Methodology for Standard Cell Compliance and Detailed Placement for Triple Patterning Lithography

Bei Yu, Xiaoqing Xu, Jhih-Rong Gao, David Z. Pan
ECE Department, University of Texas at Austin, TX, USA
{bei, xiaoqingxu, jrgao, dpan}@cerc.utexas.edu

ABSTRACT

As the feature size of semiconductor process further scales to sub-16nm technology node, triple patterning lithography (TPL) has been regarded one of the most promising lithography candidates. M1 and contact layers, which are usually deployed within standard cells, are most critical and complex parts for modern digital designs. Traditional design flow that ignores TPL in early stages may limit the potential to resolve all the TPL conflicts. In this paper, we propose a coherent framework, including standard cell compliance and detailed placement to enable TPL friendly design. Considering TPL constraints during early design stages, such as standard cell compliance, improves the layout decomposability. With the pre-coloring solutions of standard cells, we present a TPL aware detailed placement, where the layout decomposition and placement can be resolved simultaneously. Our experimental results show that, with negligible impact on critical path delay, our framework can resolve the conflicts much more easily, compared with the traditional physical design flow and followed layout decomposition.

1. INTRODUCTION

As the feature size of semiconductor process technology nodes further scales to sub-16nm, triple patterning lithography (TPL) is regarded as one of the most promising lithography candidates, along with extreme ultra-violet lithography (EUVL), directed self-assembly (DSA), and electron beam lithography (EBL) [1, 2]. TPL is a natural extension along the paradigm of double patterning lithography (DPL), which has been pushed to its limit in sub-16nm, to introduce better printability [3].

To deploy TPL process, *layout decomposition* is usually applied to divide the initial layout into three masks. Then each mask is implemented through one exposure-etch process, through which the layout can be produced. In initial layout, two features with distance less than minimum coloring distance d_{min} should be assigned into different masks. One *conflict* occurs when two features whose spacing is less than d_{min}. Sometimes the conflict can be also resolved by inserting *stitch* to split a feature into two touching parts.

TPL layout decomposition problem with conflict and stitch minimization has been well studied in the past few years [4–11]. However, most existing work suffers from one or more of the following drawbacks. (1) Because TPL layout decomposition problem is NP-hard [6], most of the decomposers are based on approximation or heuristic methods, thus some extra conflicts may be reported [8]. (2) For each design, since the library only contains fixed number of standard cells, layout decomposition would contain lots of redundant works. For example, if one cell is applied hundreds of times in a single design, it would be decomposed hundreds of times during layout decomposition. (3) Successfully carrying out these decomposition techniques requires the input layouts to be TPL-friendly. However, since all these decomposition techniques are applied at post-place/route stage, where all the design patterns are already fixed, they lack the abilities to resolve some native TPL conflict patterns, e.g., four-clique conflicts.

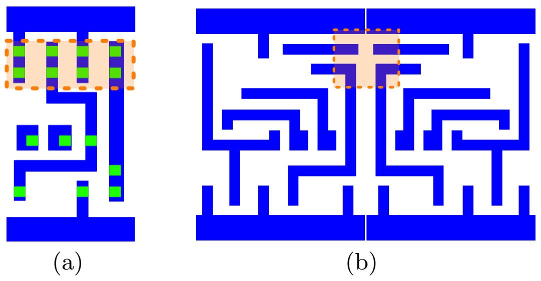

Figure 1: Native conflicts from (a) contact layer within a standard cell; (b) M1 layer between adjacent standard cells.

It is observed that the most hard-to-decompose patterns originate from contact and M1 layers. Fig. 1 shows two common native TPL conflicts in contact layer and M1 layer, respectively. As shown in Fig. 1(a), contact layout within the standard cell may generate some 4-clique patterns, which is indecomposable. Meanwhile, if placement techniques are not TPL friendly, some boundary metals may introduce native conflicts (see Fig. 1(b)). Since redesigning indecomposable patterns in the final layout requires high ECO efforts, generating TPL-friendly layouts, especially in the early design stage, becomes urgent and pivotal. Through these two examples, we can see that both TPL aware standard library design and TPL aware placement are necessary to avoid such indecomposable patterns in final layout.

Liebmann et al. in [12] proposed some guidelines to enable DPL friendly standard cell design and placement. Besides, there exist several placement studies toward different manufacturing process targets [13–15]. Recently [16, 17] proposed TPL aware detailed routing schemes. However, to our best knowledge, no previous work has addressed TPL compliance at standard cell or placement level.

In this paper, we present a systematic framework to seamlessly integrate TPL constraints in early design stages, comprehending standard cell conflict removal, standard cell precoloring and detailed placement together. Note that our framework is layout decomposition free, that is, the TPL aware detailed placement can generate optimized positions and color assignment solutions for all cells. Our main contributions are summarized as follows:

- We propose systematic standard cell compliance techniques for TPL and coloring solution generation.
- We study the standard cell pre-coloring problem, and propose effective methods.
- We present the first systematic study for the TPL aware ordered single row placement, where cell placement and

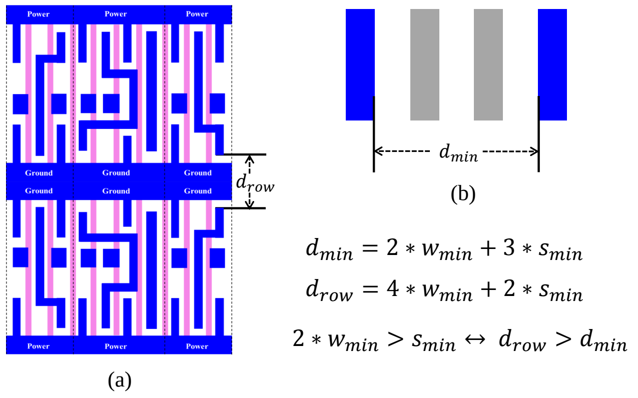
Power
Ground
d_{row}
d_{min}
(a)
(b)
$d_{min} = 2 * w_{min} + 3 * s_{min}$
$d_{row} = 4 * w_{min} + 2 * s_{min}$
$2 * w_{min} > s_{min} \leftrightarrow d_{row} > d_{min}$

Figure 2: (a) Minimum spacing between M1 wires among different rows. (b) Minimum spacing between M1 wires with the same color.

color assignment can be solved simultaneously.

