WSC is available, we will change the status of the corresponding conflict edge to Conflict Elimination Edge (CEE). For the original layout in Fig. 5 (a), suppose there is a WSC like Fig. 5 (b) eliminating its conflict edge i-k, the DG is transformed to Fig. 5 (c).

During decomposition, if two nodes are connected by a conflict elimination edge and assigned into same masks, there is a conflict. However, different from conflict edge, if the corresponding WSC of this CEE is applied, this conflict can be removed.

3.2.2 Spreading to Create Splitting Locations

Then, we also detect these WSCs, which can create new potential stitch locations on initially unsplittable polygons. This provides larger solution space for double patterning mask decomposition.

As illustrated by the same example of Fig. 5 (a), suppose a new splitting location can be created on k by modifying layout

as Fig. 5 (d). To model this WSC, polygon k is first split into two touching features k_i and k_j. An Splitting Creation Edge (SCE) is then added between k_i and k_j to indicate that this new potential stitch location results from wire spreading. The conflict edges between (i,k) and (j,k) are replaced by (i,k_i) and (j,k_j) respectively. Other edges connecting to k will be directed to either k_i or k_j, such as *o*-k_j in Fig. 5 (e).

During coloring, the two nodes linked by splitting creating edge should be in same color by default. Only if its corresponding WSC is applied, a stitch can be introduced and they can be assigned into different masks.

3.2.3 Non-compatible WSCs

Since we generate each WSC independently, they may not be applied in the same time. Two WSCs are *non-compatible*, if resulting in any of following problems:

1. Design rule violation or new conflict edge is introduced.
2. The same polygon is modified in distinct way.

Fig. 5 (f) illustrates the second case, with both WSCs in Fig. 5 (b) and (d) modeled. Obviously, these WSCs change polygon k differently, and hence only one can be picked.

After finding out all the WSCs, we will create a list of non-compatible WSC pairs based on above definitions.

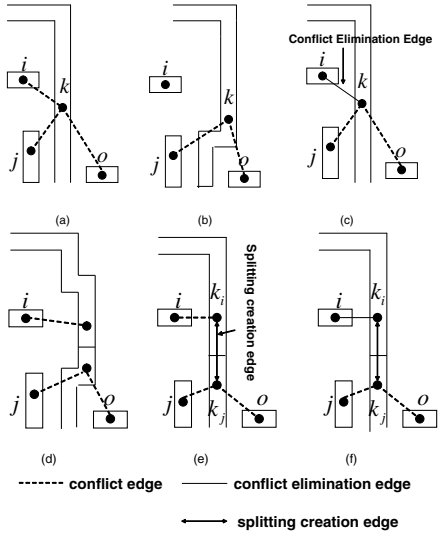


Figure 5: Wire spreading candidate modeling and decomposition graph updating.

3.3 ILP Formulation

To achieve good trade-off among conflict, stitch and layout perturbation minimization, in this section, we will formulate an integer linear programming to perform simultaneous mask decomposition and modification. The set of pre defined WSCs are the only available design pertubation configurations. Our ILP is different from [1, 3], because their formulations are not considering layout modification for decomposability improvement and simultaneous optimization. To better present, some notations are first listed in Table 1.

The co-optimization problem can be formulated as follows:

$$min(\alpha \sum_{e_{ij}\in E} c_{ij} + \epsilon \sum_{t_{ij}\in T} s_{ij} + \sum_{e_{ij}\in E^{cee}} m_{ij}^{cee} p_{ij}^{cee} + \sum_{e_{ij}\in E^{sce}} m_{ij}^{sce} p_{ij}^{sce}) \quad (1)$$

subject to

$$x_i + x_j <= 1 + c_{ij} \quad \forall e_{ij} \in E^{ce} \quad (2)$$

$$(1-x_i) + (1-x_j) <= 1 + c_{ij} \quad \forall e_{ij} \in E^{ce} \quad (3)$$

$$x_i + x_j <= 1 + c_{ij} + m_{ij}^{cee} \quad \forall e_{ij} \in E^{cee} \quad (4)$$

