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

Bei Yu, *Member, IEEE*, Xiaoqing Xu, *Student Member, IEEE*, Jhih-Rong Gao, Yibo Lin, Zhuo Li, *Senior Member, IEEE*, Charles J. Alpert, *Fellow, IEEE*, and David Z. Pan, *Fellow, IEEE*

***Abstract*—As the feature size of semiconductor process further scales to sub-16 nm technology node, triple patterning lithography (TPL) has been regarded as one of the most promising lithography candidates along with extreme ultraviolet, electron beam lithography, and directly self-assembly. M1 and contact layers, which are usually deployed within standard cells, are the 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 precoloring solutions of standard cells, we present a TPL aware detailed placement where the layout decomposition and placement can be resolved simultaneously. In addition, we propose a linear dynamic programming to solve TPL aware detailed placement with maximum displacement, which can achieve good trade-off in terms of runtime and performance. Experimental results show that our framework can achieve zero conflict, meanwhile can effectively optimize the stitch number and placement wire-length.**

***Index Terms*—Design compliance, detailed placement, dynamic programming, standard cell design, triple patterning lithography (TPL).**

I. Introduction

As THE feature size of semiconductor process technology nodes further scales to sub-16 nm, triple patterning lithography (TPL) has been regarded as one of the most promising lithography candidates, along with extreme ultraviolet lithography, directed self-assembly, and electron beam lithography [1]–[3]. TPL is a viable solution for emerging logic nodes, as other candidates suffer from either throughput problem or yield issues [4], [5]. As a natural extension along the paradigm of double patterning lithography (DPL), TPL has been pushed to its limit in sub-16 nm to introduce better printability [6].

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 on one mask two features have a spacing less than $d_{\min}$. Sometimes the conflict can be also resolved by inserting a stitch to split a feature into two touching parts. The TPL layout decomposition problem with conflict and stitch minimization has been well studied in the past few years [7]–[15]. However, most existing work suffers from one or more of the following drawbacks.

1) Because TPL layout decomposition problem is NP-hard [9], most of the decomposers are based on approximation or heuristic methods. Thus some extra conflicts may be reported.
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 ability to resolve some native TPL conflict patterns, e.g., four-clique conflicts.

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 four-clique patterns, which are 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 engineering change order efforts, generating TPL-friendly layouts, especially in the early design

Manuscript received July 22, 2014; revised October 1, 2014 and December 28, 2014; accepted January 16, 2015. Date of publication February 9, 2015; date of current version April 17, 2015. This work was supported in part by the National Science Foundation under Grant CCF-0644316 and Grant CCF-1218906, in part by Semiconductor Research Corporation (SRC) Task 2414.001, in part by the National Science Foundation of China under Grant 61128010, and in part by IBM Scholarship. This paper was recommended by Associate Editor A. Davoodi.

B. Yu, X. Xu, Y. Lin, and D. Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: bei@cerc.utexas.edu).

J-R. Gao, Z. Li, and C. J. Alpert are with Cadence Design Systems, Austin, TX 78752 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2401571

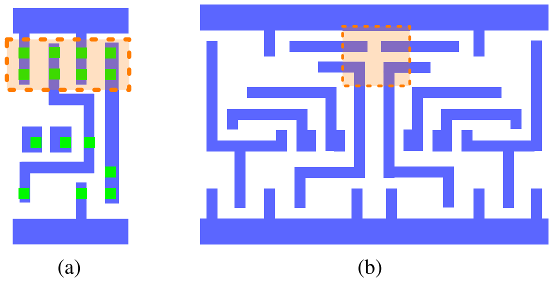


Fig. 1. Two native conflicts (in read boxes) from (a) contact layer within a standard cell and (b) M1 layer between adjacent standard cells.

stage, becomes urgent and pivotal. Through these two examples, we can see that TPL constraints should be considered in both standard cell design and placement stages, so that we can avoid indecomposable patterns in final layout.

