A Cell-Based Row-Structure Layout Decomposer for Triple Patterning Lithography

Hsi-An Chien, Szu-Yuan Han, Ye-Hong Chen, and Ting-Chi Wang
Department of Computer Science
National Tsing Hua University, Hsinchu 300, Taiwan
{hsianchien, y2k20096, ckbh1621}@gmail.com, tcwang@cs.nthu.edu.tw

ABSTRACT

In this paper, we study a cell-based row-structure layout decomposition problem for triple patterning lithography (TPL) which asks to minimize a weighted sum of coloring conflicts and stitches. We show how to extend a prior graph-based approach to solve the problem optimally under certain assumptions. Furthermore, several methods to substantially reduce the graph size and hence to accelerate the extended approach are presented. Experimental results show that our decomposer can significantly outperform a state-of-the-art work in terms of both solution quality and run time.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Design

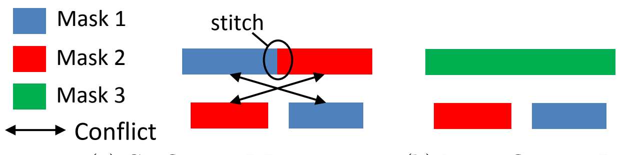
Keywords

Cell-Based Row-Structure Layout, Triple Patterning Lithography (TPL), TPL Layout Decomposition

1. INTRODUCTION

While double patterning lithography (DPL) [9] has been successfully applied in mass production of 20nm technology node, however, it will not be able to completely resolve conflicting features for layers with much denser and finer features, e.g., the first metal layer (M1 layer) in sub-20nm technology node. In addition, the next-generation lithography technologies (e.g., extreme ultraviolet lithography and electron-beam lithography) are still not ready. Accordingly, triple patterning lithography (TPL) is regarded as one of the most promising solutions for 14/10nm because it could be more flexible in mask assignment and introduce less coloring conflicts than DPL .

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ISPD'15, March 29–April 1, 2015, Monterey, CA, USA.
Copyright © 2015 ACM 978-1-4503-3399-3/15/03 ...$15.00.
http://dx.doi.org/10.1145/2717764.2717768 .



(a) Conflicts in DPL (b) No conflict in TPL
Figure 1: DPL vs. TPL Decompositions.

If the distance between two polygons on a layer is less than the minimum coloring distance d_{min} in multiple patterning lithography, the polygons must be assigned to different masks (i.e., different colors) to prevent a *coloring conflict*. Besides, a polygon can be split into two sub-polygons by inserting a stitch to further resolve conflicts. However, [1, 12] pointed out that even stitch insertion can not avoid coloring conflict in DPL. Fig. 1(a) gives an example where the distance between any two of the polygons is less than d_{min} so that there still exist two conflicts by using DPL with stitch insertion. Nonetheless, a conflict-free decomposition can be easily found by dividing the layout into three masks (i.e., TPL) as shown in Fig. 1(b).

Recently, several relevant problems in TPL such as layout decomposition [2, 4, 10, 12], TPL-aware routing [5, 6] and TPL-aware placement [3, 7, 11] have been explored and studied. Although the TPL layout decomposition problem is NP-hard for general layouts [12], it has been shown that for any cell-based row-structure layout, whether a TPL decomposition solution that has no coloring conflict and no stitch insertion exists for the M1 layer can be exactly determined in polynomial time [8]. However, the algorithm in [8] will return nothing for an indecomposable layout (i.e., there is no conflict-free solution for the layout) to help designers further resolve the conflicts in the layout. Instead, if there is a decomposition solution with the minimal total cost in terms of conflicts and stitches, it will be more helpful to designers for subsequent layout modifications.

In this paper, we study a cell-based row-structure TPL layout decomposition problem which asks to minimize a weighted sum of coloring conflicts and stitches. Different from [8], our algorithm can always find a decomposition with a minimal cost even though the given layout is indecomposable. Our contributions are summarized as follows:

- We extend an existing graph-based approach [8] to solve the cell-based row-structure TPL layout decomposition problem.
- We present several methods to substantially reduce the graph size and hence to accelerate the extended approach.

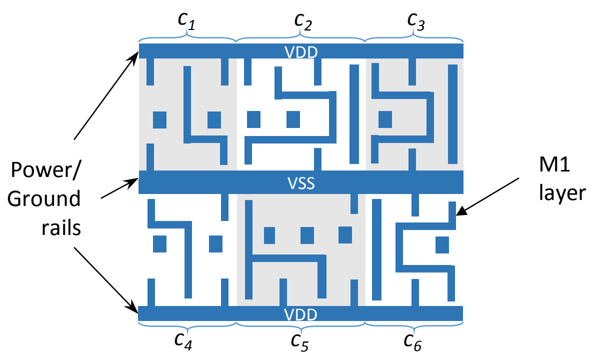


Figure 2: A sample cell-based row-structure layout.

- We parallelize our proposed approach to tackle a layout with multiple rows simultaneously.
- The superiority of our approach over prior works is demonstrated through extensive experiments.

The rest of this paper is organized as follows. Section 2 formulates a cell-based row-structure TPL layout decomposition problem for M1 layer. Section 3 describes our algorithm by extending the approach in [8] as well as explains how to parallelize the algorithm to simultaneously handle multiple rows. Section 4 presents and details several acceleration methods. Experimental results are reported in Section 5, and the conclusion is drawn in Section 6.

2. PROBLEM FORMULATION

Problem 1 (TPL Layout Decomposition). *Given a cell-based row-structure M1 layout (see an example shown in Fig. 2) and a minimum coloring spacing d_{min}, the TPL layout decomposition problem asks to find a TPL layout decomposition with a minimal weighted sum of coloring conflicts and stitches. The weighted sum is defined by Eq. (1), where α and β are user-defined parameters. #conflicts and #stitches denote the number of coloring conflicts and the number of stitches, respectively.*

$$Cost = \alpha \#conflicts + \beta \#stitches. \quad (1)$$

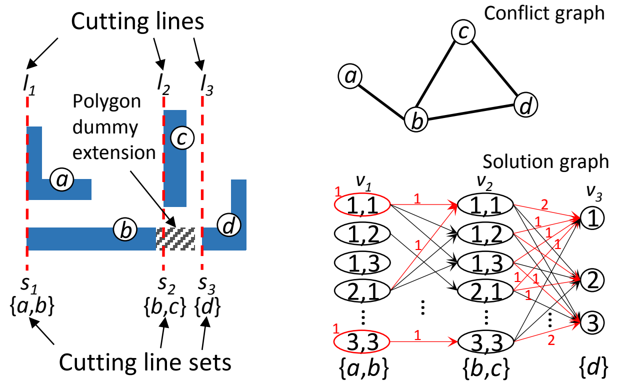
Because there is no coloring conflict between the polygons in different rows [8, 11], the TPL layout decomposition problem for each row can be individually solved. In other words, the TPL layout decomposition problem can be considered as a set of single-row decomposition problems. For an unrouted design, there is also no conflict between two polygons respectively in two non-adjacent cells in a row.