Table 1: Notation

r_i	The i_{th} layout polygons
x_i	binary variable denoting the coloring of r_i $x_i = 0$ if the color is GRAY, otherwise it is BLACK
e_{ij}	a conflict edge, a conflict elimination edge or a splitting creation edge between r_i and r_j
t_{ij}	r_i and r_j are touching each other, connected by a stitch edge or splitting creating edge
E^{ce}	the set of conflict edges
E^{cee}	the set of conflict elimination edges
E^{sce}	the set of splitting creation edges
E	the set of e_{ij}
T	the set of t_{ij}
c_{ij}	binary variable c_{ij} =1 when there is a conflict between r_i and r_j
s_{ij}	binary variable s_{ij} =1 when there is a stitch between r_i and r_j
m_{ij}^{cee}	binary variable $m_{ij}^{cee} = 1$ if and only if WSC for $e_{ij} \in E^{cee}$ is applied
m_{ij}^{sce}	binary variable $m_{ij}^{sce} = 1$ if and only if WSC for $e_{ij} \in E^{sce}$ is applied
p_{ij}^{cee}	layout modification cost, when m_{ij}^{cee} =1
p_{ij}^{sce}	layout modification cost, when m_{ij}^{sce} =1
$y_{ij,mn}^{cc}$	$(i,j) \neq (m,n)$. WSCs for $e_{ij} \in E^{cee}$ and $e_{mn} \in E^{cee}$ are not compatible.
Y^{cc}	the set of $y_{ij,mn}^{cc}$
$y_{ij,mn}^{ss}$	$(i,j) \neq (m,n)$. WSCs for $e_{ij} \in E^{sce}$ and $e_{mn} \in E^{sce}$ are not compatible.
Y^{ss}	the set of $y_{ij,mn}^{ss}$
$y_{ij,mn}^{cs}$	$(i,j) \neq (m,n)$. WSCs for $e_{ij} \in E^{cee}$ and $e_{mn} \in E^{sce}$ are not compatible.
Y^{cs}	the set of $y_{ij,mn}^{cs}$

$$(1-x_i) + (1-x_j) <= 1 + c_{ij} + m_{ij}^{cee} \quad \forall e_{ij} \in E^{cee} \quad (5)$$

$$x_i + (1-x_j) <= 1 + s_{ij} \quad \forall t_{ij} \in T \quad (6)$$

$$(1-x_i) + x_j <= 1 + s_{ij} \quad \forall t_{ij} \in T \quad (7)$$

$$m_{ij}^{sce} = s_{ij} \quad \forall e_{ij} \in E^{sce} \quad (8)$$

$$m_{ij}^{cee} + m_{mn}^{cee} <= 1 \quad \forall y_{ij,mn}^{cc} \in Y^{cc} \quad (9)$$

$$m_{ij}^{sce} + m_{mn}^{sce} <= 1 \quad \forall y_{ij,mn}^{ss} \in Y^{ss} \quad (10)$$

$$m_{ij}^{cee} + m_{mn}^{sce} <= 1 \quad \forall y_{ij,mn}^{cs} \in Y^{cs} \quad (11)$$

The objective function (1) is to minimize the weighted summation of conflicts and stitches as well as layout perturbation. The weights of α and ϵ are user-defined parameters, for assigning relative importance between these matrices. The layout penalty costs p_{ij}^{cee} and p_{ij}^{sce} are associated with respective conflict elimination and splitting creation edge.

Constraints (2)-(3) identify a conflict if two features connected by a conflict edge are in the same color. Constraints (4)-(5) are applied for conflict elimination edge. If m_{ij}^{cee} is one, the corresponding WSC is applied. As a result, the two polygons connected by e_{ij} are moved beyond min_{cs}, and they can be in the same mask without introducing conflicts. In such case, variable c_{ij} is always zero forced by the objective function. On the other side, when m_{ij}^{cee} is zero, its corresponding WSC is not used. Conflict will be detected based on the same logistic for the case of conflict edge as Constraints (2)-(3).

Constraints (6)-(7) are used to identify a stitch if two touching rectangles are colored differently. Constraint (8) follows the fact that any splitting creation edge e_{ij}^{sce} connects two touching rectangles t_{ij}. If and only if the stitch s_{ij} for t_{ij} is one, the corresponding wire spreading candidate is applied. Constraints (9)-(11) serve for the same purpose. If two WSCs are not compatible, at most one will be picked.

4. WISDOM SPEEDUP TECHNIQUES

Since the time complexity of solving ILP is quite high in general, in this section, we propose three reduction techniques to simplify the decomposition graph without losing optimality.

Definition 1 *odd/even cycle (OC/EC):* a cycle, whose total number of conflict edges and conflict elimination edges is odd/even.