There exist several placement studies toward different manufacturing process targets [16]–[22]. Liebmann *et al.* [16] proposed some guidelines to enable DPL friendly standard cell design and placement. Taghavi *et al.* [22] presented a set of algorithms to disperse local congestion. [23] and [24], 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, i.e., the TPL aware detailed placement can generate optimized positions and color assignment solutions for all cells simultaneously. Therefore, our framework does not require conventional and time consuming chip level layout decomposition. Our main contributions are summarized as follows.

1) We propose systematic standard cell compliance techniques for TPL and coloring solution generation.
2) We study the standard cell precoloring problem, and propose effective methods.
3) We present the first systematic study for the TPL aware ordered single row (TPL-OSR) placement, where cell placement and color assignment can be solved simultaneously.
4) We propose linear dynamic programming algorithm to solve TPL aware single row placement with maximum displacement, and achieve a good trade-off in terms of runtime and solution quality.
5) Our framework seamlessly integrates decomposition in each key step, therefore, no additional layout decomposition is required.
6) Experimental results show that our framework can achieve zero conflict, meanwhile can effectively reduce the stitch number.

The rest of this paper is organized as follows. Section II provides preliminaries and overview of our methodologies. Section III proposes standard cell modification to enable TPL friendly cell layout, with negligible timing impact. Section IV proposes cell precoloring techniques and look-up table (LUT)

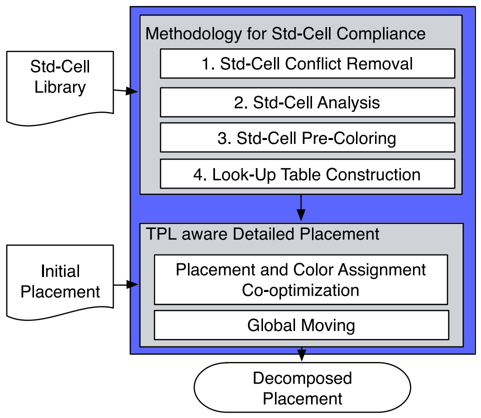

Methodology for Std-Cell Compliance
1. Std-Cell Conflict Removal
2. Std-Cell Analysis
3. Std-Cell Pre-Coloring
4. Look-Up Table Construction
Std-Cell Library
TPL aware Detailed Placement
Placement and Color Assignment Co-optimization
Global Moving
Initial Placement
Decomposed Placement

Fig. 2. Overall flow of the methodologies for standard cell compliance and detailed placement.

construction. Sections V–VII give details on our TPL aware detailed placement. Section VIII presents the experiment results, followed by the conclusion in Section IX.

II. Preliminaries

A. Row Structure Layout

Our framework assumes a row-structure layout where cells in each row have the same height, and power/ground rails are going from the very left to the very right. A similar assumption was applied in row-based TPL layout decomposition [11] as well. 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 separately. Without loss of generality, for each row the power/ground rails are assigned to the color 1 (default color). In other words, the coloring assignment results in each row are able to be merged together, without losing optimality.

B. Overall Design Flow

The overall flow of our proposed framework is illustrated in Fig. 2, which consists of two stages: 1) methodologies for standard cell compliance and 2) TPL aware detailed placement. In the first stage, standard cell compliance, we carry out standard cell conflict removal, timing analysis, standard cell precoloring, and LUT generation. After the first stage we can ensure that, for each cell, TPL friendly cell layout and a set of precoloring solutions will be provided. In the second stage, TPL aware detailed placement, we will discuss how to consider TPL constraints in the single row placement problem (see Sections VI, VII) and global moving (see Section V).

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

III. 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

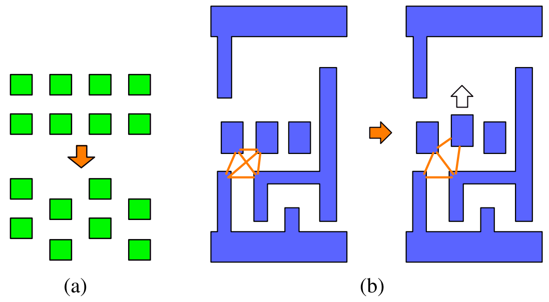


Fig. 3. Native conflict removal for (a) contact layer and (b) M1 layer.

