A Polynomial Time Triple Patterning Algorithm for Cell Based Row-Structure Layout

Haitong Tian†, Hongbo Zhang*, Qiang Ma*, Zigang Xiao†, Martin D.F. Wong†
†Department of Electrical and Computer Engineering, University of Illinois at Urbana Champaign
*Synopsys Inc., USA
Email: †{htian3,zxiao2,mdfwong}@illinois.edu,*{hongbo.zhang,qma}@synopsys.com

***Abstract*—As minimum feature size keeps shrinking, and the next generation lithography (e.g, EUV) further delays, double patterning lithography (DPL) has been widely recognized as a feasible lithography solution in 20nm technology node. However, as technology continues to scale to 14/10nm, DPL begins to show its limitations and usually generates too many undesirable stitches. Triple patterning lithography (TPL) is a natural extension of DPL to conquer the difficulties and achieve a stitch-free layout decomposition. In this paper, we study the standard cell based row-structure layout decomposition problem in TPL. Although the general TPL layout decomposition problem is NP-hard, in this paper we will show that for standard cell based TPL layout decomposition problem, it is polynomial time solvable. We propose a polynomial time algorithm to solve the problem optimally and our approach has the capability to find all stitch-free decompositions. Color balancing is also considered to ensure a balanced triple patterning decomposition. To speed up the algorithm, we further propose a hierarchical algorithm for standard cell based layout, which can reduce the run time by 34.5% on average without sacrificing the optimality. We also extend our algorithm to allow stitches for complex circuit designs, and our algorithm guarantees to find optimal solutions with minimum number of stitches.**

I. Introduction

As the technology advances, feature size of the chips continues to scale down. However, advancements in lithography technology has been slow and lagged behind. Due to the limitations of current 193nm ArF immersion lithography, the advancement of IC industry to 14nm/10nm technology node has become a challenge. Although different types of next generation lithography techniques have been discussed for years, such as extreme ultra-violet (EUV) lithography, E-beam direct write and nano-imprint techniques [1] [2], the most promising printing technique that will be used in 20nm/14nm technology node is still 193nm immersion lithography with multiple patterning techniques.

The key idea of multiple pattering lithography is to use several exposure processes for a single layer. Typically, the patterning techniques can be classified as: double patterning lithography (DPL, also known as litho-etch-litho-etch technique), triple patterning lithography (TPL, also known as litho-etch-litho-etch-litho-etch technique) and self-aligned double pattering (SADP). Due to the difficulties of bridging the mask rules and design rules in SADP [3] [4], DPL is now considered as the key enabling technique for 20nm technology node. In DPL, patterns on a single layer patterns would be assigned to two different masks to double the printing pitch. In DPL, the color assignment is usually done by applying a minimum spacing rule, and any features that are closer than d_{min} (the minimum spacing) will have to be assigned different colors. Fig. 1(b) shows an example of DPL decomposition. Because every two of them conflict with each other, a stitching is needed and feature a has to be further sliced into two parts, a_1 and a_2, to resolve coloring conflicts. Although it is always preferred to minimize the stitch number during the DPL decomposition process, stitches in DPL are usually inevitable, especially in the circumstances of high density layout. Those stitches potentially cause yield lost and increase manufacturing cost [5] [6] [7].

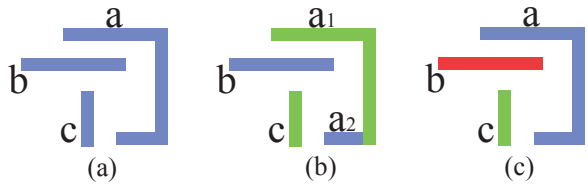

Fig. 1. (a) A simple layout (b) Patterning solution using double patterning lithography (DPL) (c) Patterning solution using triple patterning lithography (TPL). Polygons with different colors mean that they appear in different masks.

Compared to the DPL, TPL uses three masks for pattern assignment. Therefore, we can have more flexibility on color assignment and less conflicts among features. For the same layout in Fig. 1 (a), a stitch-free decomposition can be easily achieved using TPL as shown in Fig. 1 (c). Using different colors representing different masks, the TPL layout decomposition problem can be formulated as a three coloring problem, which is NP-complete. Yu et al. [8] made the first contribution in TPL decomposition with an ILP-based approach, and further proposed a semidefinite programming based approximation algorithm to deal with dense layouts. However, the ILP-based algorithm is expensive while the modified semidefinite

This work was partially supported by the National Science Foundation under grant CCF-1017516 and a grant from the semiconductor Research Corporation (SRC).

programming is losing the optimality. A graph based heuristic is proposed in [9], but it cannot guarantee to find a solution without resorting to an exponential algorithm. TPL problem is also studied in [10]. However, they also fail to guarantee to find an solution if one exists. Moreover, more stitches are introduced compared with the results in [8].

In this paper, we propose a polynomial time exact algorithm to find triple patterning decompositions of a standard cell based layout. Our contributions can be summarized as follows:

- We propose a polynomial time exact algorithm to optimally solve standard cell based row-structure TPL layout decomposition problem, and our algorithm has the capability to find all stitch-free decompositions.
- Color balancing is considered to achieve a balanced layout decomposition.
- We further improve our algorithm by first preprocessing each standard cell and then decomposing the whole layout on cell level. Experimental results show that this improved algorithm reduces the run time by 34.5% on average without sacrificing the optimality.
- To deal with more complex designs, we further extend our approach to accommodate stitches. Our extension is very efficient, and guarantees to find an optimal solution using the minimum number of stitches.
- Our approach is highly scalable, and can be easily migrated to parallel implementations to further reduce the run time.