- Our framework seamlessly integrate decomposition in each key step, therefore no additional layout decomposition is required.
- Experimental results show that our framework can achieve zero conflict, meanwhile can effectively reduce the stitch number.

The rest of the paper is organized as follows: Section 2 provides preliminaries and overview of our methodologies. Section 3 proposes standard cell modification to enable TPL friendly cell layout, with negligible timing impact. In Section 4 the pre-coloring techniques for each cell are proposed, followed by look-up table construction. Section 5 and Section 6 give details on our TPL aware detailed placement. Section 7 presents the experiment results, followed by conclusion in Section 8.

2. PRELIMINARIES

2.1 Row Structure Layout

Our framework assumes a row-structure layout, where cells in each row are with the same height, and power/ground rails are going from the very left to the very right (see Fig. 2(a)). Similar assumption was applied in row based TPL layout decomposition [8] as well. The minimum width of metal feature and the minimum spacing between neighboring metal features are denoted as w_{min} and s_{min}, respectively. Besides, we define the minimum spacing between metal features among different rows to be d_{row}. If we further analyze layout patterns in the library, it can be observed the width of power/ground rail is twice the width of metal wire within standard cells [18]. Under the row structure layout, we have the following lemma.

Lemma 1. *There is no coloring conflict between two M1 wires or contacts that are from different rows.*

PROOF. For TPL, the layout will be decomposed into three masks, which means layout features within minimum coloring distance will be assigned three colors to increase the pitch between neighboring features. Then, we can see from the Fig. 2, the minimum spacing between M1 features with the same color in TPL is $d_{min} = 2 \cdot w_{min} + 3 \cdot s_{min}$. We assume the worst case for d_{row}, which means the standard cell rows are placed as mirrored cells and allow for no routing channel. Thus, $d_{row} = 4 \cdot w_{min} + 2 \cdot s_{min}$. We should have $d_{row} > d_{min}$, which equals $2 \cdot w_{min} > s_{min}$. This condition can easily be satisfied for M1 layer. For the same reason, we can achieve similar conclusion for the contact layer. □

Based on the row-structure assumption, the whole layout can be divided into rows, and layout decomposition or coloring assignment can be carried out for each row. Without loss of generality, for each row the power/ground rails are assigned the color 1 (*default color*). Then the decomposed results for each row will not induce coloring conflicts among different rows. In other words, the coloring assignment results for each row can be merged together, without losing optimality.

2.2 Overall Design Flow

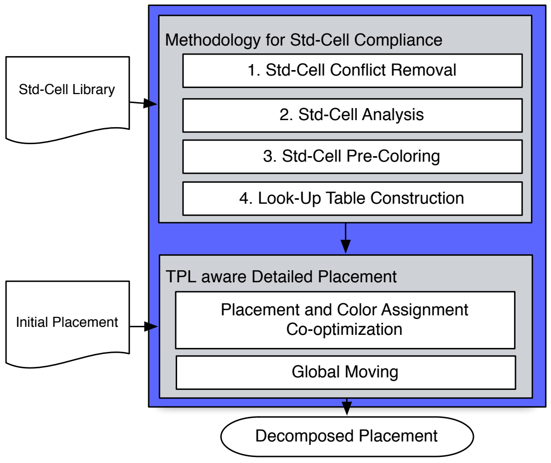
Methodology for Std-Cell Compliance
1. Std-Cell Conflict Removal
2. Std-Cell Analysis
3. Std-Cell Pre-Coloring
4. Look-Up Table Construction
Std-Cell Library
TPL aware Detailed Placement
Placement and Color Assignment Co-optimization
Global Moving
Initial Placement
Decomposed Placement

Figure 3: Overall flow of the methodologies for standard cell compliance and detailed placement.

The overall flow of our proposed framework is illustrated in Fig. 3. It consists of two stages: methodologies for standard cell compliance, and TPL aware detailed placement. The standard cell compliance techniques include standard cell conflict removal, timing analysis, standard cell pre-coloring, and lookup table generation. The standard cell compliance techniques ensure that, for each cell, TPL friendly cell layout and a set of pre-coloring solutions will be provided.

Note that since triple patterning lithography constraints are seamlessly integrated into our coherent design flow, we do not need a separate additional step of layout decomposition. In other words, the output of our framework is decomposed layouts that have resolved cell placement and color assignment simultaneously.

3. STANDARD CELL COMPLIANCE

It is observed that without considering TPL in standard cell design, the cell library may involve several cells with native TPL conflict (see Fig. 1 (a) for one example). The inner native TPL conflict cannot be resolved through either cell shift or layout decomposition. Since one cell may be applied many times in one single design, such inner native conflict may cause hundreds of coloring conflicts in final layout. To achieve TPL friendly layout after the physical design flow, we should first ensure the standard cell layout compliance for TPL. Specifically, we will manually remove all 4-clique conflicts through standard cell modification. Then, parasitic extraction and SPICE simulation are applied to analyze the timing impact for the cell modification.

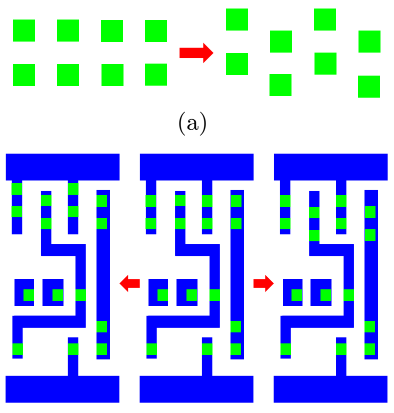

Figure 4: Contact layout modification to hexagonal packing. (a) The principle for contact shifting; (b) Demonstration of two options for contact shifting, with original layout in the middle, case 1 on the left and case 2 on the right.