TPL conflict cannot be resolved through either cell shift or layout decomposition. In addition, one cell may be applied many times in one single design, thus each inner native conflict may cause hundreds of coloring conflicts in the 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 four-clique conflicts through standard cell modification. Then, parasitic extraction and SPICE simulation are applied to analyze the timing impact for the cell modification.

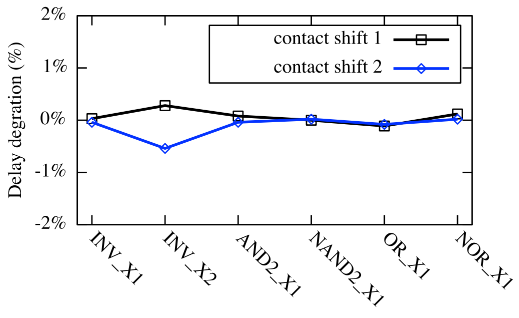
A. Native TPL Conflict Removal

For contact layer, one example of native TPL conflict removal is illustrated in Fig. 3(a), where four contacts introduce an indecomposable four-clique conflict structure. For such cases we modify the contact layout into hexagonal close packing, which allows for the most aggressive cell area shrinkage for TPL friendly layout [6]. With slight modification to the 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 cases 1 and 2, respectively, both of which will lead to TPL friendly standard cell layout. For M1 layer, we observe that most of the cells have a set of legal coloring solutions purely through stitch insertion. However, stitch insertion cannot resolve four-clique conflict from four points in a layout [12]. Thus for these cells we have to shift M1 patterns to resolve the conflict. One example of M1 layer native TPL conflict removal is illustrated in Fig. 3(b). Note that after layout modification, the layout still needs to satisfy the design rules. It shall be noted that although conventional cell migration techniques [25]–[27] might be able to automatically shift layout patterns to avoid four-clique patterns, it is hard to guarantee that the modified layout can maintain good timing performance. Therefore, in this paper, we manually modify the standard cell layout and verify timing after each shift operation.

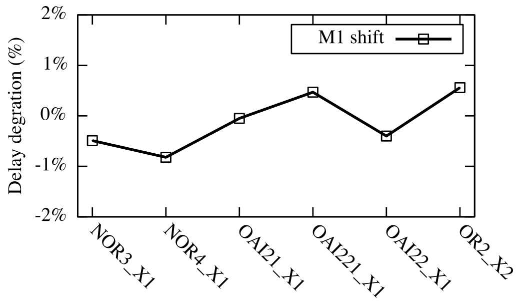
B. Timing Characterization

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.

A Nangate 45 nm Open Cell Library [28] has been scaled to 16 nm technology node. After native TPL conflict detection and



(a)



(b)

Fig. 4. Timing impact from layout modification for different types of gates. (a) Timing impact for contact layer modification. (b) Timing impact for M1 layer modification.

layout modification, we carry out the standard cell level timing analysis. Calibre xRC [29] is used to extract parasitic information of the cell layout. We use SPICE simulation to characterize different types of gates, which is based on the 16 nm Predictive Technology Model model [30]. Then, we can get the propagation delay of each gate, which is the average of rising and falling delay. For contact layer, we have original and modified layouts with cases 1 and 2 options. From the 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 pick up six most commonly used cells to measure the relative changes of propagation delay due to layout modification [see Fig. 4(a)]. It is clearly observed that, for both cases 1 and 2, the timing impact will be within 0.5% of the original propagation delay of gates, which is assumed to be insignificant timing variation. For M1 layer, if we forbid layout modification most cells still have some legal coloring solutions purely through stitch insertion. Only few cells need to shift patterns to ensure TPL friendly. For those cells with layout modification, Fig. 4(b) analyzes their timing impacts. We can see that the modification in M1 layer introduces less than 1% timing degradation, which is assumed to be insignificant. Based on the modifications on contact layer and M1 layer, we can remove all conflicts among cells with negligible timing impact. Thus, we can ensure the standard cell compliance for TPL.