The rest of the paper is organized as follows: some preliminaries of the TPL problem are discussed in section II. Our basic algorithm will be presented in section III. The extended algorithm is discussed in section IV. Section V shows the experimental results, followed by a conclusion in section VI.

II. Preliminaries

Preliminaries of standard cell based layout decomposition are introduced here, including introductions to standard cell based layout and the problem definition.

A. Standard Cell Based Row-Structure Layout

In standard cell based designs, designers are given a library of pre-designed standard cells. All standard cells in the library have the same height, with power and ground tracks going from the very left to the very right. A layout consists of multiple rows, and in each row, the cells are aligned with power and ground connecting each other. A type of cells may appear multiple times within a standard cell row.

In the 14/10nm technology node, TPL are need for the densest layer with finest features – most likely including gate, low level interconnect layers. For the gate and low level interconnect layers except metal 1 (M1) which have preferred/required directions defined, solving triple patterning problem could be trivial by modified track-coloring assignment. The most difficult part of TPL decomposition comes from M1 layer. For M1 layer, the most commonly seen properties are as follows:

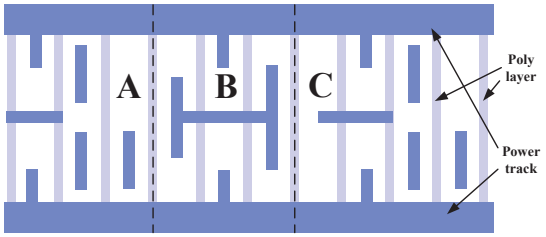

Fig. 2. Layout of part of a standard cell row. Three cells, A, B, and C lies in the standard cell row. Only poly layer and metal 1 layer are shown here for simplicity.

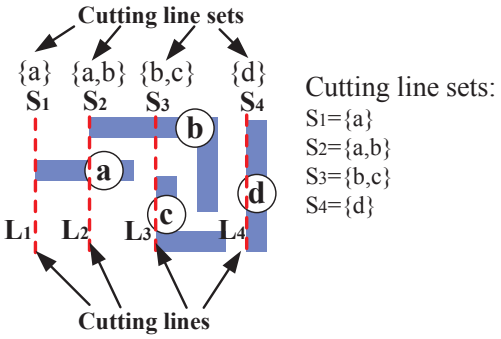

Fig. 3. Cutting lines and cutting line sets

- Power and ground tracks connect all cells in M1 layer in a row from left to right. Limited number of tracks are available between the power and ground tracks.
- Power and ground tracks are usually several times thicker than the finest features in M1 layer, which can perfectly isolate the influences between different rows.
- Wires have no preferred directions.
- Most wires are defined locally inside the cells, with few connecting different cells in the same row.

A sample standard cell based circuit layout is shown in Fig. 2. There are three cells in the layout. Power track, which refers to the power and ground connections, is on metal 1 layer. Higher metal layers are not shown here for simplicity.

B. Problem Definition

Given a M1 layer layout, our objective is to find a legal triple patterning decomposition of the power track layer while balancing the area utilization in the three masks.

III. A Polynomial Time Optimal Algorithm

In the following ,we use different colors to denote different masks. Polygons with the same color will appear in the same mask. In this section, we will introduce our polynomial time optimal algorithm. Based on coloring of standard cell rows, we propose a hierarchical approach to further accelerate our algorithm.

A. Basic Terminologies

Some terminologies used in the algorithm are first introduced here.

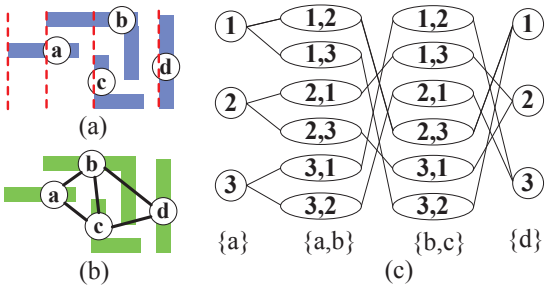

Fig. 4. (a) Input layout. (b) Constraint graph. (c) Solution graph (different numbers here denote different colors).

Definition 1 (cutting line): A *cutting line* is defined as a vertical line going from the top of the standard cell row to the bottom of it.

Definition 2 (cutting line set): A *cutting line set* is defined as the set of polygons which intersect with the same cutting line.

Let's associate each polygon in one row with a cutting line with the same x coordinate of its left boundary, and eliminate the redundant ones. We then obtain a set L of n cutting lines with $L = \{L_1, L_2, ..., L_n\}$. Note that n is at most the number of polygons in this row. Let's assume the cutting lines in L are sorted in nondecreasing order with respect to their x coordinates, i.e. $x(L_i) \leq x(L_j)$ if $i \leq j$. Each cutting line L_i is associated with a cutting line set S_i, which consists of all polygons intersecting with cutting line L_i.

