A New Graph-Theoretic, Multi-Objective Layout Decomposition Framework for Double Patterning Lithography

Jae-Seok Yang, Katrina Lu*, Minsik Cho†, Kun Yuan, and David Z. Pan
Dept. of ECE, The University of Texas at Austin, Austin, TX USA
*Intel Corporation, Hillsboro, OR USA
†IBM T. J. Watson Research Center, Yorktown Heights, NY USA
jsyang@cerc.utexas.edu, katrina.y.lu@intel.com, minsikcho@us.ibm.com
kyuan@cerc.utexas.edu, dpan@ece.utexas.edu

ABSTRACT

As Double Patterning Lithography(DPL) becomes the leading candidate for sub-30nm lithography process, we need a fast and lithography friendly decomposition framework. In this paper, we propose a multi-objective min-cut based decomposition framework for stitch minimization, balanced density, and overlay compensation, simultaneously. The key challenge of DPL is to accomplish high quality decomposition for large-scale layouts under reasonable runtime with the following objectives: a) the number of stitches is minimized, b) the balance between two decomposed layers is maximized for further enhanced patterning, c) the impact of overlay on coupling capacitance is reduced for less timing variation. We use a graph theoretic algorithm for minimum stitch insertion and balanced density. An additional decomposition constraints for self-overlay compensation are obtained by integer linear programming(ILP). With the constraints, global decomposition is executed by our modified FM graph partitioning algorithm. Experimental results show that the proposed framework is highly scalable and fast: we can decompose all 15 benchmark circuits in five minutes in a density balanced fashion, while an ILP-based approach can finish only the smallest five circuits. In addition, we can remove more than 95% of the timing variation induced by overlay for tested structures.

1. INTRODUCTION

Current lithography technology has been facing severe limitations because the lithography equipment using a 193nm wavelength light source is hard to print sub-30nm half pitch patterns.

As an ideal solution, EUV(Extreme Ultra-Violet) lithography has been proposed. However, EUV lithography equipment is not available for 22nm production due to technical barriers such as the lack of power sources, resists, and defect-free masks. An alternative choice of 22nm patterning is Double Patterning Lithography (DPL) [1–3] which prints two lines in a pitch.

DPL requires layout decomposition before manufacturing. When the distance between two patterns is less than min_s, two rectangles need to be decomposed on different masks. We can resolve conflicts by inserting stitches during decomposition as shown in Fig. 1(a). However, minimum stitch insertion is required to reduce area and improve yield. Not all the conflicts can be resolved by inserting stitches. An irresolvable conflict is called a native conflict in Fig. 1(b). The only way to remove native conflicts is to modify layout. After layout modification, we need to decompose layout iteratively. Therefore, a fast decomposition algorithm is required to shorten the time of an iterative layout modification and decomposition.

(a) Decomposable layout (b) Undecomposable layout

Figure 1: Layout decomposition.

A rule-based decomposition method was proposed in [4]. The combined approach of routing and decomposition was proposed in [5]. The paper in [6] proposed a layout decomposition method based on constraint graph construction and ILP. The paper in [7] presented a simultaneous optimization of conflicts and stitches with an ILP formulation. Our algorithm does not need to consider conflicts during decomposition explicitly because our framework guarantees conflict-free after an initial (relative) coloring. In addition, since ILP is NP-hard, the algorithm in [6,7] cannot be applied to large scale decomposition which is required to decompose layout balanced or overlay compensated. All the approaches are neither scalable nor flexible to add constraints.

First, our framework is highly scalable enough for large scale layout decomposition. Our decomposition framework consists of three steps: relative coloring, constraint insertion for timing driven decomposition(TDD), and group color assignment. In our framework, relative coloring works for removing every conflict except native conflicts, and we optimize decomposition for minimum stitch and balanced density during a group color assignment step. The optimization for decomposition is mainly based on a graph theoretical approach, which is extended from a min-cut graph partitioning algorithm such as FM [8]. Overall, our framework runs in polynomial time.

Another benefit of our framework is that balanced density can be achieved concurrently with stitch minimization. Non-uniform density can cause more pattern distortion and hotspot [9], while a balanced layout allows more space for scattering bar insertion during OPC(optical proximity correction) as well as better patterning quality. Therefore, the decomposed two layers need to be balanced as much as possible. Our framework balances local density as well as global density.

Last, our framework is flexible to add more constraints during decomposition. Because metal patterns formed by DPL can have space variation due to overlay, it can change coupling capacitance between neighboring wires, resulting in

timing variation [10–12]. Hence, as an application to show the flexibility of our framework, we propose overlay self-compensated decomposition. TDD constraints are generated by an ILP formulation to mitigate the timing variation due to overlay. The number of variables in our ILP formulation for TDD constraints generation is tractable enough for large scale decomposition. The TDD constraints are inserted as a weight on an edge of our graph modeling for color assignment. In the experimental result, we show that timing variation caused by overlay can be reduced from 8.1%∼9.3% to less than 0.5%.

Overall, our graph theoretic decomposition optimizes layout for minimum stitch insertion with balanced density and overlay compensation, simultaneously.

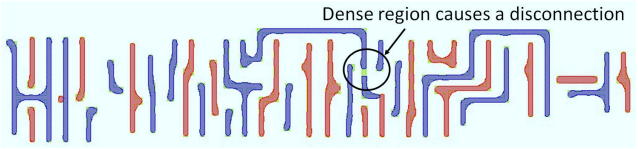
The contributions of this paper include the following:

- We show that the number of stitch insertions required to decompose a layout is equal to the cut size of graph partitioning, and stitch minimization can be achieved by a min-cut partitioning algorithm.
- Our graph based decomposition is up to 10^4 times faster than ILP based decomposition with less than 1% stitch increase.
- We show that layout decomposition plays a role in enhancing patterning quality by enforcing balanced density and overlay self-compensation.
- This is the first work dealing with lithography friendly objectives such as density and variability reduction during layout decomposition, to our best knowledge.

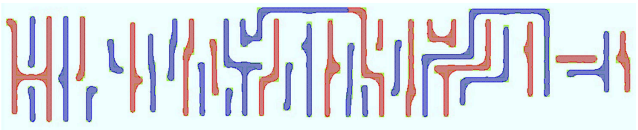
The rest of the paper is organized as follows. We introduce several requirements for decomposition in section 2. Our decomposition framework is proposed in section 3. In section 4, we will explain how to extract TDD constraints, and show our graph based TDD approach. Experimental results are shown in section 5, and we conclude in section 6.