3. LAYOUT DECOMPOSITION APPROACH

3.1 Graph Model

Since our layout decomposition approach is inspired from the algorithm in [8], we will review some basic definitions presented in [8] (see Definitions 1, 2, 3, and 4) and then elaborate our approach. Let P be the set of polygons in the M1 layer of a cell row and $X = \{x_1, x_2, ..., x_m\}$ be the set of m distinct x-coordinates of the left boundaries of all polygons in P, where x_i is to the left of x_j if $i < j$.

Definition 1 (Conflict Graph). *The conflict graph CG of P is an undirected graph. Each node in CG is associated with a polygon in P, and an edge between two nodes means that the distance between the two corresponding polygons is less than d_{min}.*



(a) Polygon dummy extension, cutting lines, and cutting line sets (b) Conflict graph and solution graph

Figure 3: Illustration of Definitions 1 to 5.

In TPL layout decomposition, two polygons connected by an edge in CG should be assigned to different masks (i.e., different colors); otherwise, they will cause a coloring conflict. In the rest of this paper, two polygons are said to have a *potential conflict* when they are connected by an edge in CG.

Definition 2 (Polygon Dummy Extension). *The polygon dummy extension for a polygon p in P is to extend the right boundary of p to a location x such that $x_{j-1} \leq x < x_j$, where $x_{j-1}, x_j \in X$ and x_j is the the largest coordinate among the left boundaries of the set of polygons each of which has an edge connecting p in CG. Note that when the left boundary of p is located at x_{j-1}, the right boundary of p remains intact after polygon dummy extension.*

Definition 3 (Cutting Line). *For each $x_i \in X$, the cutting line l_i is a vertical line from the top to the bottom of the cell row and is located at x_i.*

Definition 4 (Cutting Line Set). *After performing polygon dummy extension for each polygon in P, the cutting line set s_i of a cutting line l_i is the set of polygons intersecting with l_i.*

According to Definitions 1 $\sim$ 4, we have the set $L = \{l_1, l_2, ..., l_m\}$ of cutting lines derived from X and the set $S = \{s_1, s_2, ..., s_m\}$ of the corresponding cutting line sets. For a polygon p in P, if a cutting line l_i is located at the left boundary of p, we say p creates l_i. Clearly, each cutting line is *created* by at least one polygon.

Definition 5 (Solution Graph). *The solution graph SG of P is a directed graph, where each node in SG records a coloring solution of a cutting line set, and an edge connecting from a node v_i^j of s_i in S to a node v_{i+1}^k of s_{i+1} in S indicates the coloring solutions of v_i^j and v_{i+1}^k are compatible*[1]*. Note that all the possible coloring solutions of a cutting line set are enumerated no matter whether each of them is legal or not. Each node (each edge, respectively) in*

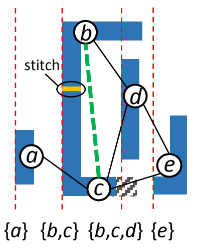
[1]If a polygon is in both s_i and s_{i+1}, its color in v_i^j and its color in v_{i+1}^k should be the same.

SG is assigned a weight indicating the amount of conflicts in the corresponding coloring solution (the corresponding coloring solutions of the two nodes connected by the edge, respectively). The computation of the weights on a node and an edge will be explicitly described later.

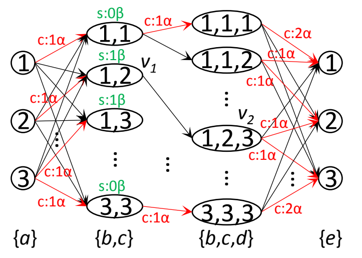
For the layout in Fig. 3(a) and the conflict graph in the top of Fig. 3(b), there are three cutting lines, l_1, l_2, and l_3, and three corresponding cutting line sets, s_1, s_2, and s_3. The right boundary of polygon b is virtually extended by polygon dummy extension such that b is cut by l_2 and hence appears in s_2. Note that the solution graph is partly illustrated in Fig. 3(b) to save space since the entire graph is a bit large; the numbers inside each node represent the coloring solution associated with the node, and the numbers 1, 2, 3 indicate three different colors. The amounts of nodes with respect to cutting line sets $\{a, b\}$ and $\{b, c\}$ are both $3^2 = 9$, as the number of polygons in each of the two sets is 2 and each polygon has three colors to choose. The numbers of edges between adjacent cutting line set pairs ($\{a, b\}$, $\{b, c\}$) and ($\{b, c\}$, $\{d\}$) are 27 and 27 individually.

A *path* in SG goes from a node of the first cutting set (i.e., s_1 in S) to a node of the last cutting set (i.e., s_m in S), and it corresponds to a decomposition of P without stitch insertion. To correctly capture the number of conflicts when finding a decomposition (i.e., a path in the graph), we now describe how to formulate the conflict weights in a node and an edge, respectively. For each cutting line set s_i, there must exist a subset s_i^* of s_i such that s_i^* is the set of polygons creating the cutting line of s_i. We assign a weight of w to a node v_i^j of s_i when the coloring solution of v_i^j induces w conflicts for the polygons in s_i^*. Since all the polygons that have potential conflicts with s_i^* will be in the cutting line set s_{i-1} due to polygon dummy extension, we assign a weight of w to an edge connecting from a node v_{i-1}^k of s_{i-1} to a node v_i^j of s_i when the coloring solutions of v_{i-1}^k and v_i^j induce w conflicts for all pairs of polygons (p, q) with $p \in s_{i-1}$ and $q \in s_i^*$. The reason why we are only concerned with the conflicts for the polygons creating a cutting line is because a polygon could be cut by multiple cutting lines but it creates exactly one cutting line. Therefore, the conflict cost of a path in the graph will not be over-counted and remain correct. Let us see an example in the solution graph of Fig. 3(b). The weight of node v_1 is 1 because there is a potential conflict between two polygons, a and b, the colors of a and b are the same (i.e., color 1), and a and b create the cutting line of $\{a, b\}$. Although polygons b and c have a potential conflict and their colors are also the same in v_2, the weight of v_2 is 0 because c is the only polygon creating the cutting line of {b, c}. The weight of edge (v_1, v_2) is 1 since there is a coloring conflict between b and c. The weight of edge (v_2, v_3) is 2 since there are a coloring conflict between b and d and a coloring conflict between c and d, respectively. Note that if there is no weight on a node or an edge in Fig. 3, Fig. 4, and Fig. 5, it means the weight of the node or edge is 0.