Consider the example shown in Fig 3, there are four cutting lines L_1, L_2, L_3 and L_4, which are shown in red dashed lines. Their corresponding cutting line sets are $S_1 = \{a\}$, $S_2 = \{a, b\}$, $S_3 = \{b, c\}$, and $S_4 = \{d\}$ respectively.

To capture all coloring conflicts among the polygons and all legal solutions of a layout, two graphs, constraint graph and solution graph, are used in our algorithm.

Definition 3 (constraint graph): The constraint graph is defined as an undirected graph where the nodes represent polygons in a given layout, and where an edge means that the distance of the two polygons it connects are within the distance threshold d_{min} (the minimum spacing rule).

Fig. 4(a) shows a simple layout with 4 polygons, and the corresponding constraint graph is shown in Fig. 4(b). If two nodes are connected in the constraint graph, they cannot be assigned the same color in a legal layout decomposition.

Definition 4 (solution graph): The solution graph is a directed graph where each node records a legal coloring solution of a cutting set, and where an edge exists between two nodes which belong to adjacent cutting lines if the coloring solutions of the two nodes are compatible to each other.

For each cutting line set S_i, all the coloring solutions can be generated by simply enumerating all possible coloring assignments. For each of the coloring solution of S_i, a node is created in the solution graph. Denote the set of nodes generated from the coloring solutions of S_i as N_i, and $N_i = \{N_i^1, N_i^2, ..., N_i^q\}$, where $q \leq 3^t$, and t is the maximum number of tracks in this row. For any node N_i^j and N_{i+1}^k, an edge is added to connect the two nodes if the two coloring solutions do not conflict with each other.

A simple example is shown in Fig. 4. There are four polygons in the layout. The constraint graph is shown in Fig. 4(b). There are four cutting lines in the layout, which are shown in red dotted lines. For the first cutting line, which is the left most one, the cutting line set includes only polygon a. It has three coloring solutions: 1, 2, and 3, which are denoted by three nodes in the solution graph. For the second cutting line, the cutting line set includes polygons a and b. Similarly, its coloring solutions are denoted as six nodes. Edges are added if two nodes are compatible with each other. The same thing is done for the third and fourth cutting lines. Fig. 4(c) shows the solution graph of the layout in Fig. 4(a).

B. Polygon Dummy Extension

In the constraint graph, a polygon may conflict with several other polygons. It is necessary to consider all conflicting polygons together to ensure a valid decomposition. However, those polygons are usually distributed in different cutting lines.

For the example shown in Fig. 5(a), there is only one polygon intersecting with each cutting line. The corresponding solution graph is shown in Fig. 5(c). A path from the left most to the right most of the solution graph corresponds to a patterning solution. For example, path (1,2,1,2) means that polygon a and c can be colored using color "1", while b and d can be colored using color "2". This solution is illegal since polygon a and c cannot be assigned the same color, which can be clearly seen from the constraint graph shown in Fig. 5(b). This is because conflicts between non-adjacent cutting lines are neglected, which leads to color assignment violations.

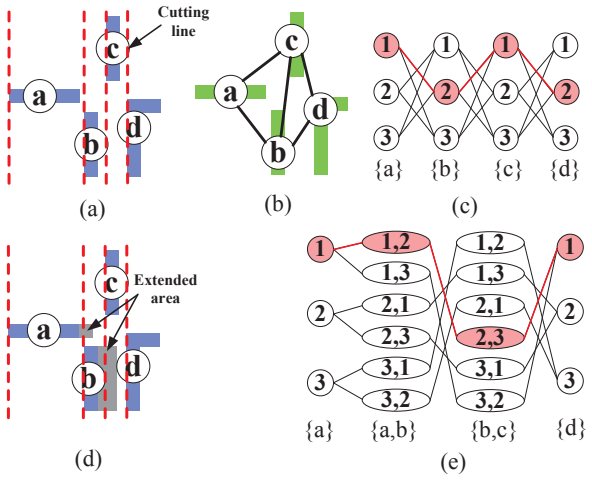

Fig. 5. (a) Input layout. (b) Constraint graph. (c) Solution graph without polygon dummy extension. (d) Input layout after polygon dummy extension. (e) Solution graph with polygon dummy extension.

Based on the constraint graph, we propose a *polygon dummy extension* method to capture the conflicts of the polygons between multiple cutting lines. For each polygon in the layout, we locate its conflicting polygon that has the largest left x coordinate. Denote its left x as x_0. Then, the right boundary of the current polygon is virtually extended to $x_0 - \delta$, where δ is a very small value and is used to ensure that the new right

boundary does not intersect with the cutting line $x = x_0$. After extending the right boundaries of the polygons, it is guaranteed that for any polygon in a cutting line set S_i, all its conflicting polygons (with smaller x coordinates) appear in the previous cutting line set S_{i-1}. Based on the modified layout, we go through each cutting line L_i and find the corresponding cutting line set S_i. The solutions of S_i can be computed. For each solution of S_i, which is represented by a node in the solution graph, its compatible nodes in the solution graph are identified and an edge is added between the two nodes. Repeating the above steps, we can build a solution graph for a given layout.