4.1 Odd-Cycle Union Optimization

Naively, ILP formulation would be performed on the entire decomposition graph. In this section, we will show that, it is sufficient to conduct ILP only on a subgraph of the DG, which is the union of all the odd-cycles. It will not lose optimality.

The overall flow is shown in Fig. 6. Since we are working on a much smaller graph, the CPU time of ILP solving is well reduced. Moreover, it only takes polynomial time to perform preprocessing and postprocessing steps, which details are discussed in Section 4.1.1 and 4.1.2 respectively. Therefore, by taking the flow of Fig. 6, the coloring assignment can be effectively accelerated.

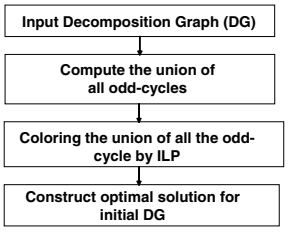


Figure 6: The flow of odd-cycle union optimization

4.1.1 Computing the union of all the odd-cycles

The naive calculation is to enumerate all the odd cycles and then find their union. This would be expensive since the number of OCs grows exponentially with respect to the size of DG. Our key idea is to make use of the concept of *cycle basis* and compute this union in a polynomial time. We do not have to dig out each individual OC.

Definition 2 *cycle basis* and *base cycle:* Given a Decomposition Graph (DG), a subset of its cycles are called cycle basis, with each one called base cycle, if any cycle in DG can be generated by performing XOR operation on the cycles in this subset.

Definition 3 XOR ($\oplus$) operation on two graphs is denoted as $(G_1 \oplus G_2)$, which is the union of these graphs, minus their common edges. XOR ($\oplus$) operation on more than two graphs is computed as $(((G_1 \oplus G_2) \oplus G_3) ... \oplus G_i)$.

As Fig. 7 (a) illustrates, there are three cycles. A-B-C-A and D-B-C-D could be a cycle basis, since the third loop A-B-D-C-A can be obtained by taking $\oplus$ on their edges.

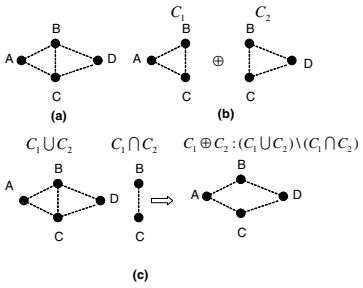


Figure 7: The concept of cycle basis

Our efficient computation for odd-cycle union is given in Algorithm 1, with timing complexity of $O(N^3)$ time. Due to page limit, the detailed proof and analysis are skipped here.

4.1.2 Optimal solution construction for decomposition graph

After solving the union of all the odd-cycles by ILP, one *optimal* coloring assignment needs to be constructed for original

Algorithm 1

Require: A decomposition graph, DG
Ensure: the union of all the odd-cycles: ODG
1: $OCB = \emptyset$
2: use depth-first search to calculate a cycle basis of DG: CB
//Line 3-11 compute one cycle basis for ODG.
3: move all the odd base cycles in CB into OCB
// now all the base cycles in CB are even cycles.
4: **while** OCB and CB share some common edges **do**
5: Make these common edges as COMMON
6: **for** any even base cycle EC in CB **do**
7: **if** EC have contain at least one COMMON edges **then**
8: move this EC from CB to OCB
9: **end if**
10: **end for**
11: **end while**
// OCB now is one cycle basis of the union of all the odd-cycles.
12: output the union of all the base cycles in OCB as ODG

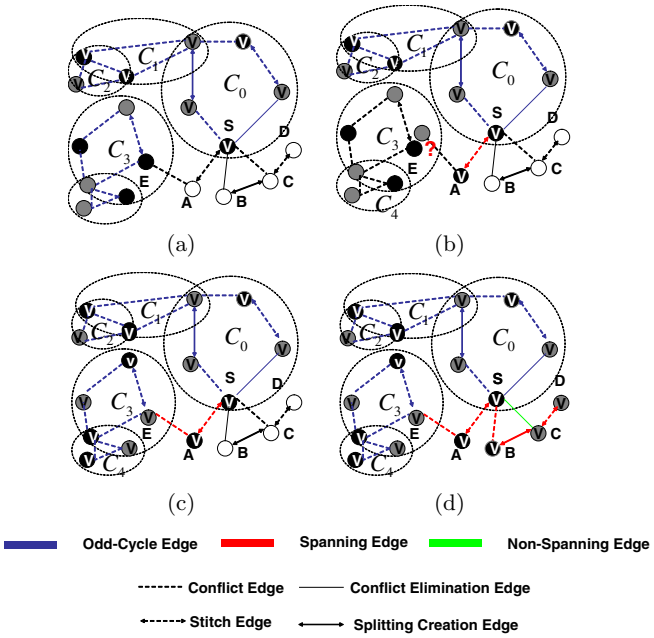


Figure 8: The procedure of solution construction for original decomposition graph. The symbol v inside each node denotes the state of visited.