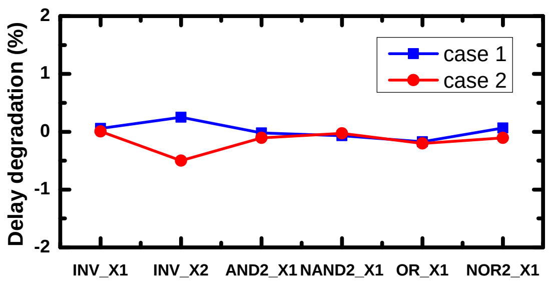

Figure 5: The timing impact from layout modification for different types of gates, including case 1 and case 2

3.1 Native TPL Conflict Removal

An example of native TPL conflict is illustrated in Fig. 4, where four contacts introduce an indecomposable 4-clique conflict structure. For such cases we modify the contact layout into hexagonal close packing [3], which also allows for the most aggressive cell area shrinkage for TPL friendly layout. Note that after modification, the layout still needs to satisfy the design rules. From the layout analysis of different cells, we have various ways to remove such 4-clique conflict. As shown in Fig. 4, with slight modification to original layout, we can either choose to move contacts connected with power or ground rails or shift contacts on the signal paths of the cell. We call these two options case 1 and case 2 respectively, both of which will lead to TPL friendly standard cell layout.

Generally, the cell layout design flexibility is beneficial for resolving conflicts between cells when they are placed next to each other. However, from a circuit designer's perspective, we want to achieve little timing variation among various layout styles of a single cell. Therefore, we need simulation results to demonstrate negligible timing impact from layout modification.

3.2 Timing Characterization

A Nangate 45nm Open Cell Library [18] has been scaled to 16nm technology node. After native TPL conflict detection and layout modification, we carry out the standard cell level timing analysis. Calibre xRC [19] is used to extract parasitic information of the cell layout. For each cell, we have original and modified layout with case 1 and case 2 options. From extraction results, we can see that the source/drain parasitic resistance of transistors varies with the position of contacts, which is the direct impact from layout modification. We use SPICE simulation to characterize different types of gates, which is based on 16nm PTM model [20]. Then, we can get the propagation delay of each gate, which is the average of rising and falling delay. We pick up six most commonly used cells to measure the relative changes of propagation delay due to layout modification (see Fig. 5). It is clearly observed that, for both case 1 and case 2, the timing impact will be within 0.5% of the original propagation delay of gates, which is assumed to be insignificant timing variation. Based on case 1 or case 2 option, we will remove all conflicts among cells of the library with negligible timing impact. Then we can ensure the standard cell compliance for triple patterning lithography.

4. STANDARD CELL PRE-COLORING

For each type of standard cell, after removing the native TPL conflicts, we provide one set of pre-coloring solutions, which can be prepared as a supplement to the library. In this section we introduce the pre-coloring problem, and propose general algorithms to solve it.

4.1 Problem Formulation

At first glance, standard cell pre-coloring is similar to cell level layout decomposition. However, different from the traditional layout decomposition, pre-coloring could have more than one solution for each cell. It is observed that for some complex cell structure, if we exhaustively enumerate the possible coloring, it would have thousands of solutions. Large solution size would impact the performance of our whole flow (see analysis in Section 5). To provide high quality pre-coloring solutions, meanwhile keep the solution size as small as possible, we define immune feature and redundant coloring solutions as follows.

Definition 1 (Immune feature). In one standard cell, an inside feature that would not conflict with any outside feature is defined as an immune feature.

It is easy to see that for one feature, if its distances to both vertical boundaries are larger than d_{min}, its color would not conflict with any other cells. Then this feature is an immune feature.

Definition 2 (Redundant coloring solutions). If two coloring solutions are only different at the immune features, these two solutions are redundant to each other.

Problem 1 (Standard Cell Pre-Coloring). *Given the input standard cell layout, and the maximum allowed stitch number $maxS$, we seek to search all coloring results that with stitch number no more than $maxS$. Meanwhile, all redundant coloring solutions should be removed.*

For example, given an AND2X1 cell as shown in Fig. 6(a), if $maxS$ is set as 1, the pre-coloring problem would search eight solutions (4 solutions with 0 stitch and 4 solutions with 1 stitch, see Fig. 7).

Given the input standard cell layout, all the stitch candidates are captured through wire projection [6] [8]. An example of AND2X1 cell is illustrated in Fig. 6 (a), where five

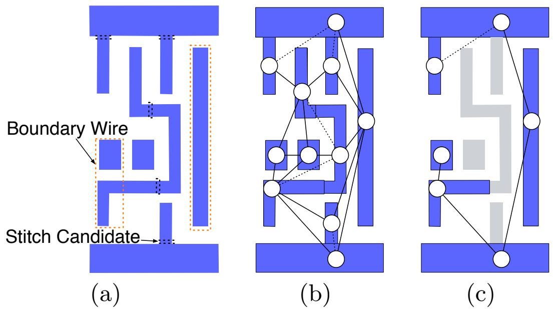

Figure 6: Constraint graph construction and simplification. (a) Input layout and all stitch candidates. (b) Constraint graph (CG) where solid edges are conflict edges and dash edges are stitch edges. (c) The simplified constraint graph (SCG) after removing immune features.

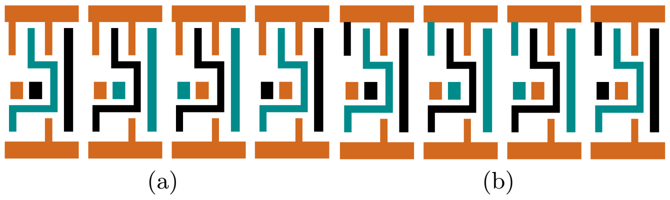

Figure 7: AND2X1 pre-coloring solutions with (a) 0 stitch; (b) 1 stitch.

stitch candidates are captured for M1 layer. Note that we forbid stitch on small features, e.g., contact, due to printability issue. In addition, different from previous stitch candidate generation, we forbid the stitch on boundary metal wires (see the red boxes in Fig. 6 (a)). The reason is based on the observation that boundary stitches tend to cause indecomposable patterns between two cells. Then an undirected constraint graph (*CG*) [6] is constructed to represent all input features and all the stitch candidates. One feature in the layout is divided into two vertices in the graph if one stitch candidate is introduced. The CG contains two sets of edges, i.e., the conflict edges and the stitch edges, respectively. Fig. 6 (b) shows the corresponding CG.