For polygon a in Fig. 5, its conflicting polygons are polygons b and c. Denote the x coordinates of polygon b and c as x_b and x_c respectively. Since the left boundary of polygon c has a larger x coordinate, the right boundary of polygon a is virtually extended to $x_c - \delta$, where δ is chosen to be small enough such that polygon a intersects with the second cutting line, but not the third one. Fig. 5(c) is the layout after polygon dummy extension, and Fig. 5(d) is the corresponding solution graph. We can see that based on the modified layout, every path in Fig. 5(d) is guaranteed to be a valid solution.

Theorem 1: There is a valid triple patterning decomposition if and only if there is a path going from the left most of the solution graph to the right most of it.

Proof: We prove the theorem by mathematical induction. The base case is that for the first cutting line L_1, all paths reaching nodes in N_1 in the solution graph are legal, since these paths contain only one node, which must be legal. Now assume for cutting line L_i, all paths reaching nodes in N_i in the solution graph are legal. Consider the next cutting line L_{i+1}, the set of solution nodes are N_{i+1}. For the set of solution nodes in N_i and N_{i+1}, edges are only added when two nodes are compatible with each other. Using polygon dummy extension, it is guaranteed that for any polygon in cutting line set S_{i+1}, all its conflicting polygons appear in the previous cutting line set S_i. This means that the solutions nodes in N_{i+1} are only affected by the nodes in N_i. Since all paths reaching nodes in N_i are already legal, adding one more legal edge to those paths guarantees that the new paths are legal.

Similarly, the reverse case can also be proved by mathematical induction. Assume triple patterning solutions exist for a given layout. For any known solution, the coloring of all the polygons are known. The base case is that for the first cutting lines L_1, we can always find a node $N_1^{k^1}$ from the node sets N_1, in which all polygons in the cutting line sets S_1 are assigned the same color as they are in the legal TPL solution. Now consider the cutting lines L_i and L_{i+1}. Denote the compatible node we find in N_i and N_{i+1} as $N_i^{k^i}$ and $N_{i+1}^{k^{i+1}}$ respectively. Since the solutions of $N_i^{k^i}$ and $N_{i+1}^{k^{i+1}}$ are contained in the legal TPL solution, by definition, there must be an edge connecting the two nodes in the solution graph. All connecting nodes forms a path in the solution graph. The proof is complete. ■

C. Power and Ground Connections

For standard cell based designs, each cell has its power and ground connections on top and bottom that goes from the very left to the very right. Since the power and ground connections appear in all cutting lines, we can pre-color them before processing other polygons. They can either be assigned the same color, or different colors. There is no need to try all combinations, since we can generate other solutions from existing ones by rotating colors. For example, if we already get a solution graph G, we can easily get another solution graph G' by changing color 1 to color 2, color 2 to color 3 and color 3 to color 1. In the algorithm, both ways of pre-coloring by assigning same and different colors to the power track are tried, and for each way of pre-coloring, the algorithm is able to find all possible coloring solutions. Therefore, the pre-coloring step does not affect the optimality of our approach. Our row based algorithm is shown in Algorithm 1.

Algorithm 1: Coloring of a standard cell row

```
1  begin
2      Initialize solution graph G to be empty;
3      P ← all polygons in a standard cell row;
4      X ← x coordinates of the left boundaries of all polygons in P;
5      Sort X in increasing order;
6      w ← size of X;
7      for i ← 1 to w do
8          Set cutting line x = X_i;
9          Find all polygons intersecting with x = X_i;
10         Compute solutions for these polygons;
11         Add the solutions into the solution graph G;
12     end
13     Find a path from the left most side to the right most side of G;
14 end
```

Solution graphs of adjacent rows can be combined together based on the power and ground connections. If two rows share the same power connections, we require the coloring of the power connections in the two solution graphs to be the same. The same principle applies for ground connections.

D. Algorithm Complexities

Assume that there are n polygons in a standard cell row, and there are at most t horizontal tracks available for routing per standard cell row. Note that t can be regarded as constant under a particular manufacturing technology. Thus, each cutting line intersects at most t polygons. Due to the dummy extension of polygons, we need to enumerate the solutions of at most $2t$ polygons per cutting line. The number of solutions is thus upper bounded by 3^{2t}. Since the cutting lines are based on the left boundaries of the polygons, there are at most n cutting lines. For the solution nodes of two successive cutting lines, 3^{4t} operations are needed to connect the compatible nodes. The overall time complexity of our approach is $O((3^{4t} + 3^{2t})n)$. Note that $3^{4t} + 3^{2t}$ is constant here. Therefore, the overall complexity is $O(n)$. This shows that standard cell based TPL problem is polynomial time solvable.