decomposition graph. Strictly speaking, *this constructed solution should have the same cost as the ILP assignment of DG, weighted by Objective (1).* The detailed steps are shown in Procedure 1, whose complexity is linear proportional to the size of graph. Its validity is guaranteed by Theorem 1.

Procedure 1:

Step1: As Fig. 8 (a) illustrates, given a DG, assume we have obtained a solution for the union of its odd-cycles C_0-C_4 by ILP, while other part of DG remains uncolored. All the edges in this union are marked as **Odd-Cycle-Edges (OCE)**.

Step2: Compute connected components for this odd-cycle union by depth first search, where no edge or vertices shared between different clusters. For example, there are two such components in Fig. 8 (a): (C_0, C_1, C_2) and (C_3, C_4).

Step3: Randomly pick one connected component, as (C_0, C_1, C_2) in Fig. 8 (a), and mark its nodes as visited sources. Then, start from these initial sources to traversal DG in a **depth-first manner** for exploring and colorings unvisited vertices. Each search phase consists in two steps: Step4 and Step5.

Step4: Given a current already-visited node s, we will explore its neighbors. If one of its edges leads to unvisited vertex t, we will mark edge s-t as **Spanning Edges(SPE)**. The mask of t

will be assigned based on following *coloring propagation rules*:

coloring propagation rule: if the type of edge s-t is a conflict edge or conflict elimination edge, we will assign the opposite color of s to t; otherwise, s and t will be on the same mask.

This rule ensures neither conflict/stitch nor layout modification will be introduced when a non-visited node is reached and colored through a SPE. As shown in Figure. 8 (b), the coloring of A can be determined as BLACK by propagating the coloring of feature S through a stitch edge.

Step5: We will further check whether t is contained by some unvisited connected component, computed in *Step2*. If so, we will further assign the coloring of this entire component in one time, using its existing ILP solution. However, there might be a *coloring inconsistence problem*. As illustrated by Figure. 8 (b), after we propagate the solution of A, the color of E would be GRAY. However, E also belongs to a unvisited connected component (C_3, C_4), and it is assigned BLACK by ILP. Under such case, we can simply flip the existing ILP solution of this entire component (C_3, C_4) for maintaining coherence, shown by Figure. 8 (c).

Step6: The newly found t and CC will be marked as visited, and *Step* 4&5 will be recursively called on these nodes. □

Above steps can be repeated until all the nodes have been visited, as Figure. 8 (d) shows. All the OCEs and SPEs form a **depth first forest**. The edges which are in DG but not belonging to this forest are called **Non-Spanning Edges (NSE)**. When we explore the neighbors of an visited node s, these NSEs connect to other already-visited vertices. Edge C-S is an example of NSE. When we find node S from C, S has already been processed and colored.

The correctness of above procedure is stated as following theorem. The detailed proof is skipped here.

Theorem 1 The solution generated by Procedure 1 is an optimal solution to the ILP formulation in (1) for the original decomposition graph .

4.2 Coloring-Independent Groups

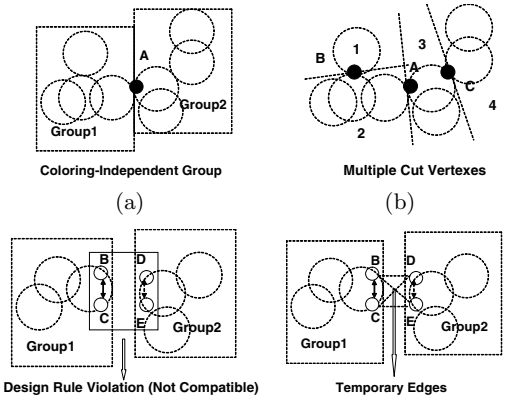


Figure 9: Figures (a) and (b) explain the concept of coloring-independent groups. Figures (c) and (d) explain how to handle wire spreading candidate.

After finding the union of all the odd-cycles, we compute graphically-disjointed connected components as [1–3] for improving scalability. Furthermore, we observe that, in terms of mask decomposition problem, each connected component may still be divided into several coloring-independent groups. This reduces ILP problem size to a greater extent.