Basically, SG is almost the same as the one defined in [8] except that it allows (1) the coloring solution of a node to have conflict(s), (2) two nodes to be connecting by an edge when the colors of two different polygons respectively in the nodes cause a coloring conflict, and (3) a node (an edge, respectively) to have a weight. Apparently, the following lemma, i.e., Lemma 1, holds for SG.



(a) Cutting lines, cutting line sets, and conflict graph



(b) Solution graph, where two tags, c and s, indicate conflict weight and stitch weight, respectively

Figure 4: Stitch insertion: α and β in (b) are user-defined parameters in Eq. (1).

Lemma 1. *Each possible decomposition of P without stitch insertion corresponds to a path in SG .*

Lemma 2. *The TPL layout decomposition problem without stitch insertion for P can be optimally solved by finding a least-cost path in the solution graph SG of P, where the cost of a path in SG is the accumulated cost of nodes and edges along the path.*

PROOF. Suppose that a valid decomposition which has no stitch insertion and no coloring conflicts exists in P. [8] has proved that they can get a valid decomposition by finding a path in their proposed solution graph for P. By the definition of our solution graph SG for P, such a valid decomposition can be also found from a path with cost of 0 in SG. When there is no conflict-free decomposition in P, the optimal decomposition is the one with minimum conflicts. According to Lemma 1, the optimal decomposition must exist in SG. Besides, the cost of a path in SG exactly indicates the amount of coloring conflicts of the decomposition associated with the path. Thus, the corresponding decomposition of a shortest (i.e., least-cost) path in SG is an optimal TPL layout decomposition. □

3.2 Stitch Insertion

Given a layout with a set of stitch candidates, the polygons are fractured based on the stitch candidates, and the conflict graph is constructed for the fractured polygons. If a stitch candidate is between two polygons, the stitch edge connecting two nodes corresponding to the polygons in the conflict graph denotes that a *stitch* occurs when the polygons are assigned to different masks. Fig. 4 gives an example to describe how to formulate the stitch weights in a solution graph. As can be seen in Fig. 4(a), there is one stitch edge shown in green dashed line connecting two polygons b and c in the conflict graph, and the solution graph with conflict weights and stitch weights is shown in Fig. 4(b). Conflict weight is represented in red color with a tag c, and stitch weight is represented in green color with a tag s. Similar to the computation for conflict weight, for each cutting line set s, there must exist a subset s^* of s such that s^* is the set of polygons creating the cutting line of s. We assign a weight of w to a node v of s when the coloring solution of v induces w stitches for the polygons in s^*. For example, there is a stitch weight of 1β on the node v_1 of cutting line set $\{b, c\}$, since b and c are the polygons creating the cutting line of $\{b, c\}$ and their colors in the coloring solution (i.e., (1, 2)) are different. Although the colors of b and c are also different in the coloring solution of the node v_2 of cutting line set $\{b, c, d\}$, the stitch weight on v_2 is 0 because b and c are not the polygons creating the cutting line of $\{b, c, d\}$ but d is. In this way, the stitch cost of a path in the graph will not be over-counted and remain correct.

Lemma 3. *The TPL layout decomposition problem for P with a given set of stitch candidates can be optimally solved by finding a shortest path in the solution graph of the fractured polygons of P.*

Proof. With the given stitch candidates, the polygons of P are fractured into more polygons. According to Lemma 1, the optimal decomposition of the fractured polygons still exists in the corresponding solution graph. Similar to the proof of Lemma 2, we can find a shortest path (i.e., a decomposition with a minimum weighted sum of Eq. (1)) in the solution graph to optimally solve the layout decomposition problem. □

3.3 Hierarchical Method

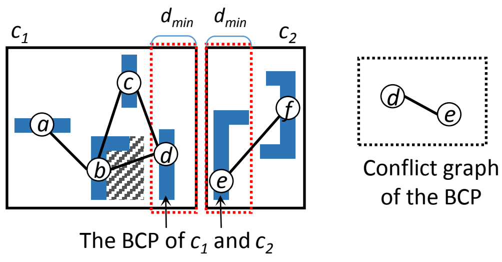
Since the concepts of conflict graph, polygon dummy extension, cutting line, cutting line set, and solution graph are all applicable to a standard cell as well, we also adopt a speedup method in a similar way as [8]. The solution graph for each cell in the cell library is first built and stored in a table. For a set of cells placed in a row, the solution graph of the layout in the row can be first constructed by sequentially copying the solution graph of each cell in the row directly from the one stored in the table according to the cell order from left to right. But there might be some additional potential conflicts introduced between two adjacent cells. In order to capture the additional potential conflicts, the *boundary conflicting polygon set* of two adjacent cells is introduced and defined as follows.

Definition 6 (Boundary Conflicting Polygon Set).

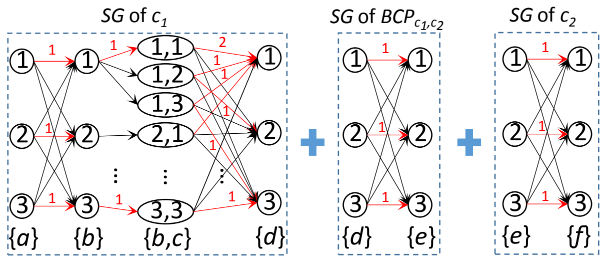
The boundary conflicting polygon set (BCP) for two adjacent cells c_i and c_j is a subset of polygons in c_i and c_j, where c_i is to the left of c_j. A polygon p in c_i (c_j, respectively) will be in the BCP when p is within a distance of d_{min} from the right boundary of c_i (the left boundary of c_j, respectively).