Note that this is a very pessimistic upper bound. In practice, there are seldom $2t$ polygons intersecting with a cutting line. Even there are $2t$ polygons, the number of solutions are far less than 3^{2t}, as lots of solutions can be pruned away based on the constraint graph. Moreover, different rows can be solved independently. For each row, our algorithm guarantees to find all possible solutions. Therefore, the parallel implementation

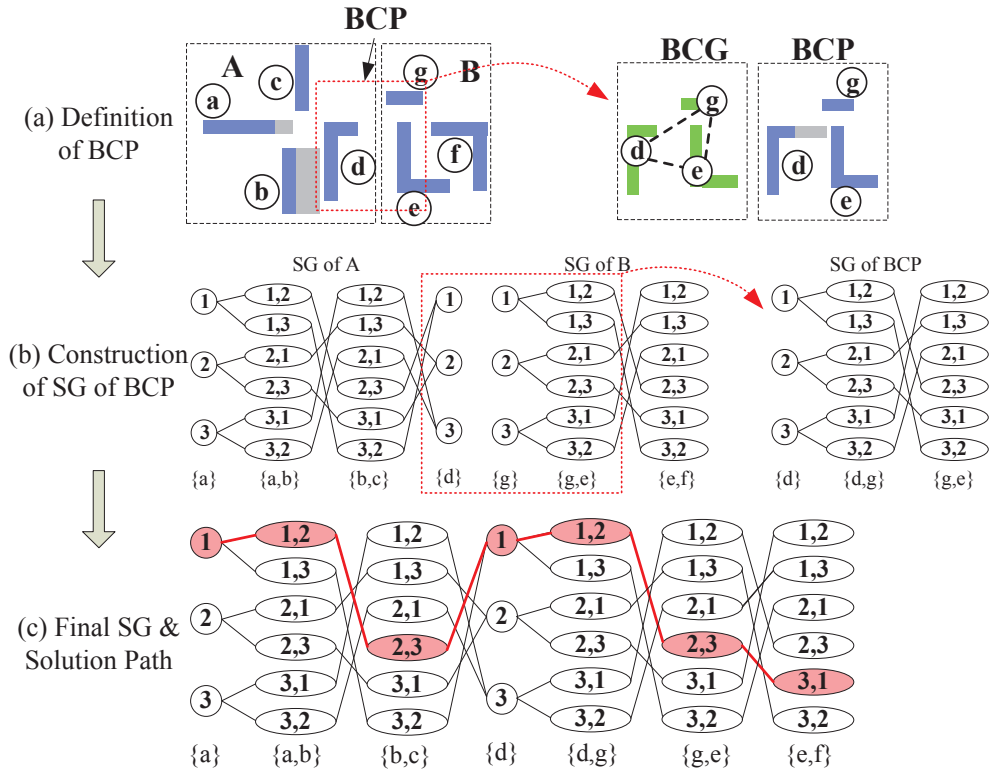

Fig. 6. Illustration of BCP without connections. 'SG' denotes 'solution graph'. (a) Three polygons, d, e, and g, appear in the BCP. (b) Solution graph of cell A, B, and the BCP. (c) Final solution graph and a sample solution path, which is shown in red color.

does not affect the optimality of our algorithm, and the solution graph will be the same as that without parallel implementation.

E. Hierarchical Speedup Approach

For standard cell based circuit designs, millions of elements in a chip are typically composed of hundreds of basic cells in the standard cell library. If the solution graphs of all basic cells are precomputed, they can be reused in the higher hierarchy to color a given layout. In practice, the number of elements in a chip usually overwhelms the number of basic cells in the standard cell library. Thus, cell based hierarchical approach is expected to greatly accelerate the run time compared with coloring a given layout as one large graph.

For each cell in the cell library, the constraint graph and solution graph are constructed. These two graphs can be constructed the same way as that for the standard cell rows. To connect different cells in a standard cell row, connections between cells are considered.

1) Boundary Polygons between Adjacent Cells: Adjacent cells in a standard cell row may introduce additional coloring conflicts. If polygons of adjacent cells are within the distance d_{min}, they have to be assigned different colors. To capture such constraints, we introduce additional conflicting edges recording all coloring conflicts between these polygons. Define *boundary conflicting polygons* (BCP) as the set of polygons within a distance of d_{min} from the boundaries of the adjacent cells. For the BCPs of two adjacent cells, all conflicting edges are identified. Based on the constraint graphs of the two cells and these conflicting edges, a *boundary constraint graph* (BCG) is constructed, which represents all conflicting relations between the BCPs. Polygon dummy extension is performed based on BCG. After that, we go over the left boundaries of polygons in BCP, and compute a solution graph for BCP. By combining the solution graphs of the two standard cells and the solution graph of BCPs, we can build a larger solution graph, which contains all possible legal patterning solutions for the adjacent cells.

A simple example with two cells is illustrated in Fig 6. For cell A and cell B, their BCPs are first identified. Based on the BCPs and their constraint graph, polygon dummy extension is performed. Then, the solution graph for BCPs can be computed. By combining the solution graphs for cell A, B, and the BCPs, the final solution graph can be computed as shown in Fig 6(c).

2) Connections between Cells: Additional coloring conflicts also include connections between different cells. If there are connections between two polygons for adjacent cells, they should be assigned the same color. Note that BCP also contains all connecting polygons in adjacent cells. When enumerating solutions for BCPs, connected polygons are assigned the same color. For connection that goes across several cells, we can group up the cells it covers and treat them as one large cell. The row based method can be used directly to compute the constraint graph and solution graph of the large cell. Then, the cell is treated as a regular cell in the algorithm.