4.2 SCG Solution Enumeration

Since in CG some vertices represent the immune features, to avoid redundant coloring solutions, these features are temporarily removed. We denote the remained graph as *simplified constraint graph* (SCG). A backtracking algorithm [21] is proposed to the simplified CG to enumerate all possible coloring solutions. For example, given the SCG shown in Fig. 6(c), there are 24 solutions. It should be mentioned that since all power/ground rails are assigned default color, the colors of corresponding vertices are assigned before the backtracking process.

4.3 CG Solution Verification

Until now we have enumerated all coloring solutions for simplified constraint graph (SCG). However, under the maximum stitch number $maxS$ constraint, not all the SCG solutions can achieve legal layout decomposition in initial constraint graph (CG). Therefore, CG solution verification is proposed to each generated solution. Since SCG is a sub-set of CG, the verification can be viewed as layout decomposition with pre-colored features on SCG. If a coloring solution for whole CG can be found with stitch number less than $maxS$, it would be stored as one pre-coloring solution.

Algorithm 1 CG Solution Verification

```
Input: set of initial coloring solutions $S'$ for SCG;
 1: Generate corresponding coloring solutions $S$ for CG;
 2: for each coloring solution $s_i \in S$ do
 3:     $minCost \leftarrow \infty$;
 4:     BRANCH-AND-BOUND(0, $s_i$);
 5:     if $minCost < maxS$ then
 6:         Output $s_i$ as legal pre-coloring solution;
 7:     end if
 8: end for

 9: function BRANCH-AND-BOUND($t, s_i$)
10:     if t $\geq$ size[$s_i$] then
11:         if GET-COST( ) $< minCost$ then
12:             $minCost \leftarrow$ GET-COST();
13:         end if
14:     else if LOWER-BOUND( ) $> minCost$ then
15:         Return;
16:     else if $s_i[t] \neq -1$ then
17:         BRANCH-AND-BOUND($t+1, s_i$);
18:     else                                        ▷ $s_i[t] = -1$
19:         for each available color $c$ do;
20:             $s_i[t] \leftarrow c$;
21:             BRANCH-AND-BOUND($t+1, s_i$);
22:             $s_i[t] \leftarrow -1$;
23:         end for
24:     end if
25: end function
```

As shown in Algorithm 1, the CG solution verification is based on branch and bound [21]. Given the coloring solutions $S' = \{s'_1, s'_2 \ldots s'_n\}$ for SCG, at the beginning the corresponding coloring solutions $S = \{s_1, s_2, \ldots, s_n\}$ for CG are generated (line 1). Then we iteratively check each coloring solution s_i (lines 2−6). For one coloring solution s_i, if vertex t belongs to SCG, $s_i[t]$ should be already assigned one legal color. If t does not belong to SCG, $s_i[t] \leftarrow -1$. The BRANCH-AND-BOUND() algorithm traverses the decision tree with a depth first search (DFS) methods (lines $7-19$). For each vertex t, if $s_i[t]$ has been assigned one legal color in SCG, we skip t and travel to the next vertex. Otherwise, every legal color would be assigned to t before traveling to the next vertex. Different from exhaustive search, search space can be effectively reduced through pruning process (lines $11-12$). The function LOWER-BOUND() is to get lower bound by calculating current stitch number. Note that if one conflict is found, then the function returns a large value. Before checking any legal color of vertex t, we calculate its lower bound first. If LOWER-BOUND() is larger than $minCost$, we shall not branch from t, since all the children solutions will be of higher cost than $minCost$. Through the travel, all vertices have been assigned legal colors, stored in s_i. After the travel, if $minCost \leq maxS$, then s_i is one of the pre-coloring solutions (lines $5-6$).

It shall be noted that although other optimal layout decomposition techniques, like integer linear programming (ILP), may be modified as the verification engine, our branch and bound based method is easy to implement and effective for standard cell level problem size. Even for the most complex cell, SCG solution enumeration and CG solution verification can be finished in 5 seconds.

4.4 Look-Up Table Construction

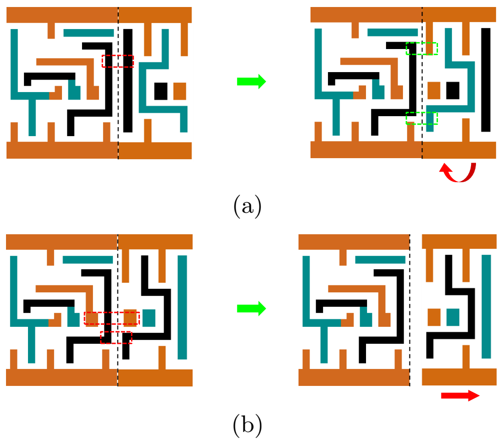

Figure 8: Two techniques for removing conflicts during placement. (a) Flip the cell; (b) Shift the cell.

For each cell c_i in the library, we have generated a set of pre-coloring solutions $S_i = \{s_{i1}, s_{i2}, \ldots, s_{iv}\}$. We further pre-compute the decomposability of each cell pair, and store them in a lookup table. For example, if two cells c_i, c_j are assigned with $p-$th and $q-$th coloring solutions, respectively, then LUT(i,p,j,q) would store the minimum distance required when c_i is to the left of c_j. That is, if two colored cells can be legally abutted to each other, the corresponding value would be 0. Otherwise, the value would be the distance required to keep two cells decomposable. Meanwhile, for each cell, its stitch number in different coloring solutions are also stored. It shall be noted that during the Lookup table construction, the cell flipping is considered, and related values are stored as well.

5. TPL AWARE SINGLE ROW PLACEMENT

In this section we solve a single row placement, where the orders of all cells on the row are determined. When TPL process is not considered, this row based design problem is called *Ordered Single Row* (OSR) problem, which has been well studied [22–24]. Here we revisit the OSR problem with the TPL process consideration.