Similarly, conflict graph, polygon dummy extension, cutting line, cutting line set, and solution graph are also applicable to a BCP. Therefore, we then combine the solution graphs of two adjacent cells with the solution graph of the corresponding BCP to capture the potential conflicts between the cells. Fig. 5 gives an example to show how to construct the solution graph for two adjacent cells. We first construct the conflict graph for the BCP of two adjacent cells c_1 and c_2 (see Fig. 5(a)). Based on the conflict graph, we then construct the solution graph for the BCP in Fig. 5(b). Finally, by combining the three solution graphs of c_1, c_2, and the BCP, the final solution graph (see Fig. 5(c)) is done. Note that the solution graph of the BCP for each pair of adjacent cells with a fixed distance between them can be also built and stored in the table in advance for speedup.

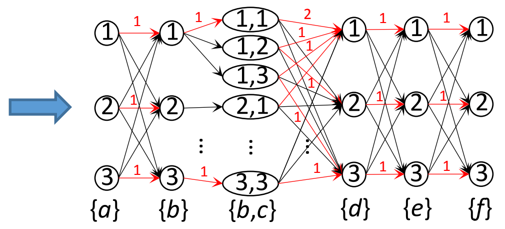
For a routed design, there might be some M1 wire connections overlapping with a set of cells in a row so that the solution graphs, which are associated with the cells in the set and are stored in the table, can not be directly used. The cells covered by the wire connections can be divided into disjoint sets of consecutive cells, where two cells covered by



(a) Two cells, c_1 and c_2, placed in a row, and the conflict graph of the BCP, BCP_{c_1,c_2}=$\{d, e\}$, of c_1 and c_2



(b) Three solution graphs (SGs) of c_1, c_2, and BCP_{c_1,c_2}, where the solution graph of c_1 is partly shown



(c) Final solution graph for the layout

Figure 5: Illustration of constructing the solution graph for a layout of two adjacent cells.

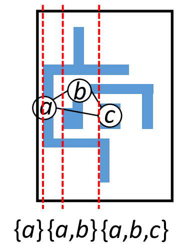
a wire connection must be in the same set. Similar to the algorithm in [8], each set can be considered as one large cell when constructing the solution graph by our approach.

3.4 Power/Ground Rail

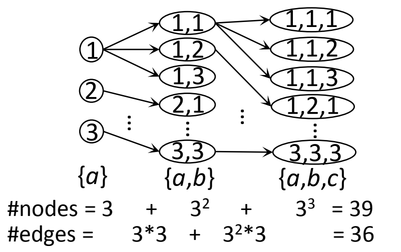
Though all the cutting lines must cut the power (VDD) and the ground (VSS) rails in a row, it is not necessary to add both VDD and VSS into each cutting line set. We can pre-assign colors to VDD and VSS (the colors are either the same or different) before constructing the solution graph SG. The coloring conflict between VDD/VSS and one polygon p is captured in SG by increasing the weight by 1 for each node v of the cutting line set s, where the cutting line of s is created by p and p's color specified by the coloring solution of v causes a coloring conflict with VDD/VSS. In addition, we only need to try one color combination for each pre-coloring case of VDD and VSS when constructing SG, because the colors of SG can be rotated to get another solution graph for a different color combination. Finally, the best decomposition result of the row can be chosen from the two pre-coloring cases for VDD and VSS. For the hierarchical method (mentioned in Section 3.3), we therefore need to build a table for one pre-coloring case and another table for the other pre-coloring case.

3.5 Parallelism for Multiple Rows

Given a design with m rows, the TPL decomposition problem of each row can be independently and simultaneously



(a) A standard cell



(b) The corresponding solution graph

Figure 6: The solution graph of a standard cell.

solved by our approach, because there is no coloring conflict between the polygons in different rows. Thus, we can directly parallelize our algorithm by assigning the TPL decomposition problem of each row to one thread. For each row, it will select a best decomposition solution from the two pre-coloring cases for VDD and VSS, and the result of each row is just combined together directly from the first row to the last row. If two adjacent rows have different colors for the sharing VDD or VSS connection, the colors of the second row need to be rotated to make the color of the sharing VDD or VSS connection the same.

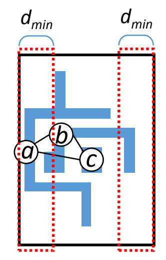
4. GRAPH REDUCTION TECHNIQUES

In this section, we will present several techniques to reduce the size of solution graph so as to accelerate our approach without loss of decomposition quality. For easy understanding, there are no stitches in the layouts and no weights in the solution graphs of the remaining figures, and the solution graphs are partly presented. Besides, the discussions in Sections 4.1 $\sim$ 4.3 are for one pre-coloring case of VDD and VSS.

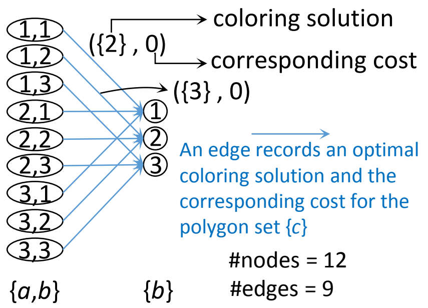
4.1 Simple Solution Graph

Fig. 6(b) shows the solution graph of a standard cell in Fig. 6(a), where the solution graph has totally 39 nodes and 36 edges. With a careful observation on the three cutting line sets $\{a\}$, $\{a, b\}$, and $\{a, b, c\}$, $\{a, b, c\}$ includes all the polygons in the cell so that there is no need to keep the other cutting line sets when constructing the solution graph. As a result, the numbers of nodes and edges in the graph are reduced to 27 and 0, respectively. Thus, we can find that the original construction method of solution graph is not clever and could create too many nodes and edges which can be abandoned to reduce the graph size. Hence, we devise a new graph model called *simple solution graph* to simplify the original solution graph. Let P_{c_i} be the set of polygons in the M1 layer of a cell c_i in the cell library. The *left boundary set* (LBS) (*right boundary set* (RBS), respectively) of c_i is a subset of P_{c_i} such that each polygon in the LBS (RBS, respectively) is within a distance of d_{min} from the left (right, respectively) boundary of c_i. Let LBS_{c_i} and RBS_{c_i} denote the LBS and the RBS of c_i, respectively. $OP_{c_i} = P_{c_i} \setminus (LBS_{c_i} \cup RBS_{c_i})$ is the set of the *other polygons* (OP) of c_i. Without loss of generality, we assume that LBS_{c_i} and RBS_{c_i} are not empty sets.