An example of boundary connection between adjacent cells

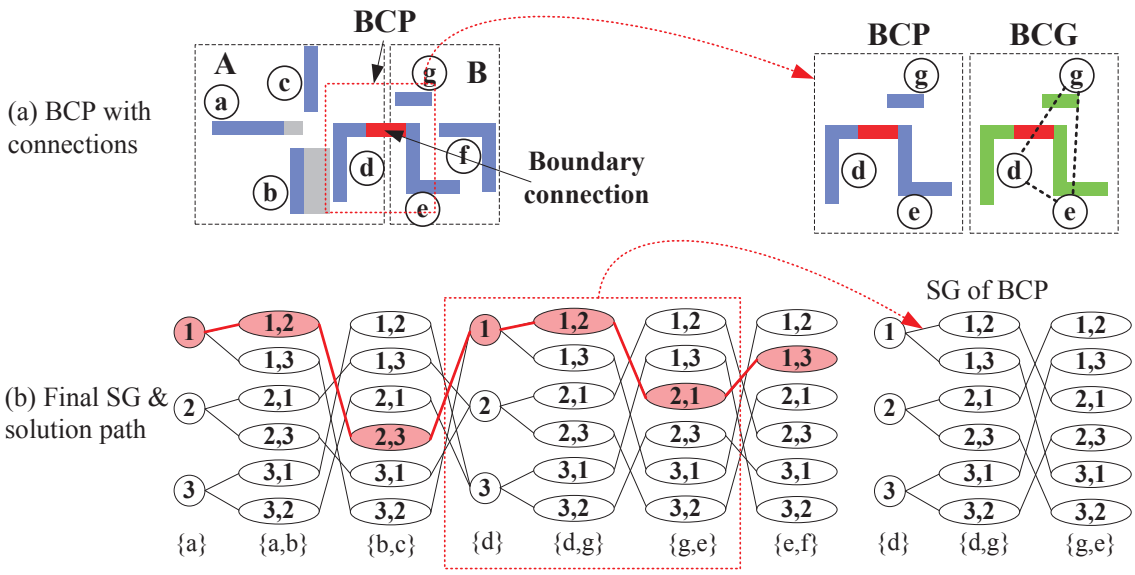

Fig. 7. Illustration of BCP with connections. 'SG' denotes 'solution graph'. (a) Three polygons, d, e, and g, appear in the BCP. (b) Final solution graph and a sample solution path, which is shown in red color.

Algorithm 2: Hierarchical Speedup Approach

```
1  begin
2      C_lib ← all standard cells in the library;
3      C_row ← all standard cells in a row;
4      foreach Cell C_i in C_lib do
5          Build constraint graph G_i;
6          Build solution graph S_i;
7      end
8      m ← number of long connections in C_row ;
9      for j ← 1 to m do
10         C_new ← all the cells the jth connection covers;
11         Build its constraint graph G_new;
12         Build its solution graph S_new;
13         C_lib ←C_new;
14     end
15     w ← size of C_row ;
16     for j ← 1 to w do
17         Build partial solution graph G for the first jth cells in C_row;
18     end
19     Find a path from the left most side to the right most side of G;
20 end
```

is shown in Fig. 7 to illustrate the above ideas. For the two connecting polygons d and e, they are assigned the same color in the solution graph. The flow of our hierarchical approach is shown in Algorithm 2.

F. Color Balancing

For a good TPL layout decomposition, the area utilization of the three masks should be balanced so that none of them dominate the other ones. These well balanced decompositions fully take advantage of each mask, and maximally benefit from the manufacturing process. The area of polygons on each mask is used as the metric to evaluate the quality of a patterning solution. After obtaining the solution graph, a balanced patterning solution is chosen as follows.

Three variables are used here, each representing the total area of polygons with the same color. The solution graph is scanned from the leftmost cutting line to the rightmost cutting line. In each step, the color with the largest polygon areas is assigned the lowest priority, while the color with the smallest polygon areas has the highest priority. New polygons will be assigned the color that is legal and has the highest priority. Note that new polygons can only be assigned the color that is compatible with the color assignments of previous polygons.

IV. TPL Incorporating Stitches

Since stitches potentially introduce many undesirable effects and will increase the manufacturing cost, it is always preferable to design a circuit layout which is three colorable. However, in practice, there may be complex cell layouts which are impossible to decompose into three masks without introducing stitches. For those layouts, we first find a set of legal stitch positions and decompose the original polygons into a set of smaller stitch polygons. Then, a modified solution graph is constructed based on our previous optimal TPL algorithm. Lastly, a shortest path algorithm is invoked to get an optimal solution with the minimum number of stitches.

A. Stitch Position Identification

The same method is adopted as what is used in [8] to identify all stitch candidate positions. For a given layout, layout graph simplification technique [8] is first performed to find the polygons that potentially require stitches. Then, node projection is invoked to find all projected segments on those polygons. Based on the projection results, all legal stitch positions are computed. Note that a stitch position is legal if it does not intersect with any projected segments. If the vertical stitch is illegal, the horizontal one will be tried. The original polygons are decomposed into a set of new polygons by the stitches. The constraint graph can be constructed based on those decomposed polygons. Besides the constraint edges,

there also exists stitch edges in the constraint graph. There is a stitch edge connecting two nodes if a stitch candidate exists between the two corresponding polygons.