Fig. 9 (a) shows a simple motivational example, which is a connected component. We observe that, node A is the only common vertex between group one and two, and these two groups can still be solved by ILP individually while maintaining optimality. The reason is that, after their respective optimal solutions are obtained, if the coloring of feature A from both groups are different, we could simply flip the solutions of one group without effecting overall solution quality.

In graph theory, such node as A is called cut vertex or articulation point, whose removal creates disconnected coloring-independent groups, which are also observed as biconnected components in [15]. Generally, if there are multiple cut vertices, the initial graph can be decomposed into a chain of coloring-independent groups linked by these articulation points. As Fig. 9 (b) illustrates, four groups 1-2-3-4 are connected sequentially with cut vertices A-C as boundary nodes. Similar to the simple case of Fig. 9 (a), we can solve each group individually without losing optimality. Their solutions can be merged by appropriately flipping the coloring of certain groups.

With wire spreading candidate in mind, the decomposition graph would need to be modified temporarily for correctly computing either connected component or coloring-independent groups. As shown in Fig. 9 (c), group1 and group2 are graphically disjointed. However, design rule is violated, when their respective WSCs for CEE B-C and SCE D-E are applied simultaneously. If we decompose these two components/groups separately, both WSCs may be picked in the same time, generating unfeasible solution. Therefore, to resolve this issue, for each pair of non-compatible WSCs, we add some temporary edges between their associated nodes as Fig. 9 (d) illustrates. Connected components or coloring-independent groups will be calculated for this temporarily-updated graph.

4.3 Suboptimal Solution Pruning

Given any coloring-independent group, we can further simplify it by performing solution pruning for underlying substructures, *sequential path*, defined as follows:

Definition 4 *sequential path (SP):* an acyclic linked list of nodes is a sequential path, if

- Except two ending features, all the nodes must have a degree of two.
- Any WSC associated with this list and any other WSC outside are compatible.
- This list can not be totally included in another list which satisfy first two conditions.

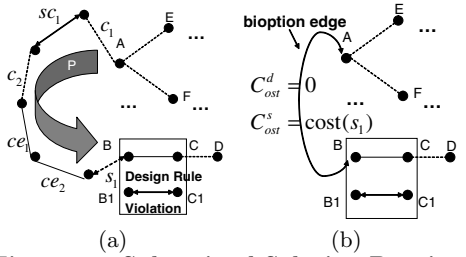


Figure 10: Suboptimal Solution Pruning.

Based on the above definition, in Fig 10 (a), Path P is a sequential path with node A and B as ending features. It is composed of two CEs c_1-c_2, one SE s_1, one SCE sc_1 and two CEEs ce_1-ce_2. Ending node A can not be extended to E or F, because in that case, A becomes a internal vertex but has a degree of three. This violates condition 1. Edge B-C can not be further included in this SP as well since it is not compatible with splitting creating edge B1-C1, which is outside of path P. Condition 2 does not hold, then.

The nice property of SP is that, besides two ending nodes, each sequential path will not have coloring or design modification interaction with other parts of this graph. In other words,

given coloring configurations of two ending nodes, the best decomposition of a SP can be uniquely determined.

There are only two possible configurations for each SP, depending on whether head and tail features have same color. As Fig 10 (b) shows, since there are four CEs/CEEs, when A and B are assigned into same mask, no stitches or layout modifications are needed for zero conflict solution. As a result, the best ILP penalty C^s_{ost} for path P is zero. On the other hand, if A and B are assigned into different masks, one of s_1, sc_1 and ce_1-ce_2 should be applied to resolve potential conflict. Assume the cost of s_1 is the smallest, we pick it as a local optima C^d_{ost}.

Therefore, graphically, we can replace this whole SP by a bioption edge, which only stores possible optimal costs, C^s_{ost} and C^d_{ost}. While formulating ILP problem, we simply apply following four equations to check whether node A and B are in same/different masks, where binary variable d_{ij} is zero/one.

$$x_A + (1 - x_B) <= 1 + d_{ij} \quad \forall e^b_{ij} \quad bioption \quad edge \tag{12}$$

$$(1 - x_A) + x_B <= 1 + d_{ij} \quad \forall e^b_{ij} \tag{13}$$

$$x_A + x_B <= 2 - d_{ij} \quad \forall e^b_{ij} \tag{14}$$

$$(1 - x_A) + (1 - x_B) <= 2 - d_{ij} \quad \forall e^b_{ij} \tag{15}$$

A expression of $C^s_{ost}(1 - d_{i,j}) + C^d_{ost}d_{i,j}$ will be added into Objective (1) to take into account corresponding optimal ILP penalty for the represented sequential path. Comparatively, in original ILP, for sequential path P, we need to introduce one additional binary variable for each internal vertex. Moreover, for every edge, at least two constraints out of Constraints(2)-(8) should be specified. As a result, by conducting suboptimal solution pruning, it reduces number of variables and constraints.