Definition 7 **(Simple Solution Graph).** *The simple solution graph of P_{c_i} is defined as a solution graph for $LBS_{c_i} \cup RBS_{c_i}$. Each edge e connecting from a node of LBS_{c_i} to a node of RBS_{c_i} records an optimal coloring solution of OP_{c_i} and a cost of w computed by Eq. (1) for the optimal coloring solution, given the two associated coloring solutions of the nodes of e.*



(a) LBS and RBS



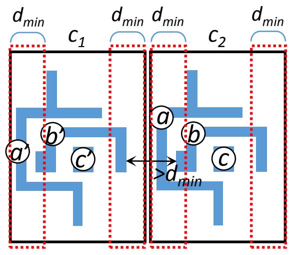
(b) Simple solution graph

Figure 7: The simple solution graph of the cell in Fig. 6(a).

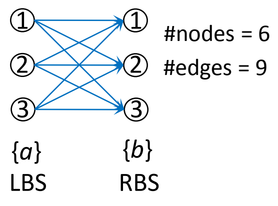
It can be observed that LBS_{c_i} and RBS_{c_i} are basically the sets of boundary conflicting polygons close to c_i's left boundary and right boundary, respectively, but OP_{c_i} is not. Because OP_{c_i} does not cause any conflicts with the polygons in other cells, we can try to remove the coloring solutions of OP_{c_i} from the solution graph, which will not affect the decomposition quality. For the cell in Fig. 6(a), its simple solution graph (see Fig. 7(b)) is constructed based on its LBS and RBS (see Fig. 7(a)). Each edge between the nodes of the LBS and the RBS is associated with an optimal coloring solution and the corresponding cost for OP_{c_i}. The number of nodes enumerated by both the LBS and RBS is 12 and the number of edges between the nodes is 9. As compared with the solution graph in Fig. 6(b), we can see that the simple solution graph gets a significant reduction in both the numbers of nodes and edges.

To find the coloring solution of OP_{c_i} associated with each edge in the simple solution graph, we construct the solution graph SG for OP_{c_i} and copy the nodes of both LBS_{c_i} and LBS_{c_i} to SG. Besides, a node of LBS_{c_i} to a node of the first cutting line set of SG (a node of the last cutting line set of SG to a node of RBS_{c_i}, respectively) is connected by an edge when the two nodes are compatible. Then, we find the shortest path in SG from a node of LBS_{c_i} to a node of RBS_{c_i} to get an optimal coloring solution of OP_{c_i} when the colors of $LBS_{c_i} \cup RBS_{c_i}$ are specified by the coloring solutions of the two nodes. Actually, we can easily find the shortest paths from a node v of LBS_{c_i} to all nodes of RBS_{c_i} by running one breadth-first search starting from v till all nodes of RBS_{c_i} are visited, since SG is a directed acyclic graph.

Moreover, we find that some polygons in LBS_{c_i} (RBS_{c_i}, respectively) can be removed and added into OP_{c_i} to further reduce the size of the simple solution graph without loss of decomposition quality. To determine if a polygon p of c_i can be removed from LBS_{c_i} or RBS_{c_i}, we can check if there is no potential conflict between p and any polygon of each cell c_j in the library when c_i and c_j are abutted. For example, there are two cells c_1 and c_2 in Fig. 8(a), and polygon b in c_2 has no potential conflict with the polygons in c_1. Thus, b is removed from LBS_{c_2}, and the simple solution graph of c_2 as shown in Fig. 8(b) will be constructed based on the new LBS_{c_2} and the original RBS_{c_2}. Consequently, the graph is further simplified relative to the one in Fig. 7(b).



(a) Two abutted cells c_1 and c_2

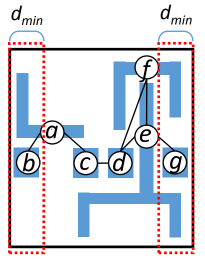


(b) Simple solution graph of c_2

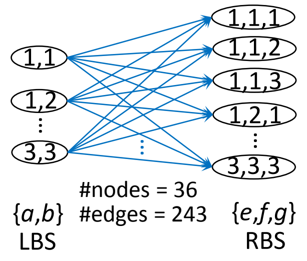
Figure 8: The simple solution graph of the cell in Fig. 6(a) with the reduced LBS.

4.2 Reduced Simple Solution Graph

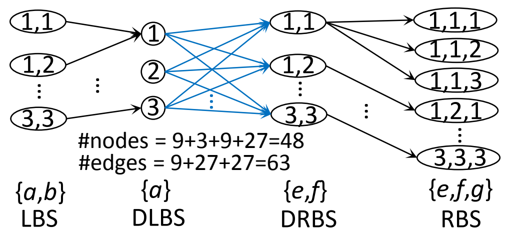
Since the simple solution graph of a cell is constructed based on the LBS and the RBS, the amount of edges of the graph could dramatically increase as the number of polygons in both the LBS and the RBS increases. If the LBS has n polygons and the RBS has m polygons, the number of the edges between the nodes in the graph could be up to 3^{m+n}. Fig. 9(a) and Fig. 9(b) give an example showing the total number of polygons in both the LBS and the RBS is 5 and the number of edges is 3^5=243. To reduce the number of edges, we create the *dummy LBS* (abbreviated as DLBS) and the *dummy RBS* (abbreviated as DRBS) for a LBS and a RBS, respectively. The DLBS (DRBS, respectively) of c_i is a subset of LBS_{c_i} (RBS_{c_i}, respectively) in which each polygon in the DLBS (DRBS, respectively) has a potential conflict with at least a polygon in RBS_{c_i} (LBS_{c_i}, respectively) or OP_{c_i}. Besides, if a polygon is in both LBS_{c_i} and RBS_{c_i}, it will be also added to the DLBS and DRBS of c_i. Let $DLBS_{c_i}$ and $DRBS_{c_i}$ denote the DLBS and DRBS of c_i, respectively. Now, the *reduced simple solution graph* of c_i is constructed based on the four polygon sets LBS_{c_i}, $DLBS_{c_i}$, $DRBS_{c_i}$, and RBS_{c_i} in the order of LBS_{c_i}, $DLBS_{c_i}$, $DRBS_{c_i}$, RBS_{c_i}. Different from a simple solution graph, the optimal coloring solution and the cost of OP_{c_i} are recorded on each edge between the nodes of $DLBS_{c_i}$ and $DRBS_{c_i}$. Note that we assign weights 0 to the nodes of both $DLBS_{c_i}$ and $DRBS_{c_i}$ and to the edges between the nodes of LBS_{c_i} and $DLBS_{c_i}$ ($DRBS_{c_i}$ and RBS_{c_i}, respectively).