IV. Standard Cell Precoloring

For each type of standard cell, after removing the native TPL conflicts, we seek a set of precoloring solutions. These cell solutions are prepared as a supplement to the library. In this section, we first describe the cell precoloring problem formulation; then, we introduce our algorithms to solve this problem.

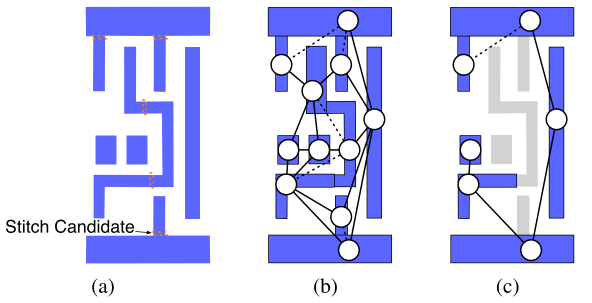


Fig. 5. CG construction and simplification. (a) Input layout and all stitch candidates. (b) CG where solid edges are conflict edges and dash edges are stitch edges. (c) SCG after removing immune features.

A. Problem Formulation

Given the input standard cell layout, all the stitch candidates are captured through wire projection [12]. One feature in the layout is divided into two touching parts, if one stitch candidate is introduced. Then a constraint graph (CG) is constructed to represent all input features and all the stitch candidates. A CG is an undirected graph, where each vertex is associated with one input layout feature. In a CG, there is a stitch edge iff the two corresponding touching vertices are connected through one stitch candidate, while there is a conflict edge iff two untouched vertices are within minimum coloring distance $d_{\min}$. For example, given an input layout shown in Fig. 5(a), five stitch candidates are generated through wire projection. The CG is illustrated in Fig. 5(b), where the conflict edges and the stitch edges are shown as solid edges and dash edges, respectively. Note that, we forbid stitch on small features, e.g., contact, due to printability issue. Different from previous stitch candidate generation, we forbid the stitch on boundary metal wires due to the observation that boundary stitches tend to cause indecomposable patterns between two cells.

Based on the CG, the standard cell precoloring problem is to search all possible coloring solutions. At first glance, this problem is similar to cell level layout decomposition. However, different from the conventional layout decomposition, for each cell precoloring could have more than one solution. It is observed that for some complex cell structures, if we exhaustively enumerate all possible colorings, it would have thousands of solutions. Large solution size would impact the performance of our whole flow. Therefore, to provide high quality precoloring solutions, meanwhile keeping 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 does 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 Precoloring): Given the input standard cell layout and the maximum allowed stitch number, max$S$, the CG is constructed. Standard cell precoloring problem searches all coloring solutions on CG such that the stitch number is no more than max$S$. Meanwhile, no two solutions are redundant with each other.

Since in CG some vertices represent the immune features, to avoid redundant coloring solutions, these features are temporarily removed. We denote the remaining graph as a simplified CG (SCG). For example, for the CG in Fig. 5(b), the corresponding SCG is shown in Fig. 5(c). Our standard cell precoloring algorithm consists of two stages: coloring solution enumeration on SCG, and solution verification on CG.

Algorithm 1 SCG Solution Enumeration

```
Require: SCG $G = \{V, CE, SE\}$;
1:  BACKTRACK(0, $G$);
2:  return All color solutions in $G$;

3:  function BACKTRACK($t$, $G$)
4:      if t $\geq$ size[$G$] then
5:          Store current color solution;
6:      else
7:          for all legal color $c$ do;
8:              $G[t] \leftarrow c$;
9:              BACKTRACK($t+1$, $G$);
10:             $G[t] \leftarrow -1$;
11:         end for
12:     end if
13: end function
```

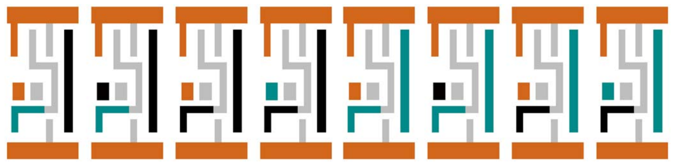


Fig. 6. AND2X1 cell example: eight enumerated solutions for SCG.