5.1 Problem Formulation

To formalize the OSR problem under TPL process, we introduce the following notations. We consider an input single row as m ordered sites $R = \{r_1, r_2, \ldots, r_m\}$, and an input n movable cells $C = \{c_1, c_2, \ldots, c_n\}$ whose order is determined. That is, c_i is to the left of c_j, if $i < j$. Each cell c_i has v_i different coloring solutions. A *cell-color pair* (i,p) denotes that cell c_i is assigned to the $p-$th color solution, where $p \in [1, v_i]$. Meanwhile, $s(i,p)$ gives the corresponding stitch number for (i,p). The horizontal position of cell c_i is given by $x(c_i)$, and the cell width is given by $w(c_i)$. All the cells in other rows are with fixed positions. A single row placement is *legal* if and only if any two cells c_i, c_j meets the following non-overlap constraint:

$$x(c_i) + w(c_i) + \text{LUT}(i,p,j,q) \le x(c_j), \quad \text{if } (i,p)\&(j,q)$$

where LUT(i,p,j,q) is corresponding LUT value mentioned in Section 4.4. Based on all these notations, we define the TPL aware Ordered Single Row (*TPL-OSR*) problem as follows.

Problem 2 (TPL aware Ordered Single Row Problem). *Given a single row placement, we seek a legal placement and cell color assignment, so that the half-perimeter wire-length (HPWL) of all nets and the total stitch number are minimized.*

Compared with traditional OSR problem, TPL-OSR problem faces two special challenges: (1) TPL-OSR not only needs to solve cell placement, but also needs to assign appropriate coloring solutions for cells to minimize the stitch number. In other words, cell placement and color assignment should be solved simultaneously. (2) In conventional OSR problem, if the sum of all cell width is less than row capacity, it is guaranteed that there would be one legal placement solution. However, for TPL-OSR problem, since some extra sites may be spared to resolve coloring conflicts, before coloring assignment we cannot calculate the required site number.

In addition, it shall be noted that compared with conventional color assignment problem, in TPL-OSR the solution space is much larger. That is, to resolve the coloring conflict between two abutted cells c_i, c_j, apart from picking up compatible coloring solutions, TPL-OSR can seek to flip cells (see Fig. 8 (a)) or shift cells (see Fig. 8 (b)).

5.2 Graph Model for TPL-OSR

In this subsection we propose a graph model that correctly captures the cost of HPWL and the stitch number. Furthermore, we will show that performing a shortest path algorithm on the graph model can optimally solve the TPL-OSR problem.

To consider cell placement and cell color assignment simultaneously, a directed acyclic graph $G = (V, E)$ is constructed. The graph G is with vertex set V and edge set E. $V = \{\{0, \ldots, m\} \times \{0, \ldots, N\}, t\}$, where $N = \sum_{i=1}^{n} v_i$. The vertex in the first row and the first column is defined as vertex s. We can see that each column corresponds to one site's start point, and each row is related to one specified color assignment of one cell. Without loss of generality, we label each row as $r(i,p)$ if it is related to cell c_i with $p-$th coloring solution. The edge set E is composed of three sets of edges: horizontal edges E_h, ending edges E_e, and diagonal edges E_d.

$$\begin{aligned}
E_h =& \{(i, j-1) \rightarrow (i,j) | 0 \le i \le N, 1 \le j \le m\} \\
E_e =& \{(i,m) \rightarrow t | i \in [1, N]\} \\
E_d =& \{(r(i-1,p), k) \rightarrow (r(i,q), k + w(c_i) + \\
& LUT(i-1,p,i,q)) | i \in [2,n], p \in [1, v_{i-1}], q \in [1, v_i]\}
\end{aligned}$$

We denote each edge by its start and end point. A legal TPL-OSR solution corresponds to finding a directed path from the vertex s to vertex t. Sometimes one row cannot insert all the cells, therefore ending edges are introduced. With these ending edges, the graph model can guarantee to find out one path from s to t.

To simultaneously minimize the HPWL and stitch number, we define the cost on edges as follows. (1) All horizontal edges are with zero cost. (2) For ending edge $\{(r(i,p), m) \rightarrow t\}$, it is labelled by the cost $(n-i) \cdot M$, where M is a large number. (3) For diagonal edge $\{(r(i,p), k) \rightarrow (r(j,q), k + w(c_j) + LUT(i,p,j,q))\}$, it is labelled by the cost as follows:

$$\Delta WL + \alpha \cdot s(i,p) + \alpha \cdot s(j,q)$$

where ΔWL is the HPWL increment of placing c_j in position $q - \text{LUT}(i,p,j,q)$. Here α is a user-defined parameter for assigning relative importance between the HPWL and the

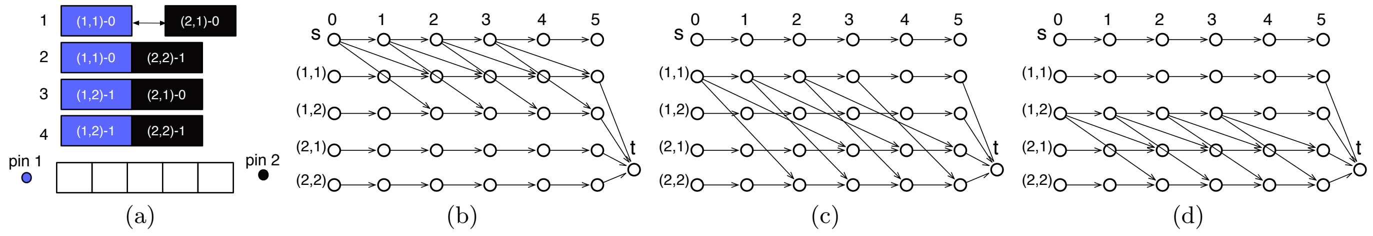

Figure 10: Example for the TPL-OSR problem. (a) two cells with different coloring solutions to be placed into a 5 sites row; Graph models with diagonal edges (b) from s vertex to first cell; (c) from c1_1 to second cell; (d) from c1_2 to second cell.