(a) Conflict graph

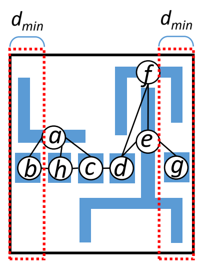


(b) Simple solution graph

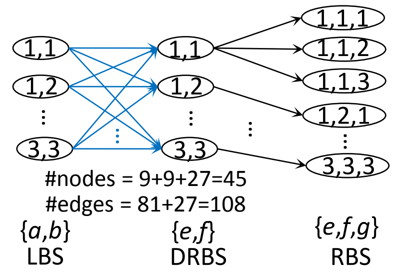


(c) Reduced simple solution graph

Figure 9: The reduced simple solution graph of a cell.



(a) Conflict graph



(b) Reduced simple solution graph

Figure 10: The reduced simple solution graph of a cell.

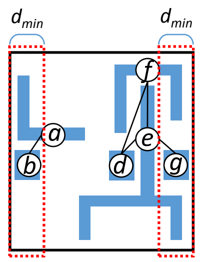
The reason why adding two dummy sets $DLBS_{c_i}$ and $DRBS_{c_i}$ is because the amount of the coloring solutions of OP_{c_i} (i.e., the edges between LBS_{c_i} and RBS_{c_i} in the simple solution graph) can be further reduced when $DLBS_{c_i} \subset LBS_{c_i}$ or $DRBS_{c_i} \subset RBS_{c_i}$. Besides, the decomposition quality for c_i still remains unaffected since there is no potential conflict between a polygon in $LBS_{c_i} \setminus DLBS_{c_i}$ ($RBS_{c_i} \setminus DRBS_{c_i}$, respectively) and a polygon in OP_{c_i}. Fig. 9(c) demonstrates the reduced simple solution graph for the simple solution graph in Fig. 9(b). Apparently, the total graph size is reduced after adding the DLBS and the DRBS though the number of nodes is more than before.

There are three scenarios for the construction of a reduced simple solution graph, where both the DLBS and DRBS are not empty sets for the scenarios 1 and 2:

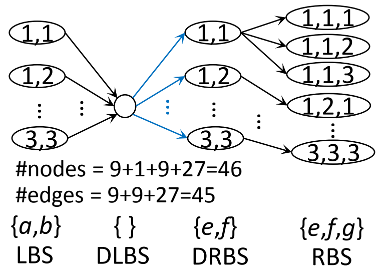
1. DLBS is not equal to LBS and DRBS is not equal to RBS:
   As the aforementioned description for Fig. 9(c), both the DLBS and the DRBS are added into the graph for reducing the graph size.

2. DLBS is equal to LBS or DRBS is equal to RBS:
   When the DLBS (DRBS, respectively) is equal to the LBS (RBS, respectively) for a cell, we do not create the DLBS(DRBS, respectively) to avoid enlarging the graph size instead. Fig. 10 shows an example in which the DLBS is equal to the LBS for the cell so that the DLBS is excluded from the reduced simple solution graph.

3. DLBS and/or DRBS is an empty set:
   If the DLBS (DRBS, respectively) for a cell is an empty set, a pseudo node with weight of 0 is created for the DLBS (DRBS, respectively). As can be seen from Fig. 11, the DLBS of the cell is empty since there is no conflict between the polygons of the LBS and the others. Accordingly, we build the reduced solution graph with the pseudo node of the DLBS for the cell as shown

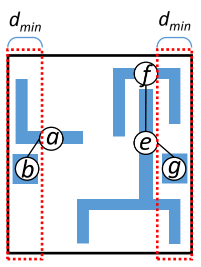


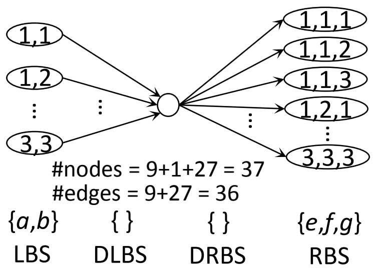
(a) Conflict graph



(b) Reduced simple solution graph

Figure 11: The reduced simple solution graph of a cell.





(a) Conflict graph (b) Reduced simple solution graph

Figure 12: The reduced simple solution graph of a cell.

in Fig. 11(b). Moreover, if both the DLBS and the DRBS are empty, we can merge their pseudo nodes into one, and there will be only one coloring solution of the OP recorded on the merged pseudo node (see the example in Fig. 12).

4.3 Reduced Simple Solution Graph for BCP

As we redesign the solution graph to be the simple solution graph for a cell, the solution graph for a BCP cannot be directly used to combine with a simple solution graph. Thus, we also devise a reduced simple solution graph for the BCP of two adjacent cells c_i and c_j. RBS_{c_i} of the left cell c_i and LBS_{c_j} of the right cell c_j, respectively, can be considered as the LBS and the RBS of the BCP, while the OP of the BCP is an empty set. As a result, the reduced simple solution graph of the BCP can be easily constructed in the same way as before. Fig. 13 gives an example. The simple solution graph for the BCP of c_1 and c_2 is shown on the left side of Fig. 13(b). Obviously, there is no potential conflict between RBS_{c_1} and LBS_{c_2}, and hence we can create two empty sets, $DRBS_{c_1}$ and $DLBS_{c_2}$, to reduce the graph size (i.e., the third scenario in Section 4.3.2). Finally, the reduced simple solution graph for the BCP of c_1 and c_2 is shown on the right side of Fig. 13(b).

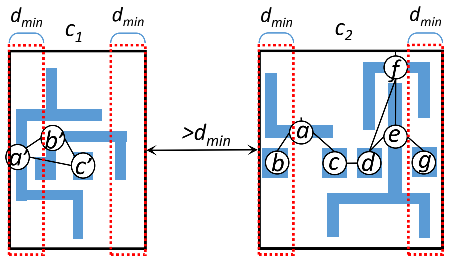
4.4 Overall Approach

Given a cell library and a set of stitch candidates for each cell, we first off-line build a look-up table in each pre-coloring case for VDD and VSS, where the table contains the reduced simple solution graph for each single cell and the set of reduced simple solution graphs for the BCPs of each cell pair.[2] For a row r of n cells, we can construct a solution graph for r under each pre-coloring case, by combining the reduced simple solution graphs stored in the corresponding table for each cell and each pair of adjacent cells in r. The shortest path in the solution graph corresponds to the best decomposition for a pre-coloring case, and therefore the decomposition of r is the one with less cost between the two pre-coloring cases.