2. DECOMPOSITION REQUIREMENTS AND MOTIVATION

2.1 Balanced Density



(a) Unbalanced decomposition: 38%(Red) and 62% (Blue)

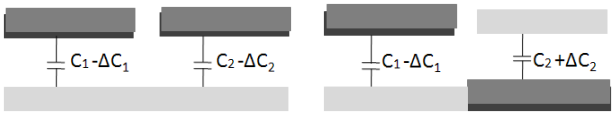


(b) Balanced decomposition: 48%(Red) an 52% (Blue)

Figure 2: Aerial image comparison of decomposed layout with different density.

Even though two features within minimum space are assigned to different masks, unbalanced density can cause lithography hotspot as well as lowered CD uniformity due to irregular pitch. The aerial image of an unbalanced decomposition can have a patterning problem as shown in Fig. 2(a). The balanced decomposition of the same circuit in Fig. 2(b) does not have the problem. Even though the decomposed layout in Fig. 2 does not represent a general case, balanced decomposition provides more lithography friendly layout because of higher regularity [13]. Therefore, balanced density should be considered in decomposition algorithms.

2.2 Overlay compensation



(a) Decomposition without overlay compensation

(b) Decomposition with overlay compensation

Figure 3: Overlay compensated decomposition.

The main problem of litho-etch-litho-etch(LELE) DPL comes from overlay between the 1^{st} and the 2^{nd} patterning step. Even though manufacturing engineers are working to gain more overlay control, it is impossible to remove overlay completely. Therefore, it is desirable to compensate overlay during design phase. Fig. 3 shows an observation how to compensate the overlay effect. When we do not consider overlay during decomposition, the variations of the coupling capacitance between two metal layers are in the same direction(Fig. 3(a)). However, Fig. 3(b) shows less timing variation because C_1 decreases while C_2 increases with overlay. In the section IV, we propose a decomposition method in order to compensate overlay effect.

3. BI-PARTITIONING BASED DECOMPOSITION

3.1 Overall decomposition flow

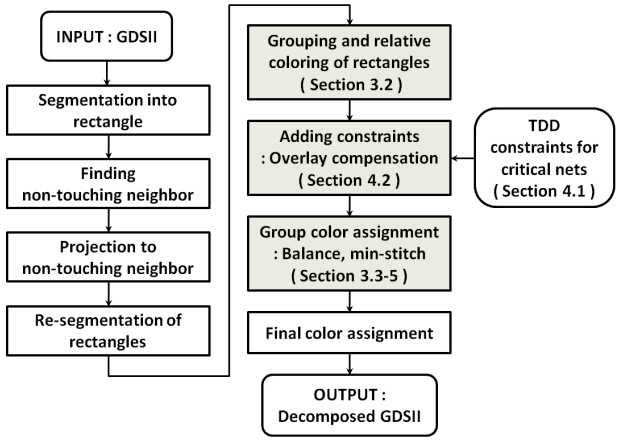
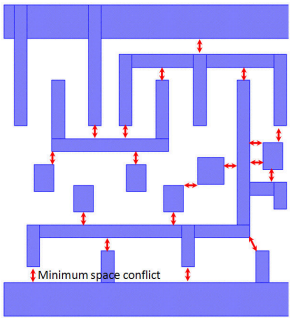


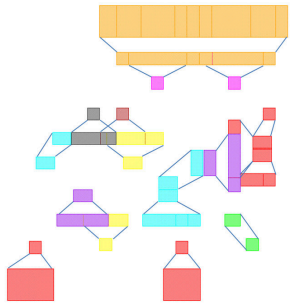
Figure 4: Overall flow of the decomposition framework.

The overall flow of our framework is shown in Fig. 4. The first step is to divide every shape into rectangles which are basic units in our framework because polygon is difficult to find neighboring shape, and dividing into single grid unit can increase complexity and memory usage.

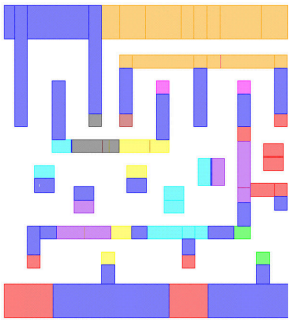
Fig. 5 shows the decomposition result after each step. A rectangle is divided into smaller ones based on projected region from a non-touching neighboring rectangle in Fig. 5(b). The re-segmented rectangles based on the projection in Fig. 5(c) are grouped by traversing non-touching neighbors in Fig. 5(d). If a re-segmented rectangle does not have a non-touching neighbor, the rectangle becomes an independent group by



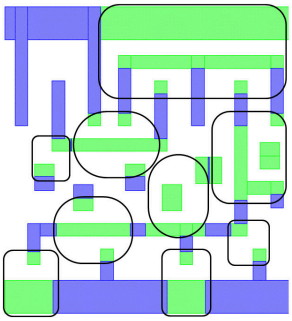
(a) Segmentation into rectangles and finding conflicts



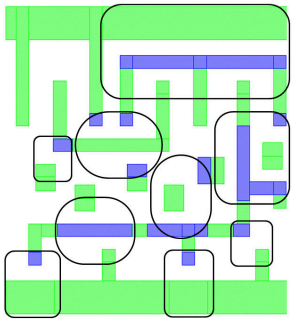
(b) Projection to conflict rectangles within min_s



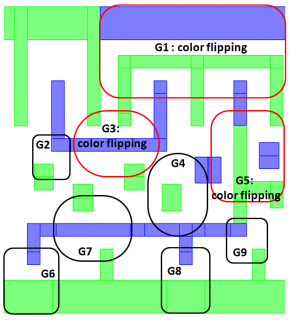
(c) Re-segmentation based on projected rectangles



(d) Rectangle grouping for initial coloring



(e) Relative coloring to remove conflicts within groups



(f) Group color assignment to minimize stitch insertion

Figure 5: Intermediate decomposition results of the framework (AOI2 cell, Metal1).

itself. A group consisted of more than two rectangles is defined as a dependent group. There are nine dependent groups in Fig. 5(d). In Fig. 5(e), an initial (relative) color is assigned to the rectangles to remove every conflict. There are 23 stitches after relative color assignment which does not consider the stitch minimization. We will explain grouping and relative coloring in section 3.2.