In essence, this pruning technique shares its spirit with the dimension reduction technique presented by [16], in which the iso-cap property in multi-core processor design is discovered and employed to make a complex problem feasible.

5. EXPERIMENTAL RESULTS

We implement our algorithm in C++ and test on Intel Core 3.0GHz Linux machine with 32G RAM. OpenAcess2.2 [17] is used for interfacing with GDSII directly. Moreover, we choose glpk [18] as our solver for integer linear programming. ISCAS-85&89 benchmarks are scaled down and modified as our test cases. The metal one layer is used for experimental purpose, because it is one of the most trouble some layers in terms of double patterning lithography. The minimum width and spacing become 40nm. The minimum coloring space for double patterning is set as 65nm, and minimum overlapping margin for stitch insertion is 10nm.

5.1 Statistics on Decomposition Graph

The detailed statistics of constructed decomposition graphs are shown in Table 2. The first column denotes circuit name. Columns "#ce" and "#se" under "initial DG" are the total number of conflict edges and stitch edges in initial decomposition graph. Columns "#cee" and "#sce" under "updated DG" show the respective number of conflict elimination and splitting creation edges, added in WSC generation and modeling step. "#cee" plus "#sce" equal to the total number of WSCs "#WSC". "total" is the summation number of all the test cases, and "ratio" is computed percentage of corresponding metrics.

From Table 2, we have WSCs which can eliminate 8% conflict edge and create 9% more stitch candidates. Although these percentages seem relatively small, however, since DPL layout decomposition has a ripple effect, it could remove more than one conflicts or stitches by just applying one WSC. On the other side, the increased graph size due to "#cee" and "#sce" would degrade the performance of ILP. As we show later, with proposed graph reduction techniques, this side effect is well encountered.

Table 2: Statistics on decomposition graph.

circuit	initial DG		updated DG		
	#ce	#se	#cee	#sce	#WSC
C432	1063	964	36	14	50
C499	2428	1437	78	88	166
C880	2464	2439	177	196	373
C1355	3101	3768	74	104	178
C1908	5109	5648	262	96	358
C2670	8750	8655	596	420	1016
C3540	10896	10864	850	768	1618
C5315	16049	15654	1112	670	1782
C6288	13389	11014	264	530	794
C7552	22516	23525	1453	1122	2575
S1488	5273	4284	499	428	927
S38417	69270	57204	6302	2908	9210
S35932	86540	58661	8553	7634	16187
S38584	170079	7140	14191	7764	21955
S15850	169147	124969	12422	8920	21342
total	586074	336226	46869	31662	78531
avg	1	1	0.08	0.09	-

Table 3: Result Comparison

	[2]				WISDOM			
circuit	$cflt$	$stitch$	$WL(e^5)$	CPU	$cflt$	$stitch$	$WL(e^5)$	CPU
C432	55	11	2.781	0.27	48	14	2.784	0.09
C499	258	11	5.792	0.74	214	11	5.809	0.27
C880	125	105	2.920	0.62	32	83	2.925	0.2
C1355	82	89	86.790	0.66	31	102	86.810	0.32
C1908	99	346	14.440	1.91	55	343	14.462	0.4
C2670	254	749	23.730	3.13	49	655	23.857	1.08
C3540	472	643	30.162	3.38	67	619	30.350	0.73
C5315	413	1234	43.700	3.6	89	949	43.967	1.1
C6288	912	331	35.240	5.78	663	340	35.340	0.76
C7552	708	1544	62.300	4.5	166	1338	62.303	1.6
S1488	274	316	14.300	2.01	60	134	14.372	0.44
S38417	3866	868	184.000	24.88	2518	471	184.552	5.51
S35932	11731	1383	407.400	203.24	7006	875	409.240	14.22
S38584	11254	948	443.000	127.75	6635	1139	444.580	11.67
S15850	11579	3392	431.200	66.15	7198	2103	433.780	12.26
total	42082	11970	1787	448.62	24831	9176	1795	50.6
ratio	1	1	1	1	0.59	0.77	1.004	0.11

5.2 Result Comparison

For comparison, we implement a post-routing mask decomposition algorithm [2], which is the extension of [1]. The metric *unresolvable conflict edge* they applied is actually consistent with our definition of conflict, which is described from another point of view. This algorithm is performed directly on the initial DG, which is obtained in the first step of our flow.