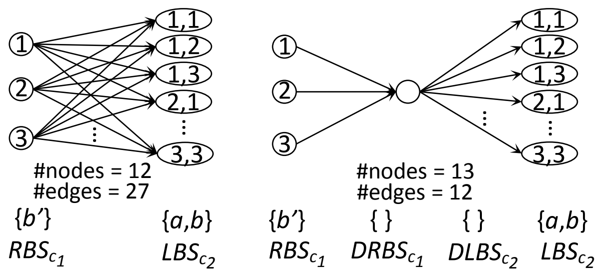
5. EXPERIMENTAL RESULTS

The proposed algorithm has been implemented in C++ language, and all experiments were conducted on a Linux

[2]The number of reduced simple solution graphs in the set is $k + 1$ for the BCPs of two adjacent cells, where k is the minimum number of sites (white space) between the cells such that there is no potential conflict between the cells. The reduced simple solution graph for the BCP of the cells with q sites between the cells is the same as the one with k sites if $q > k$.



(a) Two adjacent cells c_1 and c_2



(b) Simple solution graph on the left side and reduced simple solution graph on the right side for the BCP of c_1 and c_2

Figure 13: The reduced simple solution graph of a BCP.

workstation with 2.0 GHz Intel Xeon CPU and 96 GB memory. Two benchmark suites provided by the authors of [11] and [10] are used. The first one is a set of placed circuits from the OpenSPARC T1 designs, and the other is the set of ISCAS-85&89 routed circuits. The parameter setting such as d_{min}, α, and β for each benchmark is the same as [11] and [10]. Finally, The stitch candidates of all test cases are generated by [4], which is the same as [10].

In Tables 1 $\sim$ 3, we report the value of Eq. (1) (in the "Cost" column), the number of conflicts (in the "#C" column), the number of stitches (in the "#S" column), and the runtime measured in second on t threads (in #T-t). "Ours" and "Ours-GR" represent our algorithms with and without graph reduction, respectively.

To the best of our knowledge, the algorithm in [10] is the current state-of-the-art TPL decomposer. For making fair comparisons, we ran the decomposer provided by [10] on our machine.[3] Note that the released TPL decomposer does not consider the layout density issue. First, we compare our algorithm with the TPL decomposer [10] on the OpenSPARC T1 designs whose layouts are much denser than the ISCAS-85&89 circuits. As can be seen from Table 1, our decomposer with graph reduction successfully obtained the decomposition for each test case in a few seconds; however, the one in [10] could not complete in the time limit (i.e., five hours) for almost half of the test cases. In addition, our approach produced much better decomposition quality and ran much faster for every test case that [10] could also complete in the time limit; though we used more stitches, the total cost is still less than that produced by [10].

We also make an analysis for the graph reduction methods proposed in this paper. The comparisons between our approaches with and without graph reduction are shown in Table 2. We can find that the amount of nodes and the amount of edges are averagely reduced by 94% and 77%, respectively. Consequently, there are 76% $\sim$ 80% runtime

[3]Since the algorithm in [10] cannot be executed in a multi-threaded mode, we ran it on a single thread.

improvements on average in the multi-threaded modes with 4, 2, and 1 threads.

To further observe the decomposition quality that our decomposer can achieve, we compare with the algorithms in [8] and [10] on the frequently used benchmarks ISCAS-85&89 circuits in Table 3. Note that in Table 3, the results of [10] are obtained by executing the decomposer of [10] on our machine and are slightly different from those reported in [10], while the results of [8] are directly quoted from [8]. Apparently, it supports again that our approach can find the decomposition without any conflict for a layout if the layout is decomposable (i.e., conflict-free), when compared with [8]. Moreover, as compared with [10], our approach also produced better or same results in terms of the amount of conflicts, the amount of stitches, and, consequently, the total cost.

6. CONCLUSIONS

In this work, we extend an existing approach to solve a row-structure TPL layout decomposition problem. Several methods to substantially reduce the graph size and hence to accelerate the extended approach are also presented. The experimental results show that our algorithm is not only effective but also efficient.

7. REFERENCES

[1] V. O. Anton, N. Peter, H. Judy, G. Ronald, and N. Robert. Pattern split rules! a feasibility study of rule based pitch decomposition for double patterning. In *Proc. of SPIE*, volume 6730, 2007.

[2] S.-Y. Fang, Y.-W. Chang, and W.-Y. Chen. A novel layout decomposition algorithm for triple patterning lithography. In *Proc. of DAC*, pages 1185–1190, 2012.

[3] J. Kuang, W.-K. Chow, and E. F. Y. Young. Triple patterning lithography aware optimization for standard cell based design. In *Proc. of ICCAD*, pages 108–115, 2014.

[4] J. Kuang and E. F. Y. Young. An efficient layout decomposition approach for triple patterning lithography. In *Proc. of DAC*, pages 1–6, 2013.

[5] Y.-H. Lin, B. Yu, D. Z. Pan, and Y.-L. Li. Triad: A triple patterning lithography aware detailed router. In *Proc. of ICCAD*, pages 123–129, 2012.

[6] H. Z. Q. Ma and M. D. F. Wong. Triple patterning aware routing and its comparison with double patterning aware routing in 14nm technology. In *Proc. of DAC*, pages 591–596, 2012.

[7] H. Tian, Y. Du, H. Zhang, Z. Xiao, and M. D. F. Wong. Triple patterning aware detailed placement with constrained pattern assignment. In *Proc. of ICCAD*, pages 116–123, 2014.

[8] H. Tian, H. Zhang, Q. Ma, Z. Xiao, and M. D. F. Wong. A polynomial time triple patterning algorithm for cell based row-structure layout. In *Proc. of ICCAD*, pages 57–64, 2012.

[9] Y. Xu and C. Chu. A matching based decomposer for double patterning lithography. In *Proc. of ISPD*, pages 121–126, 2010.