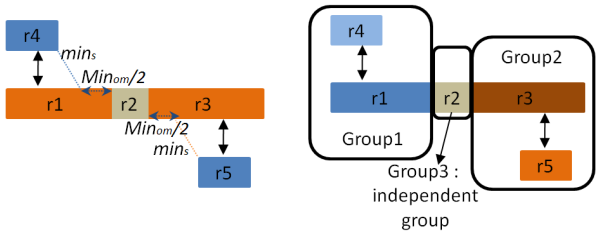
Layout decomposition at the relative coloring step is done without any objective. To minimize stitches, we introduce group color determination with the min-cut partitioning based algorithm in section 3.4. Fig. 5(f) shows the final coloring result after stitch minimization. If color1 is assigned as a group color, all the rectangles in the group have to flip their pre-assigned relative color. And vice versa, all the rectangles in the group of color0 should keep their initial color assignment. For example, all the rectangles in G1, G3, and G5 flipped their initial color in Fig. 5(f). Notice that an independent group may flip its color as well. After color flipping, three stitches are required to resolve conflicts.

The separation of relative coloring and group color assignment enable us to find a native conflict during the relative coloring step, which reduces the design time for iteratively removing native conflicts and decomposition.

Algorithm 1 Grouping_and_Relative_Coloring(DFS)

```
A set of re-segmented rectangles S
RelativeColor = 0
for each rectangle s ∈ S do
    Create new group G
    s →relative_color = Relative_Color
    G →rectangle.insert(s)
    A set of non_touching_neighbors N
    for each rectangle n ∈ N do
        Recursive_Relative_Coloring(G,n,INV(RelativeColor))
    end for
end for
Recursive_Relative_Coloring(G,s,RelativeColor)
s →relative_color = RelativeColor
G →rectangle.insert(s)
A set of non_touching_neighbors N
for each rectangle n ∈ N do
    Recursive_Relative_Coloring(G,n,INV(RelativeColor))
end for
```

3.2 Grouping and Relative Coloring



(a) Re-segmentation (b) Relative coloring

Figure 6: Re-segmentation and grouping process.

A rectangle is divided into smaller pieces of rectangles based on the candidate location for stitch insertion. For example, in Fig. 6(a), a rectangle is divided into three small rectangles ($r1$, $r2$, $r3$). Projection from $r4$ makes $r1$ to be colored differently from $r4$. In addition, to get enough margin for overlap of stitch insertion, $r1$ should be extended by $min_{om}/2$, where min_{om} is the minimum overlap margin for overlay. $r3$ and $r5$ should have a different color in a similar way. $r2$ does not have a non-touching neighbor which means that we can assign any color to $r2$. The re-segmented rectangles are grouped by a neighboring relation in Fig. 6(b). A relative color is assigned to all the grouped rectangles, which indicates their relative relationship to other rectangles in the same group. Grouping and relative coloring can be implemented simultaneously with Algorithm 1.

3.3 Group Color Assignment Problem

In a given layout, let $G = \{g_i | 1 \leq i \leq n\}$ be a set of groups and $H = \{h_w | 1 \leq w \leq x\}$ be a set of stitch candidates and $S = \{s_j | 1 \leq j \leq m\}$ be a set of stitches to be inserted. Every boundary between two touching groups becomes a candidate for stitch insertion. x is the number of touching pairs of groups. The goal of our problem is to find minimum set of S from H. When two touching groups have different colors, h_w becomes an element of S. Let $G_0 = \{g_{i_0} | 1 \leq i_0 \leq n_0\}$ be a set of groups assigned to $color0$ and $G_1 = \{g_{i_1} | 1 \leq i_1 \leq n_1\}$ be a set of groups assigned to $color1$. The group color

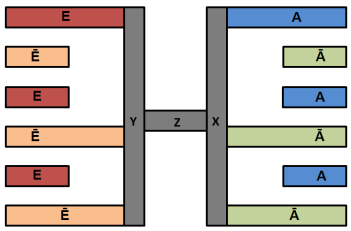
assignment is a process which assigns g_i to either G_0 or G_1 to minimize m, the element number of S.

Let A_w and B_w be the w^{th} touching pair between two groups. If A_w and B_w have different colors, stitch insertion is required and h_w becomes an element of S. From the observation, an objective function for stitch minimization can be formulated as minimizing $\sum_w A_w \bigoplus B_w$, which is formulated with an ILP in (1).

$$\begin{aligned} Minimize &: \sum X_w \\ Subject\ \ To &: A_w - B_w \leq X_w \\ &: B_w - A_w \leq X_w \end{aligned} \tag{1}$$

3.4 Min-Cut Based Stitch Minimization

To link the boolean objective function with layout segmented into rectangles, a boolean variable is assigned to each rectangle to indicate the group and relative color. For example, rectangle group A in Fig. 7(a) is expressed by A and $\overline{A}$ to indicate group and relative color. Independent rectangle groups represented by X, Y, and Z do not have a complementary variable.



(a) Boolean variable assignment after grouping and relative coloring

Minimize :
A⊕X + X⊕Z + Y⊕Z + E⊕Y
+ 2(Ā⊕X) + 2(Ē⊕Y)
Subject to :
A∈{0,1}, E∈{0,1}, X ∈{0,1},
Y ∈{0,1}, Z ∈{0,1}

(b) Formulation for minimum stitch insertion

Figure 7: An example showing group color assignment formulation after relative coloring.

There are five groups in Fig. 7(a). Our goal is to determine the five binary variables to minimize the objective function in Fig. 7(b). The ILP formulation in (1) is not an efficient way for the problem because ILP is in class NP-Hard. There is no way to find the exact solution for the formula in polynomial time because the number of solution sets is 2^n, when there are n groups in a layout. Therefore, we need an efficient heuristic method for our problem.

If we present the objective function as a graph, according to the following theorem, the group color assignment for minimum stitch insertion can be achieved with a graph bi-partitioning algorithm. Even though a graph partitioning does not guarantee an optimal solution, efficient heuristics such as FM have been studied.