We can not compare the work of [3, 10] directly. In [3], they work on extremely fine metrics, conflict grid. In [10], their algorithm is designed planar graph only, while our decomposition graph is not. We are also not able to compare with another DPL-driven post-routing layout modification work [14], because our targeted problem is different. In [14], they focus on technology migration for stand-cell library only, and area minimization is one of their objectives. We work on full-chip design perturbation, and the chip size is fixed.

Table 3 lists the comparison of decomposition results, where "cflt" and "stitch" are the number of conflicts and stitches in the colored layout. The column "WL" shows the wire length in terms of "nm". CPU is the computational time in terms of "second", including both graph construction and ILP solving steps.

As we can see, our algorithm significantly outperforms [2] in terms of quality, which generates a solution with 41% and 23% reduction on conflict and stitch number respectively. The layout perturbation ratio is only 0.4%. With respect to runtime, we achieve 9X speed-up. For some benchmark, such as c432, WISDOM does produce more stitches. The reason is that in our experimental setting, conflict elimination is set as higher priority job over splitting reduction. The number of stitches could increase comparatively, to better remove the conflicts.

Table 4: The effectiveness of various acceleration methods

circuit	CPU(base)	odd-cycle union optimization					+c-independent			+suboptimal solution pruning				
		#nodes		#edges		CPU	#CC	#CG	CPU	#var		#con		CPU
		w/o	w	w/o	w		w/o	w		w/o	w	w/o	w	
C432	0.35	2350	715	3398	798	0.26	127	138	0.22	1596	792	1502	914	0.09
C499	1.2	3996	1539	7485	1926	0.78	164	180	0.55	3854	1933	3449	2013	0.27
C880	0.64	5525	2195	6562	2428	0.57	293	311	0.64	4864	1904	4604	2254	0.2
C1355	0.87	7893	3084	9633	7893	0.67	551	585	0.82	6482	2796	6291	3449	0.32
C1908	2.26	12122	4709	15321	12122	1.35	635	699	1.52	10366	4578	9828	5497	0.4
C2670	2.96	18784	8788	23875	18784	2.05	916	977	2.93	20078	8849	18765	10229	1.08
C3540	4.53	23878	10108	29925	23878	2.61	1196	1227	2.77	22784	9092	21469	10603	0.73
C5315	3.91	34556	15245	44201	17320	2.92	1779	1920	2.18	34650	14567	32424	17056	1.1
C6288	5.82	29192	8460	40968	9181	3.03	1266	1668	4.89	18368	8006	17239	9486	0.76
C7552	5.22	50650	21565	62801	24029	4.14	2716	2869	1.27	48066	20061	45441	23769	1.6
S1488	1.38	10538	4752	9771	5430	1.2	585	622	1.38	100680	4046	10050	4686	0.44
S38417	27.54	141535	47592	127928	52660	14.67	5791	6989	12.53	105342	44790	99055	5222	5.51
S35932	223.56	311931	139035	149018	152925	57.51	17123	20960	29.58	305906	131384	288123	147123	14.22
S38584	129.85	338771	117352	181101	131147	49.46	13241	16605	27.97	262388	114148	245135	126849	11.67
S15850	83.32	326454	132463	298576	147940	41.93	14960	18207	31.73	295942	123066	277157	137241	12.26
total	493.41	1318175	517602	1010563	608461	183.15	61343	73957	120.98	1241366	490012	1080532	506391	50.65
ratio	1	1	0.39	1	0.60	0.37	1	1.20	0.24	1	0.39	1	0.47	0.1

Although after WISDOM, there are several conflicts and stitches remaining in the design, we observe they are mainly resulting from the pins/vias in standard cells and highly-congested routing paths. While we search WSCs, most of these features are set as fixed for avoiding modifying design and timing too much. Thus, our WISDOM can be used in combination with high-level DPL-friendly design methodologies [11–14].