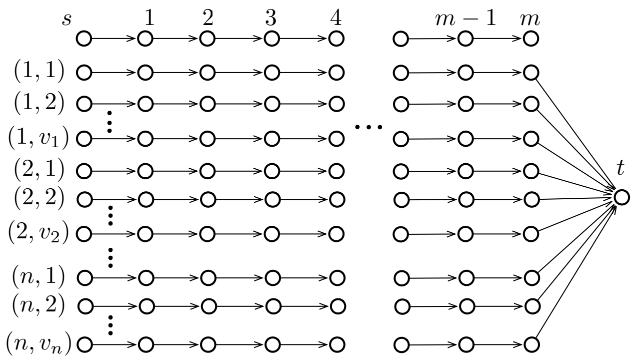

Figure 9: Graph model for the TPL-OSR problem (only the horizontal edges and ending edges are showed).

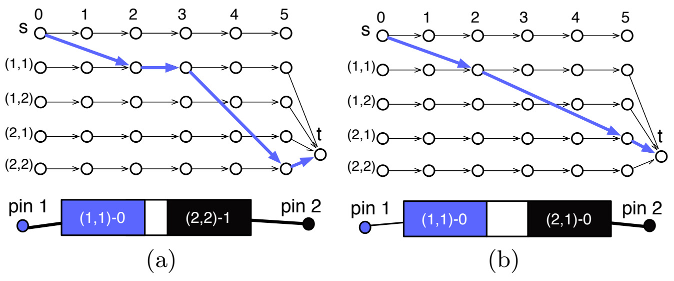

Figure 11: Shortest path solutions on the graph model with (a) 1 stitch; (b) 0 stitch.

stitch number. In our framework, α is set as 10. The general structure of G is shown in Fig. 9. Note that for clarity here we do not show the diagonal edges.

One example of the graph model is illustrated in Fig. 10, where two cells c_1 and c_2 are to placed in a row with 5 sites. Each cell has two different coloring solutions, and corresponding required stitch number. For example, the label (2,1)-0 means c_2 is assigned to the first coloring solution, with no stitch. The graph model is shown in Fig. 10(b)(c)(d), where each figure shows different part of diagonal edges. Cells c_1 and c_2 are connected with pin 1 and pin 2, respectively. Therefore, c_1 tends to be on the left side of row, while c_2 tends to be on the right side. Fig. 11 gives two shortest path solutions with the same HPWL. Because the second one is with less stitch number, it would be selected as the solution for TPL-OSR problem.

Since G is a directed acyclic graph, the shortest path can be calculated using topological traversal of G in $O(mnK)$ steps, where K is the maximal pre-coloring solution number for each cell. To apply topological traversal, a dynamic programming algorithm is proposed to find the shortest path from the s vertex to the t vertex.

5.3 Two Stage Speedup

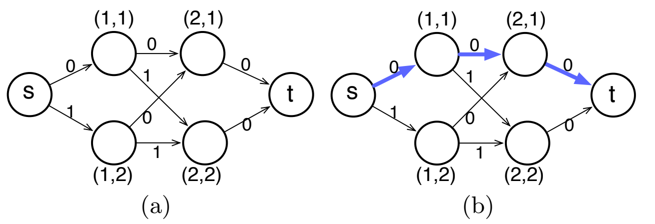

Figure 12: First stage model to solve color assignment. In this example edge cost only considers the stitch number minimization.

Although the shortest path algorithm can be solved in $O(mnK)$, for practical design when each cell could allow many pre-coloring solutions, the proposed graph model may still suffer from long runtime penalty. To achieve a better trade-off between runtime and performance, here we propose a two-stage speedup technique. The main idea is that the whole previous graph model is decomposed into two smaller graph models, one for color assignment, and another for cell placement.

To solve the example in Fig. 10, the first stage graph model is illustrated in Fig. 12 (a), where the cost of each edge corresponds to the stitch number required for each cell-color pair (i, p). Note that in our framework, relative positions among cells are also considered in the edge cost. A shortest path on the graph corresponds to a color assignment with minimum stitch number.

Our second stage is for cell placement, and the previous color assignment solutions are considered here. That is, if in previous color assignment cells c_{i-1} and c_i are assigned its p−th and q−th coloring solutions, then the width of cell c_i is changed from $w(c_i)$ to $w(c_i) + \text{LUT}(i-1, p, i, q)$. By this way, the extra site to resolve coloring conflicts are prepared for cell placement. Based on the updated cell widths, the graph model in [24] can be directly applied here.

The first graph model can be solved in $O(nK)$, while the second graph model can be resolved in $O(mn)$. Therefore, although the speedup technique can not achieve optimal solution of TPL-OSR problem, applying the two-stage graph model can reduce the complexity from $O(mnK)$ to $O(nK + mn)$.

Algorithm 2 TPL aware Detailed Placement

```
Input: cells to be placed;
 1: repeat
 2:     Sort all rows;
 3:     Label all rows as $FREE$;
 4:     for each row $row_i$ do
 5:         Solve TPL-OSR prolbem for $row_i$;
 6:         if exist unsolved cells then
 7:             Global Moving;
 8:             Update cell widths considering assigned colors;
 9:             Solve cell placement (OSR) for $row_i$;
10:         end if
11:         Label $row_i$ as $BUSY$;
12:     end for
13: until no significant improvement
```

6. OVERALL PLACEMENT SCHEME

In this section we present our overall scheme for the TPL aware detailed placement.

The flow of our detailed placement algorithm is summarized in Algorithm 2. In each main loop, rows are sorted that the row with more cells occupied would be solved earlier. At the beginning, all rows are labeled as $FREE$, which means it can be inserted additional cells (line 3). For each row row_i, we propose TPL-OSR algorithm as introduced in Section 5 to solve color assignment and cell placement simultaneously. Note that sometimes TPL-OSR cannot guarantee to assign all cells into row, due to extra sites required to resolve coloring conflicts.