Theorem 1 *The number of stitches in layout decomposition is equal to the cut size of the bi-partitioning problem in graph theory.*

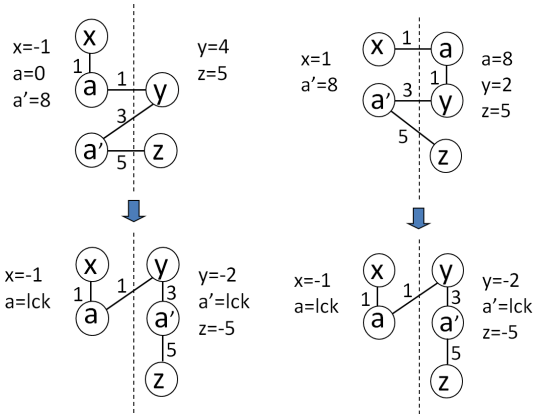
Proof. Let $G = \{g_i | 1 \leq i \leq n\}$ be a set of groups and $S = \{s_j | 1 \leq j \leq m\}$ be a set of stitches. After group color assignment, $G_0 = \{g_{i_0} | 1 \leq i_0 \leq n_0\}$ is a set of groups assigned to *color*0 and $G_1 = \{g_{i_1} | 1 \leq i_1 \leq n_1\}$ is a set of groups assigned to *color*1. s_j appears only when two touching groups are assigned to g_{i_0} and g_{i_1} because a stitch disappears when two touching groups have the same color. Therefore, the number of stitches, m, is equal to the number of times that two touching groups have different colors.

In a similar way, let $C = \{c_k | 1 \leq k \leq p\}$ be a set of cuts and $V = \{v_l | 1 \leq l \leq q\}$ be a set of vertexes in the graph for bi-partitioning. The cut in bi-partitioning appears when two connected vertexes are in a different partition. Therefore, the cut size, p is equal to the sum of the edge weight linked between other partitions. When we consider G as a vertex in a graph, G_0 and G_1 become two partitioned set of vertexes. Therefore, m becomes p with graph partitioning expression, and the number of stitches is equal to the cut size of the bi-partitioning. □

According to Theorem 1, the solution of a min-stitch formulation is the same as the results of a min-cut bi-partitioning. The only difference from a conventional min-cut partitioning algorithm is that a vertex representing the group color is incompatible with a complementary vertex of the same group. We define the two nodes to be in different partitions as a repulsive pair.

3.5 Modified Graph Min-Cut partitioning

We implement the classic FM partitioning algorithm with the following modification in order to support repulsive layout pairs and balanced density.



(a) A and $\overline{A}$ are in the same partition

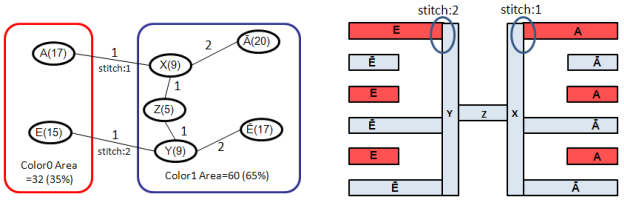
(b) A and $\overline{A}$ are in the different partitions

Figure 8: An example showing how to handle a repulsive layout pair in FM partitioning algorithm.

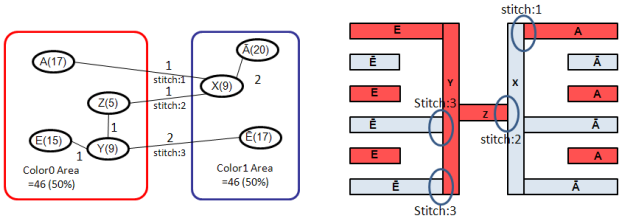
Repulsive pairs cannot be guaranteed to be assigned to different partitions simply by assigning negative weights because min-cut partitioning can fall in local minima. Therefore, in our method, we process the repulsive pairs in two different ways. First, if the two nodes are in the same partition, their moving gains are calculated independently. The one with a higher gain is subject to be moved first, then upon moving, the other one will be locked as well with a gain of zero. In Fig. 8(a), A and $\overline{A}$ form a repulsive pair, but they are in the same partition, so their gains are zero and eight respectively. When $\overline{A}$ is moved, A is locked as a result.

On the other hand, if the two nodes are in the opposite partition, their gains are kept the same and the value is the sum of their original gains. Upon picking one of them as the candidate for moving, a swap is carried out to ensure they are still kept on opposite sides. If A and $\overline{A}$ are in opposite partition like shown in Fig. 8(b), the sum of their original gains is eight, so their gains become eight. It results in a swap in A and $\overline{A}$ when A is picked for moving. Regardless of what the initial partition is in this example, it results in the same final solution under our method.

In Fig. 9, we construct a graph to decompose the layout in Fig. 7. The pair of group colors represented by A and $\overline{A}$ is a repulsive pair. E and $\overline{E}$ also need to be in different partitions. Edge weight between two vertexes means the number



(a) Min-cut partitioning and the decomposed layout



(b) Balanced min-cut partitioning and the decomposed layout

Figure 9: Group color assignment with different partitioning scheme and decomposition results.

of touching pairs between two groups. For example, since $\overline{A}$ and X have two touching points, the edge weight between $\overline{A}$ and X becomes two. Fig. 9(a) shows min-cut partitioning and the corresponding decomposed layout for minimum stitch insertion. $Color0$ is assigned to all rectangles belonging to group$A$ and group$E$, while rectangles in $\overline{A}$, $\overline{E}$, X, Y and Z are assigned to $Color1$

During partitioning, we can control stitch count and density balance. Fig. 9(b) shows the balanced partitioning and the decomposed layout which has four stitches. Global balance can be achieved by adding a weight on a vertex in the graph. In order to maintain a balance between two partitions, we need to start with a balanced initial partitioning solution and then keep tracks of the total weight on both partitions. Every time before moving a vertex into another partition, a check is carried out to ensure the area of the original partition does not drop below a certain threshold after moving. To consider local balance, we need to divide the layout into subdivisions and assign a balance constraint to each of them . The local balance is guaranteed by not moving a vertex to another partition if the move breaks any of the balance constraints. A balanced partitioning result with the area weight is shown in Fig. 9(b). The decomposed layout is balanced evenly with four stitch insertions. The weight of the vertexes indicates the total area of rectangles in a group.