5.3 Efficiency

We further study the effectiveness of various acceleration techniques in Table 4. "CPU(base)" shows the runtime of our algorithm without proposed speed-up approaches, in terms of "seconds". For fair comparison, the layout partition technique in [2] is applied in this baseline. Then, we add in each acceleration technique incrementally. In all the columns, "w/o" and "w" show the respective data without and with certain method, and "CPU" is the resulting runtime after it is adopted. We double check simulation results, and ensure that no solution quality is lost by applying these graph reduction approaches.

The columns under "odd-cycle union optimization" show the graph statistics of initial DG and its odd-cycle union. The number of nodes and edges are reduced by 61% and 40% respectively in average. This reduction is significant in terms of ILP solving, which has exponential complexity with respective to problem size. As seen from Table 4, we achieve 2.7x speed-up.

The effectiveness of coloring-dependent group computing is investigated then with results listed under "+c-independent". "#CC" is the number of connected components, computed by the layout partition technique in [2]. By using cut vertex, we further divide each component to multiple coloring-independent groups, which can also be solved by ILP individually. The total number of such groups is shown in "#CG", which is averagely 20% more than "#CC". With this technique, the ILP problem size becomes smaller, and the runtime is further reduced by 37%.

Last, we apply the technique of suboptimal solution pruning, where the results are listed under "+suboptimal solution pruning". "#var" and "#con" are the total number of variables and constraints in the ILP formulations. As we can see, solution pruning technique can reduce their number by 61% and 53%, respectively. Therefore, the coloring assignment can be effectively accelerated by another 2.4X.

6. CONCLUSION

In this paper, we have developed a wire spreading enhanced decomposition of masks algorithm for double patterning lithography. Our approach is featured by integer linear programming and efficient graph reduction techniques. The experimental results are very promising.

7. REFERENCES

[1] 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.

[2] Andrew B. Kahng, Chul-Hong Park, Xu Xu, and Hailong Yao. Revisiting the layout decomposition problem for double patterning lithography. In *Proc. of SPIE*, October 2008.

[3] Kun Yuan, Jae-Seok Yang, and David Z. Pan. Double patterning layout decomposition for simultaneous conflict and stitch minimization. In *Proc. Int. Symp. on Physical Design*, March 2009.

[4] Yue Xu and Chris Chu. GREMA: Graph Reduction based Efficient Mask Assignment for double patterning lithography. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2009.

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

[6] Y. Borodovsky. Lithography 2009 overview of opportunities. In *Semicon West*, 2009.

[7] G. Bailey, A. Tritchkov, J. Park, L. Hong, V. Wiaux, E. Hendrickx, S. Verhaegen, P. Xie, and J. Versluijs. Double pattern EDA solutions for 32nm HP and beyond. In *Proc. of SPIE*, volume 6521, 2007.

[8] 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.

[9] Jae-Seok Yang, Katria Lu, Minsik Cho, Kun Yuan, and David Z. Pan. A New Graph-theoretic, Multi-objective Layout Decomposition Framework for Double Patterning Lithography. In *Proc. Asia and South Pacific Design Automation Conf.*, Janunary 2010.

[10] Yue Xu and Chris Chu. A Matching Based Decomposer for Double Patterning Lithography. In *Proc. Int. Symp. on Physical Design*, March 2010.

[11] Minsik Cho, Yongchan Ban, and David Z. Pan. Double patterning technology friendly detailed routing. In *Proc. Int. Conf. on Computer Aided Design*, October 2008.

[12] Xin Gao and Luca Macchiarulo. Enhancing Double-Patterning Detailed Routing With Lazy Coloring and Within-Path Conflict Avoidance. In *Proc. Design, Automation and Test in Eurpoe*, March 2010.

[13] Kun Yuan, Katrina Lu, and David Z. Pan. Double patterning lithography friendly detailed routing with redundant via consideration. In *Proc. Design Automation Conf.*, July 2009.

[14] Chin-Hsiung Hsu, Yao-Wen Chang, and Sani R. Nassif. Simultanous layout migration and decomposition for double patterning technology. In *Proc. Int. Conf. on Computer Aided Design*, November 2009.

[15] Piotr Bermany, Andrew B. Kahng, Devendra Vidhani, Huijuan Wang, and Alexander Zelikovsky. Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(2):175–187, Feb 2000.

[16] Yifang Liu, Weiping Shi, and Jiang Hu. Buffering Interconnect for Multicore Processor Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(12):2183–2196, December 2008.

[17] http://www.si2.org/?page=69.

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