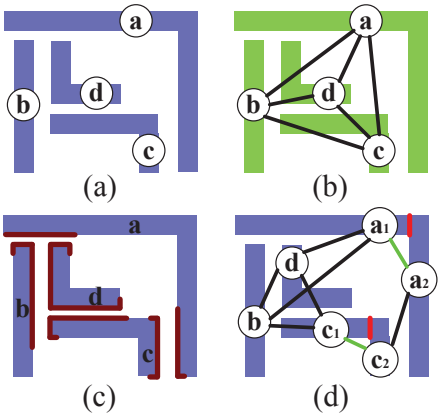

Fig. 8. (a) Input layout. (b) Constraint graph of the input layout in (a). (c) The node projection. Projection edges are shown in bolded brown lines. (d) Constraint graph after stitch decomposition. Polygon a is decomposed into polygons a_1 and a_2. Polygon c is decomposed into polygons c_1 and c_2. The stitch positions are shown using bolded red lines. Stitch edges are shown in bolded green lines.

A simple example is shown in Fig. 8. Obviously, the layout shown in Fig. 8(a) is not three colorable since the constraint graph of the four nodes forms a complete graph, which is shown in Fig. 8(b). The node projection result is shown in Fig. 8(c). Two legal stitch positions are computed based on the node projection results, which are shown in Fig. 8(d). We can see that after adding the two stitches, legal triple patterning decompositions can be achieved.

B. Coloring a Standard Cell Row

After finding all stitch positions, coloring of the new layout is similar to the TPL algorithm without stitches. The solution graph here is different from that without stitches. Weights are assigned to edges in the solution graph. If two nodes corresponding to cutting line set S_i and S_{i+1} requires c stitches[1], the weight of the edge will be assigned as c. Similarly, hierarchical approach can also be adopted to further reduce the run time.

C. Finding an Optimal Decomposition

Once we construct the solution graph, finding an optimal solution is quite straightforward. A shortest path algorithm can be employed to get an optimal decomposition with the minimum number of stitches. Note that the way we construct the solution graph is intrinsically beneficial for the shortest path formulation. If we go through the solution graph from left to right based on the cutting lines, all the nodes we visited are already in topological order. Unlike the ILP formulation which is very slow and the semidefinite programming formulation which loses its optimality in [8], the shortest path formulation is very fast and guarantees to find an optimal solution.

[1] c is an integer, which reflects how many stitches are needed from one coloring solution to another.

Similar to the TPL algorithm without stitches, the TPL algorithm with stitches also runs in polynomial time. The time complexity is $O(n+s)$, where n is the number of polygons and s is the number of stitch candidates in a give layout. Details are not shown here due to page limits.

V. Experimental Results

The algorithm is implemented in C++ and run on a Linux server with 4GB RAM and a 2.8 GHZ CPU. NanGate FreePDK45 Generic Open Cell Library [11] is used to generate all benchmarks. We randomly select the standard cells in the cell library, and align them adjacently in different rows of a chip. The size of the standard cells are proportionally scaled down to reflect 14nm technology node. Connections between adjacent cells are randomly generated between their boundary constraint polygons. d_{min} is set to be 82nm. Wires on metal one layer are used for all experiments, as more wires including power tracks are on layer one and they also have more complex shapes compared with other layers.

A. Results of the Basic TPL Algorithm

Five benchmarks, $C1$ to $C5$, are generated with increased number of polygons. The detailed results of our approach are shown in Table I. For the largest benchmark with over 26 million polygons, the run time is within an hour.

TABLE I
Triple Decomposition Results

Test Cases	n	Balanced Area Ratio	Random Area Ratio	T (s)
C1	106690	1 : 1 : 1	1 : 0.27 : 0.23	10
C2	674841	1 : 1 : 1	1 : 0.26 : 0.24	66
C3	2695803	1 : 1 : 1	1 : 0.25 : 0.24	264
C4	10782073	1 : 1 : 1	1 : 0.25 : 0.24	1062
C5	26949406	1 : 1 : 1	1 : 0.26 : 0.24	2655

Note 1: "n" denotes the number of polygons in the benchmark.
Note 2: The run time is the results of our hierarchical algorithm.
Note 3: The area of the power track is subtracted.

TABLE II
Run Time Comparisons

Test Cases	n	Tracks	T1 (s)	T2 (s)	Improve (%)
C1	106690	143	10	16	35.6
C2	674841	358	66	101	34.2
C3	2695803	715	264	403	34.4
C4	10782073	1429	1062	1610	34.0
C5	26949406	715	2655	4028	34.1
Ave.	8241763	672	812	1232	34.5

Note: Column of "T1" shows the run time of our hierarchal algorithm, while column "T2" shows the results of our row based algorithm.

The third column shows the results using our color balancing technique, while the results of a random color selection approach is shown in column four. For the random color selection approach, we go through the solution graph once and randomly assign the polygons a valid color. With the coloring balancing technique, we can achieve a much balanced decomposition compared with the result of a random color

selection approach. Run time is shown in the last column in Table I. We can see that the run time is linearly correlated to the number of polygons in the benchmark, which further verifies that the algorithm is a polynomial time algorithm.

We also compare the run time between our basic approach and our hierarchal approach, and the results are shown in Table II. Our proposed hierarchal cell based algorithm can further improve the run time by 34.5% on average without affecting the optimality of our algorithm. This clearly verifies the effectiveness of the hierarchal cell based algorithm.