3.6 Complexity Analysis

Let N be the number of rectangles, and E be the number of neighboring pairs. Segmentation from polygon into a rectangle takes O(N). Finding neighbors need O(NlogN) because sorting according to coordinate is required. The complexity of projection to non-touching neighbor is O(E). Grouping and relative coloring using DFS requires O(N+E). Group color assignment with min-cut partitioning can be done in linear time, O(N). Therefore, the complexity of the framework is O(NlogN). Since generating TDD constraints runs for selected nets and the number of variable is usually less than one hundred, we can ignore the complexity of the TDD part.

The proposed decomposition framework works in polynomial time, while ILP based approaches [6,7] are in NP-hard class.

4. TIMING DRIVEN DECOMPOSITION

4.1 TDD constraints for overlay compensation

We can get a robustly decomposed layout against timing variation due to overlay. Delay variation due to overlay is expressed in Eqn 2. Since non-coupling capacitance such as metal to ground and output loading is independent of overlay, $\Delta Delay_{overlay}$ does not need to consider non-coupling capacitance.

$$\Delta Delay_{overlay} = Delay_{no\_overlay} - Delay_{overlay} \quad (2)$$

In Eqn 3, let C_c be a coupling capacitance when there is no overlay with distance d_0. ϵ is the permittivity of an isolation material such as SiO_2. When we consider space variation due to overlay, C_c becomes $C_{c\_overlay}$ which has a distance variation presented by Δd_0. Let g be a variation of the coupling capacitance between horizontally parallel patterns, and h be a variation of the coupling capacitance between vertically parallel patterns.

$$\begin{aligned} &C_c = \epsilon Area/d_o, \Delta C_c = d_o/(d_o + \Delta d_o) \\ &C_{c\_overlay} = C_c \Delta C_c = \epsilon Area/(d_o + \Delta d_o) \\ &g \;\; and \;\; h = 1 - \Delta C_c = \Delta d_o/(d_o + \Delta d_o) \end{aligned} \quad (3)$$

Let a_k be the multiplication factor of the coupling capacitances for a horizontally parallel pattern and b_k be the multiplication factor for vertically parallel patterns. a_k and b_k are defined as k_{th} coupling capacitance times the resistance from driver to the location of the coupling capacitance. In equ 4, R_d is the driver resistance and R_n is the n^{th} resistance from the driver. We assume that coupling capacitance is located in the m^{th} resistance. MF_k is the miller factor [14,15] of the k^{th} coupling capacitance for equal timing determined by slew rate of victim and aggressor.

$$a_k \;\; and \;\; b_k = \left\{ R_d + \sum_{n=1}^{m-1} R_n + 0.5R_m \right\} MF_k C_{ck} \quad (4)$$

$\Delta Delay_{overlay}$ can be re-written in Eqn 5. i denotes the number of aggressors horizontally, and the number of aggressors vertically is presented by j.

$$\begin{aligned} \Delta Delay_{overlay} &= a_1 g_1 + \ldots + a_i g_i + b_1 h_1 + \ldots + b_j h_j \\ &= AG^T + BH^T \\ \text{where, } A &= [a_1 \;\; a_2 \;\; \ldots \;\; a_i], \; G = [g_1 \;\; g_2 \;\; \ldots \;\; g_i] \\ B &= [b_1 \;\; b_2 \;\; \ldots \;\; b_j], \; H = [h_1 \;\; h_2 \;\; \ldots \;\; h_j] \end{aligned} \quad (5)$$

Distance variation due to translation overlay is expressed in Eqn 6 [10]. For simplicity, we will focus on translation overlay in this paper. Our work can be extended to include rotation and magnification overlay. λ is the amplitude of translation overlay, and θ is the translation angle which is a totally random value. γ represents the relative geometry relation between the 1^{st} and the 2^{nd} pattern. Then, our problem is defined to assign γ to reduce overlay effect on timing.

$$\Delta d_o = \lambda cos(\theta - \gamma) \quad (6)$$

From Eqn 3 and 6, we can derive g_i and h_i. Since g_i has two possible choices according to γ (Note that the horizontally parallel pattern can have two geometric relations: $\gamma = \pi/2$ or $\gamma = 3\pi/2$), g_i is modeled with a binary variable, x_i which indicate geometric relations. For example, if x_i has

one, it means that $\gamma = \pi/2$ is preferred to reduce overlay effect, while the opposite geometric relation ($\gamma = 3\pi/2$) is preferred if x_i is zero. We can simplify g_i when d_0 is relatively larger than λ. Since overlay amplitude should be controlled to less than 10% of spaces(d_0) between two patterns, $d_0 + \lambda sin(\theta)$ can be simplified to d_0. h_j has a $\pi/2$ phase difference with g_i. After simplication, we present g_i and h_j in Eqn 7.

$$\begin{aligned} g_i =& x_i\lambda cos(\theta - \pi/2)/(d_0 + \lambda cos(\theta - \pi/2)) \\ &+(1 - x_i)\lambda cos(\theta - 3\pi/2)/(d_0 + \lambda cos(\theta - 3\pi/2)) \\ \approx& \frac{\lambda}{d_0}(2x_i - 1)sin(\theta) \\ h_j =& y_j\lambda cos(\theta)/(d_0 + \lambda cos(\theta)) \\ &+(1 - y_j)\lambda cos(\theta - \pi)/(d_0 + \lambda cos(\theta - \pi)) \\ \approx& \frac{\lambda}{d_0}(2y_j - 1)cos(\theta) \end{aligned} \quad (7)$$

From Eqn 5 and 7, our cost function for overlay impact on timing is derived in Eqn 8. To minimize the cost function, we need to minimize $\sqrt{\alpha^2 + \beta^2}$, the amplitude of $sin(\theta+\phi)$. ϕ works only for phase difference. Since the horizontal and vertical patterns are orthogonal, we can optimize α^2 and β^2 independently, which means that the overlay impact is minimized when α^2 and β^2 have minimum values, respectively.