B. SCG Solution Enumeration

In the first step, given an SCG, we enumerate all possible coloring solutions. Our enumeration is based on backtracking algorithm [31], which usually explores implicit directed graphs to carry out a systematic search of all solutions.

The details of SCG solution enumeration are shown in Algorithm 1. Given an SCG, G, a backtracking function, BACKTRACK(0, G) is called to search the whole graph (line 1). The backtracking is a modified depth-first search of the solution space (lines 3–13). In line 7, a color c is denoted as legal, when vertex $G[t]$ is assigned color c, no conflict is introduced, and the total stitch number does not exceed max$S$. 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. For example, given the SCG shown in Fig. 5(c), if no stitch is allowed, there are eight solutions (see Fig. 6).

C. CG Solution Verification

Until now, we have enumerated all coloring solutions for SCG. However, not all the SCG solutions can achieve legal

Algorithm 2 CG Solution Verification

```
Require: Set of initial coloring solutions $S'$ for SCG;
 1: $s_i[t] = -1$ for any nodes not in SCG;
 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 precoloring 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                                        $\triangleright$ $s_i[t] = -1$
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
```

layout decomposition in the initial CG. Therefore, in the second step, 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 precolored features on SCG. If a coloring solution for whole CG can be found with stitch number less than max$S$, it would be stored as one precoloring solution. The CG solution verification is based on the branch-and-bound algorithm [31], which is very similar to backtracking in that a state space tree is used to solve a problem. However, the differences are twofold.

1) The branch-and-bound method is used only for optimization problem, i.e., only one solution is generated.
2) The branch-and-bound algorithm introduces bounding function to prune suboptimal nodes in search space.

That is, at each node of search space, we calculate a bound on the possible solution. If the bound is worse than the best solution we have found so far, then, we do not need to go to the sub-space.

The details of the CG solution verification are shown in Algorithm 2. Given an SCG coloring solutions $S' = \{s'_1, s'_2 \ldots s'_n\}$, at the beginning the corresponding CG coloring solutions $S = \{s_1, s_2, \ldots, s_n\}$ 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 method (lines 9–25). 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 the pruning process (lines 14 and 15). The function LOWER-BOUND() is to get the lower bound by calculating the 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 $\leq$ max$S$, then s_i is one of the precoloring solutions (lines 5 and 6).

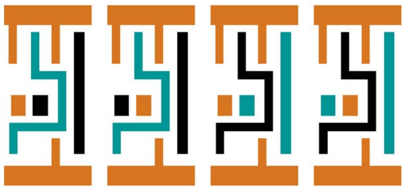


Fig. 7. AND2X1 cell example: in CG four verified solutions are stored as final coloring solutions.

It shall be noted that although other layout decomposition techniques, like integer linear programming, 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 five seconds. For the SCG solutions in Fig. 6, four solutions are verified and assigned final colors (see Fig. 7). These four solutions would be the final coloring solutions for this standard cell, and are provided as supplement to the library.

D. LUT Construction

For each cell c_i in the library, we have generated a set of precoloring solutions $S_i = \{s_{i1}, s_{i2}, \ldots, s_{iv}\}$. We further precompute the decomposability of each cell pair and store them in a LUT. For example, if two cells, c_i and c_j, are assigned with the pth and qth coloring solutions, respectively, then in LUT a value LUT(i, p, j, q) would be stored, which is the minimum distance required when c_i is to the left of c_j. If two colored cells can be legally abutted to each other, the corresponding value would be 0. Otherwise, the value would be the site number required to keep two cells decomposable. Meanwhile, for each cell, the stitch numbers in different coloring solutions are also stored. It shall be noted that during the LUT construction, the cell flipping is considered and related values are stored as well. For one cell, if there are k solutions during cell precoloring, $2k$ solutions would be stored in LUT to consider the cell flipping.