B. TPL Algorithm with Stitches

Five benchmarks, C_6 to C_{10}, are also generated using more complex standard cells. The results shown in Table III are based on the hierarchical implementation. The number of polygons, the number of tracks, the number of stitch candidate, the number of final stitches, and the run time are shown in column 2, 3, 4, 5, and 6 respectively. For the largest benchmark with over 17 million polygons, the run time is within three hours. Note that with our shortest path formulation, we guarantee that the number of stitches computed is minimum.

TABLE III
TRIPLE DECOMPOSITION RESULTS WITH STITCHES

Test Cases	n	Tracks	Stitch Candidates	Stitches	T (s)
C6	179201	143	78102	3420	80
C7	904292	322	394349	17146	388
C8	4449681	715	1940587	83916	1900
C9	10031115	1072	4382524	188854	4277
C10	17813611	1429	7778321	334642	7613

C. Comparisons with previous works

We also compared our results with the previous works in [8] and [10] using the ISCAS-85 & 89 benchmarks provided by the authors of [8]. The same settings are using as what are used in [10]. The detailed results are shown in Table IV.

Note that if the number of conflicts (shown in column named "C") is not zero, it means that the algorithm fails to find a legal decomposition. For all solved benchmarks in [8], we are able to find legal decompositions with optimal number of stitches. Our algorithm further solved 4 more benchmarks which the SDP based algorithm cannot handle. For the algorithm in [10], they use a different stitch identification method, thus solving benchmark C432. However, the stitch identification method in [10] can be easily incorporated into our framework to compute the optimal solutions. Moreover, we can solve 4 benchmarks where their approach fails. This clearly verifies the effectiveness of our approach.

TABLE IV
COMPARISONS WITH PREVIOUS WORKS

	SDP Based [8]			Algorithm in [10]			Ours		
Test Cases	C	S		C	S		C	S	
C432	3	1	×	0	6	√	–	–	×
C499	0	0	√	0	0	√	0	0	√
C880	1	6	×	1	15	×	0	7	√
C1355	1	6	×	1	7	×	0	3	√
C1908	0	1	√	1	0	×	0	1	√
C2670	2	4	×	2	14	×	0	6	√
C3540	5	6	×	2	15	×	–	–	×
C5315	7	7	×	3	11	×	–	–	×
C6288	82	131	×	19	341	×	–	–	×
C7552	12	15	×	3	46	×	–	–	×
S1488	1	1	×	0	4	√	0	2	√
S38417	44	55	×	20	122	×	–	–	×
S35932	93	18	×	46	103	×	–	–	×
S38548	63	122	×	36	280	×	–	–	×
S15850	73	91	×	36	201	×	–	–	×

Note: "C" means the number of conflicts. If $C \neq 0$, no legal solutions are found (marked with ×). "S" denotes the number of stitches.

VI. CONCLUSIONS

In this paper, we propose a polynomial time algorithm to optimally solve the standard cell based TPL problem. Our approach is highly scalable and can be implemented in parallel. Color balancing is considered to achieve a valid and balanced solution. Our approach has the capability to find all stitch-free decompositions for a standard cell based layout. To further reduce the run time, we propose a hierarchical approach, which can reduce the run time by 34.5% on average without sacrificing the optimality of the algorithm. To cope with more complex designs, we extend our approach to allow stitches. Our approach guarantees to find an optimal solution with the minimum number of stitches. Our approach is expected to bring convenience to industry on TPL problem and relieve the manufacturing bottlenecks on 14/10nm technologies.

REFERENCES

[1] H. Levinson, "Extreme ultraviolet lithography's path to manufacturing," *Journal of Micro*, 2009.

[2] B. Lin *et al.*, "Successors of arf water-immersion lithography: Euv lithography, multi-e-beam maskless lithography, or nanoimprint?" in *J Micro/Nanolith. MEMS MOEMS*. SPIE, 2008.

[3] H. Zhang, Y. Du, M. Wong, and R. Topaloglu, "Self-aligned double patterning decomposition for overlay minimization and hot spot detection," in *ACM/EDAC/IEEE Design Automation Conference*. IEEE, 2011.

[4] H. Zhang, Y. Du, M. Wong, and R. Topaloglu, "Hot spot detection for indecomposable self-aligned double patterning layout," in *Proceedings of SPIE*, 2011.

[5] J. Yang and D. Pan, "Overlay aware interconnect and timing variation modeling for double patterning technology," in *IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2008, pp. 488–493.

[6] A. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design*, NOV. 2008, pp. 465–472.

[7] D. Pan, J. Yang, K. Yuan, M. Cho, and Y. Ban, "Layout optimizations for double patterning lithography," in *IEEE International Conference on ASIC*. IEEE, 2009, pp. 726–729.

[8] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Pan, "Layout decomposition for triple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2011.

[9] Q. Li, P. Ghosh, D. Abercrombie, P. LaCour, and S. Kanodia, "14nm m1 triple patterning," in *Proceedings of the SPIE*, 2012.

[10] S. Fang, Y. Chang, and W. Chen, "A novel layout decomposition algorithm for triple patterning lithography," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 1185–1190.

[11] *Si2 Open Cell Library, available on line.*, http://www.si2.org/openeda.si2.org/projects/nangatelib.