$$\sqrt{\alpha^2 + \beta^2}sin(\theta + \phi)$$

$$\begin{aligned} \text{where,} \quad & \alpha = 2AX^T - \sum_{n=1}^{i} a_n, \quad \beta = 2BY^T - \sum_{n=1}^{j} b_n \\ & X = [x_1, x_2, \ldots, x_i], \quad Y = [y_1, y_2, \ldots, y_j] \\ & \phi = sin^{-1}\left(\frac{\beta}{\sqrt{\alpha^2 + \beta^2}}\right) \end{aligned} \quad (8)$$

Finally, we propose an ILP formulation to find the relative positions of horizontally parallel patterns to minimize α^2.

$$\begin{aligned} minimize \quad & 4\sum_{n=1}^{i}\left\{a_n\left(AW_n^T - w_{nn}\sum_{p=1}^{i}a_p\right)\right\} + \left(\sum_{p=1}^{i}a_p\right)^2 \\ s.t. \quad & w_{ii} = x_i \\ & 1 + w_{ij} \geq x_i + x_j \\ & x_i \geq w_{ij} \\ & x_j \geq w_{ij} \end{aligned} \quad (9)$$

New binary variables, w_{ij} are introduced to change quadratic integer programming to linear integer programming. x_i and w_{ij} are binary variables in (9), and W_n are defined as follows.

$$W_n = [w_{n1} \quad w_{n2} \quad \ldots \quad w_{ni}] \quad (10)$$

In a similar way, we can get a solution set of Y to minimize β^2 by substituting a to b, and x to y in the above ILP formulation.

Fig 10 shows an example pattern to apply the proposed method. a_1, a_2, b_1, and b_2 corresponding to each coupling capacitance are presented in Fig 10. Here, there are two horizontally parallel aggressors and two vertically parallel aggressors.

$$\begin{aligned} \alpha &= a_1(2x_1 - 1) + a_2(2x_2 - 1) \\ \beta &= b_1(2y_1 - 1) + b_2(2y_2 - 1) \end{aligned} \quad (11)$$

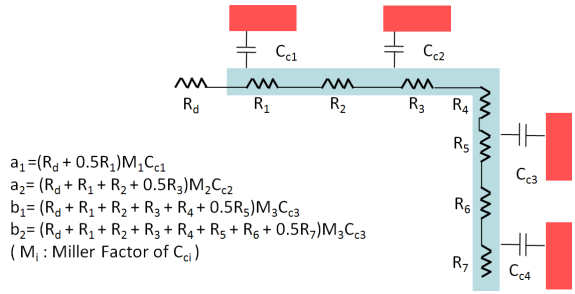
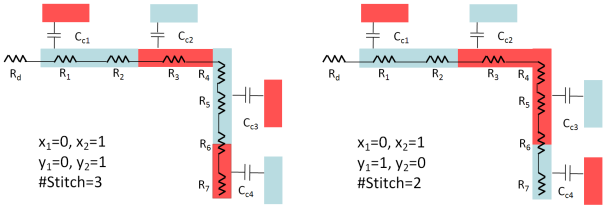


Figure 10: An example with four neighbors.

α and β of Fig 10 are shown in Eqn 11. By the proposed ILP formulation in Eqn 9, we can determine x_1 and x_2 to minimize α^2, and y_1 and y_2 to minimize β^2, respectively. Here, x_1 and x_2 mean the relative decomposition for C_{c1} and C_{c2} as shown in Fig. 11.



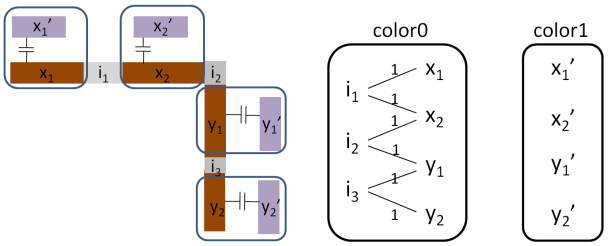
(a) Overlay compensation with three stitches

(b) The same compensation with two stitches

Figure 11: Different decomposition with TDD.

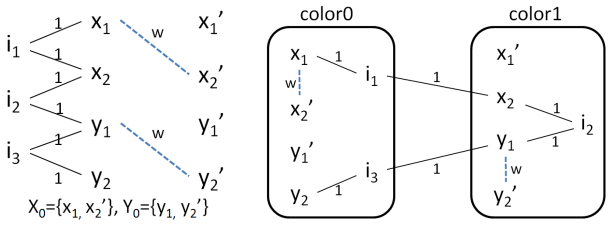
After applying our ILP formulation, we can get two different decomposition results as shown in Fig 11. Three stitches are inserted at Fig 11(a). However, a different solution for the same overlay compensation requires only two stitches. We will present our combined approach of TDD constraints and graph based stitch minimization in order to obtain stitch optimization with TDD constraints.

4.2 TDD constraints aware decomposition



(a) After relative coloring

(b) Group color assignment without TDD constraints



(c) TDD Constraints insertion

(d) Group color assignment when the edge weight(w) is bigger than one

Figure 12: Decomposition with TDD constraints.

From the previous subsection, we can get constraints for TDD. X and Y are determined by the ILP formulation. Let

X be partitioned into X_0 and X_1. In Fig 11(a), X_0 is $\{x_1\}$ while x_2 belongs to X_1. Since the complement of x_2 is x_2', we can express the constraints with $X_0 = \{x_1, x_2'\}$. Since the elements in X_0 need to have the same color, x_1 should be forced to be with x_2' in the same partition. We can make them to be in the same partition by adding weighted edge. The allowed stitch increase can be controlled by adjusting the weight. Decomposition for more overlay compensation needs to increase edge weight which results in more stitches.

Fig. 12 shows the graph based TDD procedure. The grouping and relative coloring result is presented in Fig. 12(a). Graph based decomposition without TDD constraints is in Fig. 12(b). Two edges are inserted to consider TDD constraints in Fig. 12(c). When the weight denoted by w is bigger than one, new partition result is in Fig. 12(d). Two stitches are inserted for TDD, and the decomposition result looks like Fig. 11(b).