[10] B. Yu, Y.-H. Lin, G. Luk-Pat, D. Ding, K. Lucas, and D. Z. Pan. A high-performance triple patterning layout decomposer with balanced density. In *Proc. of ICCAD*, pages 163–169, 2013.

Table 1: Comparisons with a state-of-the-art TPL decomposer [10] on the OpenSPARC T1 designs

Test Case	[10]				Ours-GR			
	Cost	#C	#S	T-1	Cost	#C	#S	T-4
alu-70	408.1	336	721	213	286.5	185	1015	4
alu-80	N/A	N/A	N/A	>18000	474.4	353	1214	8
alu-90	N/A	N/A	N/A	>18000	520.6	398	1226	5
byp-70	656.9	502	1549	830	636.0	437	1990	10
byp-80	1076.5	897	1795	2544	831.2	612	2192	14
byp-90	N/A	N/A	N/A	>18000	1012.4	780	2324	22
div-70	726.2	568	1582	913	620.7	424	1967	12
div-80	703.7	545	1587	1182	578.4	383	1954	11
div-90	656.9	502	1549	830	528.9	338	1909	11
ecc-70	N/A	N/A	N/A	>18000	134.6	89	456	3
ecc-80	207.3	159	483	485	197.5	140	575	4
ecc-90	N/A	N/A	N/A	>18000	275.4	214	614	6
efc-70	N/A	N/A	N/A	>18000	218.4	143	754	3
efc-80	N/A	N/A	N/A	>18000	247.1	172	751	4
efc-90	302.8	241	618	202	209.0	134	750	3
ctl-70	248.5	215	335	970	209.3	170	393	3
ctl-80	N/A	N/A	N/A	>18000	212.6	167	456	4
ctl-90	187.5	150	375	537	152.4	111	414	4
top-70	N/A	N/A	N/A	>18000	2320.3	1754	5663	15
top-80	3221.1	2774	4471	3807	2702.0	2118	5840	31
top-90	3506.0	3002	5040	9305	3233.9	2618	6159	36

Table 2: Comparisons of our approaches on the OpenSPARC T1 designs

Test Case	Ours					Ours-GR				
	#V	#E	T-1	T-2	T-4	#V	#E	T-1	T-2	T-4
alu-70	7.67e+06	5.45e+07	29	16	11	5.28e+05	1.55e+07	8	4	4
alu-80	7.91e+06	5.82e+07	34	18	20	5.37e+05	2.44e+07	13	7	8
alu-90	7.90e+06	5.66e+07	32	20	19	5.41e+05	2.49e+07	13	7	5
byp-70	2.75e+07	2.16e+08	114	75	43	1.75e+06	4.42e+07	23	13	10
byp-80	2.78e+07	2.19e+08	121	76	64	1.76e+06	5.56e+07	30	17	14
byp-90	2.81e+07	2.22e+08	120	75	69	1.77e+06	7.18e+07	38	22	22
div-70	2.02e+07	1.60e+08	90	50	39	1.20e+06	3.70e+07	19	11	12
div-80	2.02e+07	1.62e+08	94	51	60	1.20e+06	4.39e+07	24	14	11
div-90	2.01e+07	1.60e+08	88	55	26	1.20e+06	3.74e+07	20	11	11
ecc-70	1.02e+07	8.93e+07	51	28	25	5.06e+05	1.12e+07	5	3	3
ecc-80	1.03e+07	8.99e+07	53	29	27	5.07e+05	1.47e+07	7	4	4
ecc-90	1.04e+07	8.96e+07	55	30	22	5.11e+05	1.83e+07	9	5	6
efc-70	9.80e+06	6.08e+07	33	19	19	3.27e+05	1.13e+07	5	3	3
efc-80	9.80e+06	6.15e+07	34	18	15	3.29e+05	1.27e+07	6	4	4
efc-90	9.84e+06	6.13e+07	33	18	19	3.27e+05	1.20e+07	6	3	3
ctl-70	1.26e+07	1.03e+08	57	39	20	5.83e+05	1.57e+07	7	4	3
ctl-80	1.25e+07	1.02e+08	58	33	35	5.81e+05	1.47e+07	7	4	4
ctl-90	1.25e+07	1.01e+08	57	32	22	5.79e+05	1.46e+07	7	4	4
top-70	6.73e+07	4.87e+08	272	155	105	3.72e+06	9.67e+07	48	26	15
top-80	6.76e+07	4.92e+08	291	154	152	3.74e+06	1.08e+08	54	30	31
top-90	6.79e+07	4.97e+08	292	156	89	3.77e+06	1.20e+08	63	34	36
Ratio	1	1	1	1	1	0.06	0.23	0.20	0.20	0.24

Table 3: Comparisons with two previous works [8] and [10] on the ISCAS85&89 circuits

Test Case	[8]			[10]			Ours		
	Cost	#C	#S	Cost	#C	#S	Cost	#C	#S
C432	N/A	N/A	N/A	0.4	0	4	0.4	0	4
C499	0	0	0	0	0	0	0	0	0
C880	0.7	0	7	0.7	0	7	0.7	0	7
C1355	0.3	0	3	0.3	0	3	0.3	0	3
C1908	0.1	0	1	0.1	0	1	0.1	0	1
C2670	0.6	0	6	0.6	0	6	0.6	0	6
C3540	N/A	N/A	N/A	1.9	1	9	1.8	1	8
C5315	N/A	N/A	N/A	0.9	0	9	0.9	0	9
C6288	N/A	N/A	N/A	22.2	1	212	20.5	0	206
C7552	N/A	N/A	N/A	2.6	0	26	2.3	0	22
S1488	0.2	0	2	0.3	0	3	0.2	0	2
S38417	N/A	N/A	N/A	25.7	20	57	24.4	19	54
S35932	N/A	N/A	N/A	52	46	60	48	44	40
S38584	N/A	N/A	N/A	49.1	36	131	47.6	36	116
S15850	N/A	N/A	N/A	45.9	34	119	44.1	34	98
Ratio				1.06	1.03	1.12	1	1	1

[11] B. Yu, X. Xu, J.-R. Gao, and D. Z. Pan. Methodology for standard cell compliance and detailed placement for triple patterning lithography. In *Proc. of ICCAD*, pages 349–356, 2013.

[12] B. Yu, B. Zhang, D. Ding, and D. Z. Pan. Layout decomposition for triple patterning lithography. In *Proc. of ICCAD*, pages 1–8, 2011.