If TPL-OSR ends with unsolved cells, *Global Moving* is applied to move some cells to other rows (line 7). The basic idea behind the Global Moving is to find the "optimal row and site" for a cell in the placement region, and remove some local triple patterning conflicts. For each cell we define its "optimal region" as the site to place where the HPWL is optimal [25]. Note that one cell can be only moved to $FREE$ rows. Since some cells in the middle of row may be moved, we need to solve OSR problem to rearrange the cell positions [24]. Note that since all cells on the row have been assigned colors, cell widths should be updated to preserve extra space for coloring conflict (line $8-9$) . After solving one row_i, it is labeled as $BUSY$ (line 10).

Since the rows are placed and colored one by one sequentially, the solution obtained within one single row may not be good enough, Therefore, our scheme is able to repeatedly call the main loop, until no significant improvement is achieved.

7. EXPERIMENTAL RESULTS

We implement our standard cell pre-coloring and TPL aware detailed placement in C++, and all the experiments are performed on a Linux machine with 3.0GHz CPU. We use Design Compiler [26] to synthesize OpenSPARC T1 designs based on Nangate 45nm standard cell library [18]. During benchmark generation, all the library and benchmark are scaled down to 16nm technology node. For each benchmark, we perform placement with Cadence SOC Encounter [27] to generate initial placement result. To better compare the performance of detailed placement under different placement densities, for each circuit, we choose three different core utilization rates 0.7, 0.8, and 0.9.

We compare four different design flows for M1 layer of all the benchmarks. "**Post-Decomposition**" means the conventional TPL design flow, where Encounter is chosen as the placer and an academic decomposer [7] is applied for layout decomposition. Note here the standard cell inner native conflicts have been removed through our compliance techniques (see Section 3). We implement the greedy based detailed placement algorithm in [15], denoted as "**GREEDY**". Although the work in [15] is for the self-aligned double patterning (SADP) friendly design, the proposed detailed placement algorithm can be integrated into our framework as well. "**TPLPlacer**" and "**TPLPlacer-SPD**" are the proposed detailed placement, where the "TPLPlacer" applies the optimal graph model to solve cell placement and color assignment simultaneously, while the "TPLPlacer-SPD" uses fast two-stages graph models to solve color assignment and cell placement iteratively.

All the experimental results are listed in Table 1, where columns "CN#" and "ST#" are the conflict number and the stitch number on the final decomposed layout, respectively. Column "WLD" shows the total wire length difference between initial placement and our TPL aware placement, and column "CPU(s)" gives the runtime. First, from the table we can see that under the conventional design flow, which is placement + layout decomposition, even each standard cell itself is TPL-friendly, averagely 1,700 conflicts are reported in final decomposed layout. Second, we can see that although for some cases GREEDY can achieve 0 conflict results, in 10 out of 21 cases it cannot find out legal placement results. For those illegal results labels "N/A" are reported. The main reason is that GREEDY only shifts the cells toward right direction. For some benchmark with high cell utilization, it may cause the final placement violation. In addition, GREEDY uses a greedy method to assign cell color, thus it loses the global view to minimize the stitch number. Therefore, more stitches are reported for those cases where it finds out legal results.

We further compare our two detailed placement strategies, TPLPlacer, and TPLPlacer-SPD. From Table 1 we can see that TPLPlacer can achiever slightly better HPWL (0.22%). However, TPLPlacer-SPD can achieve 14x speedup and less stitch number (5% less). The speedup is due to the faster two-stage graph model, instead of combined graph model. The main reason for the less stitch number is that the TPLPlacer-SPD solves color assignment first, followed by cell placement. Although cell position is integrated into the color assignment model, the shortest path with less number of stitches tends to be selected. The results in Table 1 demonstrate the effectiveness of our standard cell compliance and detailed placement techniques.

8. CONCLUSION

In this paper we propose a coherent framework to seamlessly integrate the TPL aware optimizations into early design stages. To our best knowledge, this is the first work for TPL compliance at both standard cell and placement levels. An optimal graph model to simultaneously solve cell placement and color assignment is proposed, and then a two-stage graph model is presented to achieve speedup. Our framework is compared with traditional layout decomposition. The results show that considering TPL constraints in early design stages can dramatically reduce the conflict number and stitch number in final layout. As continuing growth of technology node to sub-16 nm, TPL turns out to be a definitely promising lithography solution. A dedicated design flow that integrating TPL constraints is necessary to assist in the whole process. We believe this paper will stimulate more research on TPL and TPL aware design.