5. EXPERIMENTAL RESULTS

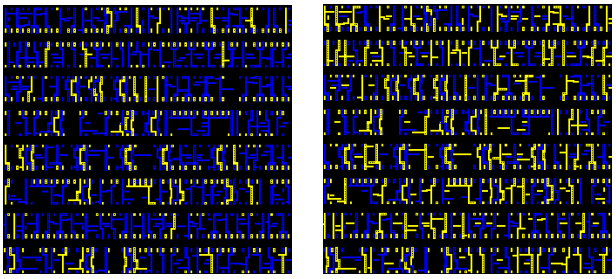
We implement the decomposition framework in C++ and OpenAccess2.2 in order to interface with GDSII directly. We test on a 3.0GHz Linux machine with 4G RAM to verify our algorithm.

First, we show the scalability and runtime of our algorithm. Poly-silicon layer in benchmark circuits is scaled down to 40nm half pitch. ISCAS-85&89 benchmark circuits are used to verify the scalability. Before decomposition, minimum space between poly-silicon was 40nm. We select min_s=42nm and min_{om}=10nm for decomposition to avoid native conflicts, which should be removed by layout modification. ISCAS-89 circuits have many native conflicts when min_s is bigger than 43nm because the ISCAS-89 benchmarks we have are not designed for double patterning friendly. Table 1 shows the runtime of decomposition as the design size increases. S38584 which is the biggest circuit in the benchmark can be decomposed evenly in 285.24s. Since min_s=42nm is close to 40nm minimum space, only a few stitches are required to resolve the conflicts.

Second, we verify the quality and efficiency of our framework. Table 2 compares our ILP formulation in (1) and our heuristic based on min-cut partitioning with respect to runtime and stitch optimization for layout decomposition. GLPK(GNU Linear Programming Kit) solver is used for ILP solving. Because decomposition with ILP formulation is intractable as circuit size increases, we divide a circuit into several parts by row of cells because we cannot decompose with ILP even for small circuits like C499. Since poly-silicon layers are isolated with other parts of the circuit by row of cells, decomposition after dividing into several rows still provides an exact solution with the ILP formulation. Note that our ILP implementation provides an optimal solution for minimum stitch insertion with more runtime because our ILP implementation does not use any speed-up technique described in [6,7], which may sacrifice optimality. We use min_s=54nm and min_{om}=20nm to enable more potential stitch insertions. Since ISCAS-89 benchmarks have native conflicts when min_s is below 54nm, we do not show their results in Table 2. When min_s is bigger than 54nm, there are native conflicts on several ISCAS-85 benchmarks. Therefore, we choose min_s=54nm based on the availability of decomposition, which is enough to show the efficiency of our framework. The quality of the min-cut graph partitioning depends on the initial partitioning. Thus, we execute the modified min-cut partitioning for twenty times and pick the case with minimum stitch. Runtime means the total runtime for twenty runs. When we do not consider balanced density, min-cut partitioning and ILP based decomposition have the same number of stitches except C1908. C1908 has one more stitch in our approach. The runtime of graph theoretic decomposition is 1.4∼9762.6 times faster than that of ILP based decomposition.

Table 1: Runtime comparison($min_s = 42nm$)

Circuit	#Group	#Touching neighbors	Runtime(second)			Results	
			Except partition	Min-cut partition	Total	Inserted Stitches	Balance ratio(%)
C432	1554	763	0.48	0.01	0.49	0	48.18
C499	3503	2260	1.12	0.23	1.35	0	48.08
C880	3105	1308	0.86	0.07	0.93	0	48.10
C1355	4630	2091	1.39	0.21	1.60	1	48.05
C1908	7403	3447	2.81	0.52	3.33	0	48.23
C2670	11325	5291	3.32	0.92	4.24	0	48.05
C3540	13934	6062	4.71	1.26	5.97	0	48.02
C5315	20393	9382	7.19	2.01	9.20	5	48.02
C6288	18836	7764	6.14	1.27	7.41	0	48.02
C7552	29642	13344	11.84	3.13	14.97	0	48.03
S1488	5952	2558	1.57	0.29	1.86	0	48.01
S15850	7983	3282	130.38	9.78	140.16	0	48.04
S35932	188556	75943	263.29	14.21	277.50	0	48.01
S38417	195448	74311	270.43	18.49	288.92	0	48.00
S38584	188298	72342	262.66	22.58	285.24	1	48.01



(a) Yellow(27%), Blue(73%) (b) Yellow(50%), Blue(50%)

Figure 13: C432 decomposed layout.

Fig. 13 compares decomposition between the unbalanced and balanced case at min_s=60nm and min_{om}=20nm for C432. The unbalanced layer has nine stitches with 27% balanced ratio while the balanced decomposition has 17 stitches with globally 50% balanced ratio. In addition, Fig. 13(b) is locally balanced because we enforce the local density balance between 40% and 60% in each cell row. The result also shows that there is the trade-off between stitch counts and density balance.

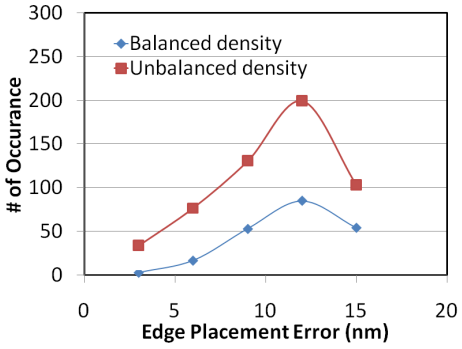


Figure 14: The balanced decomposed layout has less EPE than the unbalanced one (C432).

OPC and lithography simulation is executed using CALIBRE for the two cases in Fig. 13. Edge Placement Error(EPE) distributions after OPC are compared in Fig. 14. The balanced decomposition in Fig. 13(b) shows lower EPE distribution than the unbalanced decomposition in Fig. 13(a), which indicates that the balanced decomposition has less variation than the unbalanced decomposition.

Table 2: Decomposition results with ISCAS benchmark($min_s = 54nm$, $min_{om} = 20nm$)

Circuit	#Groups	#Touching neighbors	No balance, ILP(Exact)				No balance, Graph Partition(Proposed heuristic)				48% balance, Graph Partition(Proposed heuristic)			
			#Partitions for ILP	RunTime (total)	Inserted stitches	Balanced ratio(%)	RunTime comparison	RunTime (total)	Inserted stitches	Balanced ratio(%)	RunTime comparison	RunTime (total)	Inserted stitches	Balanced ratio(%)
C432	1512	1098	1	0.63	1	20.35	x1.4	0.46	1	33.60	x1.0	0.65	2	48.12
C499	3103	3280	12	100.85	50	24.01	x49.9	2.02	50	46.47	x49.9	2.02	50	48.50
C880	3758	2631	14	4525.57	198	30.09	x2773.0	1.63	198	47.12	x2807.4	1.61	198	48.87
C1355	4836	3083	18	702.40	114	18.91	x347.4	2.02	114	36.12	x344.0	2.04	114	48.00
C1908	7795	5472	18	37019.76	371	22.09	x9762.6	3.79	372	46.78	x10422.2	3.55	373	48.66
C2670	12863	9905	-	> 24Hr	-	-	-	6.70	947	43.51	-	6.87	948	49.30
C3540	16638	12021	-	> 24Hr	-	-	-	9.85	1034	41.46	-	10.07	1034	49.39
C5315	24483	18373	-	> 24Hr	-	-	-	17.43	1546	40.87	-	18.50	1549	48.00
C6288	19922	11577	-	> 24Hr	-	-	-	11.57	256	30.81	-	11.25	256	48.13
C7552	34309	24789	-	> 24Hr	-	-	-	30.89	2058	41.97	-	31.52	2060	48.02

Table 3: Overlay compensation with TDD

Test circuit	#Horizontal neighbors	#Vertical neighbors	Timing Variation on overlay	Edge weight	Inserted stitches	Overlay Compensation Rate(%)
net1	3	6	9.3%	0.2	0	0.0
				0.5	1	76.3
				1	3	95.9
				100	3	95.9
net2	10	9	8.1%	0.2	0	0.0
				0.5	0	0.0
				1	3	54.6
				100	10	99.9
net3	16	16	9.0%	0.2	1	41.4
				0.5	3	87.8
				1	9	99.8
				100	9	99.8

Last, we verify the effectiveness of timing driven decomposition. As a test circuit, we made three net structures with a metal spacing of 32nm assuming 3nm of overlay as shown in Table 3. By changing edge weight, we could control the number of inserted stitches. For example, we could see 41.4% overlay compensation with one stitch insertion when edge weight is 0.2 for Net3. Since the maximum peak to peak delay variation caused by overlay is 9.0%, timing variation on overlay becomes 5.274%(=9%*(1-0.414)) after one stitch insertions. It becomes 1.098%(=9%*(1-0.878)) after three stitch insertion. When we increase edge weight, we could see more stitch insertions and higher overlay compensation rate.

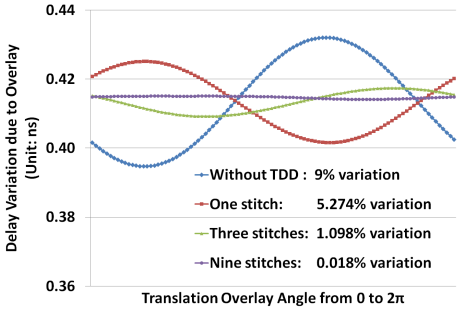


Figure 15: Reduction of timing variation as more stitches are inserted (Net3).

Fig. 15 compares overlay compensation according to different stitch insertion for Net3. As more stitches are inserted, timing fluctuation along the translation angle reduces. When nine stitches are inserted, we can see that there is no timing variation due to overlay. The bottom peak variation is not symmetric with top peak variation in the graph because of the second order term of $sin(\theta)$ and $cos(\theta)$ in Eqn 7. Note that ignoring the second order term with the assumption that $d_0 + \alpha$ is approximately equal to d_0 is reasonable because we could compensate overlay more than 95% in every test structure.

6. CONCLUSIONS

In this paper, we propose an efficient and flexible layout decomposition framework with a graph theoretical approach. All the benchmark circuits can be decomposed in five minutes with balanced density. Our framework can expedite decomposition which requires iterative executions and fixing layout in order to remove native conflicts. Since the decomposition framework is flexible to add constraints, we extend our work to timing driven decomposition which reduces the timing variation due to overlay. As a future work, we can extend the framework for correlation aware decomposition and multiple decomposition using a multi-way partitioning algorithm.

7. REFERENCES

[1] D. Laidler, P. Leray, K. D'have, and S. Cheng. Sources of Overlay Error in Double Patterning Integration Schemes. In *Proc. SPIE 6922*, 2008.

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

[3] David Z. Pan, Jae-Seok Yang, Kun Yuan, Minsik Cho, , and Yongchan Ban. Layout Optimizations for Double Patterning Lithography. In *IEEE 8th International Conference on ASIC (ASICON)*, Oct 2009.

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

[5] M. Cho, Y. Ban, and D. Z. Pan. Double Patterning Technology Friendly Detailed Routing. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2008.

[6] A.B. Kahng, C.-H. Park, and H. Yao. Layout Decomposition for Double Patterning Lithography. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2008.

[7] K. Yuan, J.-S. Yang, and D. Z. Pan. Double Patterning Layout Decomposition for Simultaneous Conflict and Stitch Minimization. In *Proc. Int. Symp. on Physical Design*, March 2009.

[8] C.M. Fiduccia and R.M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *Proc. Design Automation Conf.*, June 1982.

[9] Alfred K. Wong. *Resolution Enhancement Techniques in Optical Lithography*. SPIE Publications, 2001.

[10] J.-S. Yang and D. Z. Pan. Overlay Aware Interconnect and Timing Variation Modeling for Double Patterning Technology. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2008.

[11] R. S. Ghaida and P. Gupta. Design-Overlay Interactions in Metal Double Patterning. In *Proc. SPIE 7275*, 2009.

[12] E. Y. Chin and A. R. Neureuther. Variability aware interconnect timing models for double patterning. In *Proc. SPIE 7275*, 2009.

[13] J. Rubinstein and A. Neureuther. Post-decomposition assessment of double patterning layout. In *Proc. SPIE 6924*, 2008.

[14] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Miller Factor for Gate-Level Coupling Delay Calculation. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2000.

[15] A. B. Kahng, S. Muddu, and E. Sarto. On Switch Factor Based Analysis of Coupled RC Interconnects. In *Proc. Design Automation Conf.*, June 2000.