Table 1: Comparisons of Detailed Placement Algorithms

bench	Post-Decomposition			GREEDY [15]				TPLPlacer				TPLPlacer-SPD			
	CN#	ST#	CPU(s)	CN#	ST#	WLD	CPU(s)	CN#	ST#	WLD	CPU(s)	CN#	ST#	WLD	CPU(s)
alu-70	605	4092	0.7	0	1254	+0.06%	2.0	0	1013	-0.94%	107.2	0	994	-0.77%	4.2
alu-80	656	4100	0.6	N/A	N/A	N/A	N/A	0	1011	-1.70%	114.8	0	994	-1.48%	4.6
alu-90	596	3585	0.5	N/A	N/A	N/A	N/A	0	1006	-2.38%	120.2	0	994	-2.2%	4.8
byp-70	1683	9943	2.4	0	3254	0.14	0.97	0	2743	-5.98%	382.5	0	2545	-5.69%	9.2
byp-80	1918	10316	2.6	N/A	N/A	N/A	N/A	0	2889	-2.58%	343.0	0	2545	-2.12%	7.9
byp-90	2285	10790	3.0	N/A	N/A	N/A	N/A	0	3136	+1.74%	361.9	0	2514	+4.31%	7.1
div-70	1329	6017	2.2	0	2368	+0.08%	1.89	0	2119	-3.84%	117.6	0	2017	-3.28%	5.6
div-80	1365	5965	2.1	0	2379	+0.08%	1.87	0	2090	-2.06%	135.6	0	2017	-1.63%	6.1
div-90	1345	5536	2.1	0	2365	+0.02%	1.87	0	2080	-4.79%	155.2	0	2017	-4.37%	6.4
ecc-70	206	3852	0.9	N/A	N/A	N/A	N/A	0	247	-4.76%	69.4	0	228	-4.6%	1.7
ecc-80	265	3366	1.0	0	433	+0.43%	0.44	0	274	-2.51%	58.2	0	228	-2.19%	1.5
ecc-90	370	4015	1.1	N/A	N/A	N/A	N/A	0	369	-1.28%	68.5	0	228	-1.53%	1.4
efc-70	503	3333	0.7	0	1131	+0.0 %	5.46	0	1005	-1.32%	32.4	0	1005	-1.31%	6.2
efc-80	570	4361	0.6	N/A	N/A	N/A	N/A	0	1008	-3.35%	37.7	0	1005	-3.26%	6.3
efc-90	534	4040	0.8	0	1133	+0.0 %	5.4	0	1005	+0.35%	39.0	0	1005	+0.35%	6.3
ctl-70	425	2583	1.3	0	703	+0.23%	3.8	0	573	-1.75%	67.3	0	553	-1.56%	5.3
ctl-80	529	3332	1.4	0	714	+0.5 %	3.8	0	561	-2.26%	78.8	0	553	-2.04%	5.5
ctl-90	519	3241	1.5	0	726	+0.4 %	3.8	0	556	-0.63%	85.4	0	553	-0.5%	5.6
top-70	5893	27981	9	N/A	N/A	N/A	N/A	0	8069	-10.6%	1948.0	0	8034	-10.4%	43.5
top-80	6775	32352	10.3	N/A	N/A	N/A	N/A	0	8120	-5.45%	1696.7	0	8015	-4.9%	36.8
top-90	7313	29343	11.4	N/A	N/A	N/A	N/A	0	8710	-4.41%	1850.9	0	7876	+2.09%	32.7
Average	**1700**	8664	2.68	N/A	N/A	N/A	N/A	**0**	2314	-2.88%	142.9	**0**	2186	-2.24%	9.94
Ratio									**1.0**		**1.0**		**0.95**		**0.07**

9. ACKNOWLEDGMENT

This work is supported in part by NSF grants CCF-0644316 and CCF-1218906, SRC task 2414.001, NSFC grant 61128010, and IBM Scholarship.

10. REFERENCES

[1] ITRS. [Online]. Available: http://www.itrs.net

[2] B. Yu, J.-R. Gao, D. Ding, Y. Ban, J.-S. Yang, K. Yuan, M. Cho, and D. Z. Pan, "Dealing with IC manufacturability in extreme scaling," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 240–242.

[3] K. Lucas, C. Cork, B. Yu, G. Luk-Pat, B. Painter, and D. Z. Pan, "Implications of triple patterning for 14 nm node design and patterning," in *Proc. of SPIE*, vol. 8327, 2012.

[4] C. Cork, J.-C. Madre, and L. Barnes, "Comparison of triple-patterning decomposition algorithms using aperiodic tiling patterns," in *Proc. of SPIE*, vol. 7028, 2008.

[5] R. S. Ghaida, K. B. Agarwal, L. W. Liebmann, S. R. Nassif, and P. Gupta, "A novel methodology for triple/multiple-patterning layout decomposition," in *Proc. of SPIE*, vol. 8327, 2011.

[6] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 1–8.

[7] S.-Y. Fang, W.-Y. Chen, and Y.-W. Chang, "A novel layout decomposition algorithm for triple patterning lithography," in *IEEE/ACM Design Automation Conference (DAC)*, 2012.

[8] H. Tian, H. Zhang, Q. Ma, Z. Xiao, and M. Wong, "A polynomial time triple patterning algorithm for cell based row-structure layout," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012.

[9] B. Yu, J.-R. Gao, and D. Z. Pan, "Triple patterning lithography (TPL) layout decomposition using end-cutting," in *Proc. of SPIE*, vol. 8684, 2013.

[10] J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *IEEE/ACM Design Automation Conference (DAC)*, 2013.

[11] 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 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013.

[12] L. Liebmann, D. Pietromonaco, and M. Graf, "Decomposition-aware standard cell design flows to enable double-patterning technology," in *Proc. of SPIE*, vol. 7974, 2011.

[13] S. Hu and J. Hu, "Pattern sensitive placement for manufacturability," in *ACM International Symposium on Physical Design (ISPD)*, 2007, pp. 27–34.

[14] M. Gupta, K. Jeong, and A. B. Kahng, "Timing yield-aware color reassignment and detailed placement perturbation for double patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2009, pp. 607–614.

[15] J.-R. Gao, B. Yu, R. Huang, and D. Z. Pan, "Self-aligned double patterning friendly configuration for standard cell library considering placement," in *SPIE Intl. Symp. Advanced Lithography*, 2013.

[16] Q. Ma, H. Zhang, and M. D. F. Wong, "Triple patterning aware routing and its comparison with double patterning aware routing in 14nm technology," in *IEEE/ACM Design Automation Conference (DAC)*, 2012, pp. 591–596.

[17] Y.-H. Lin, B. Yu, D. Z. Pan, and Y.-L. Li, "TRIAD: A triple patterning lithography aware detailed router," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012.

[18] "NanGate FreePDK45 Generic Open Cell Library," http://www.si2.org/openeda.si2.org/projects/nangatelib.

[19] "Mentor Calibre," http://www.mentor.com.

[20] "Predictive Technology Model ver. 2.1," http://ptm.asu.edu.

[21] T. C. Hu and M.-T. Shing, *Combinatorial Algorithms: Enlarged Second Edition.* Courier Dover Publications, 2002.

[22] A. B. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of linear placements for wirelength minimization with free sites," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 1999, pp. 241–244.

[23] U. Brenner and J. Vygen, "Faster optimal single-row placement with fixed ordering," in *Proc. Design, Automation and Test in Eurpoe*, 2000, pp. 117–121.

[24] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2005, pp. 891–898.

[25] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. on Circuits and Systems*, vol. 28, no. 1, pp. 12–18, 1981.

[26] "Synopsys Design Compiler," http://www.synopsys.com.

[27] "Cadence SOC Encounter," http://www.cadence.com/.