V. Overall Placement Scheme

In this section, we present our overall scheme for the whole design level TPL aware detailed placement. As mentioned in [32], unless the target objectives of detailed placement and global placement are the same, detailed placement could ruin

Algorithm 3 TPL Aware Detailed Placement

```
Require: Cells to be placed;
 1: Move cells for better wire-length;
 2: repeat
 3:     Sort all rows;
 4:     Label all rows as FREE;
 5:     for each row $row_i$ do
 6:         Solve single row problem for $row_i$;
 7:         if exist unsolved cells then
 8:             Global Moving;
 9:             Update cell widths considering assigned colors;
10:             Solve OSR problem for $row_i$;
11:         end if
12:         Label $row_i$ as BUSY;
13:     end for
14: until no significant improvement;
```

the optimized objective of global placement, such as timing and global routability. Therefore, inspired by [32], our TPL aware detailed placement formulates maximum displacement constraint D_{disp} as follows:

$$D_{\text{disp}} \geq \max_{c \in C} (|x_c - x_{c0}| + |y_c - y_{c0}|)$$

where C is a set of cells, (x_c, y_c) is the current position of cell c, (x_{c0}, y_{c0}) is the original position of cell c in initial placement.

Algorithm 3 summarizes the overall flow of the detailed placement scheme. At the beginning, we try to move each cell to another place to reduce the half perimeter wire-length, while the new place should be within the range of maximum displacement constraint. Then all rows are labeled as *FREE*, which means additional cells can be inserted (line 4). In each main loop, rows are sorted such that the row with more cells occupied would be solved earlier. For each row row$_i$, we carry out single row TPL aware detailed placement as introduced in Sections VI and VII, to solve color assignment and cell placement simultaneously. Note that sometimes in one row we cannot assign all cells legal positions, due to extra sites required to resolve coloring conflicts. If single row problem ends with unsolved cells, global moving is applied to move some cells to other rows (line 8). 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 half-perimeter wire-length (HPWL) is optimal [33]. During global moving the maximum displacement constraint should be satisfied as well. Since some cells in the middle of a row may be moved, we need to solve OSR problem to rearrange the cell positions [34]. Note that since all cells on the row have been assigned colors, cell widths would be updated to preserve extra space for coloring conflict (lines 9 and 10). After solving one row$_i$, it is labeled as *BUSY* (line 12). 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 (line 14).

TABLE I
NOTATIONS USED IN TPL-OSR PROBLEM

m	site number
n	cell number
C	a set of cells $\{c_1, c_2, \ldots, c_n\}$
v_i	pre-coloring solution number for cell c_i
K	$\max\{v_1, v_2, \ldots, v_n\}$
(i, p)	cell c_i is assigned to p-th color solution
$\text{LUT}(i, p, j, q)$	min distance required between (i, p) & (j, q)
$s(i, p)$	stitch number for (i, p)
$x(i)$	horizontal position of c_i
$w(i)$	width of c_i
$a(i)$	assigned color for c_i
$range(i)$	displacement range of c_i

VI. TPL-OSR PLACEMENT

In this section, we solve a single row placement, where the orders of all cells on the row are determined. When the TPL related constraints are not considered, this row-based design problem is the well studied OSR problem [34]–[37]. Here, we revisit the OSR problem with the TPL process consideration. For convenience, Table I lists the notations used in this section.

A. Problem Formulation

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. Besides, for each cell c_i, we define its displacement range as range(i), which is the bounds on the x-coordinate of c_i that is within the maximum displacement constraint. A cell-color pair (i, p) denotes that cell c_i is assigned to the pth 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 $x(i)$, and the cell width is given by $w(i)$. 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 and c_j, meet the following nonoverlap constraint:

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

where $\text{LUT}(i, p, j, q)$ is the minimum distance required between (i, p) & (j, q). Based on all these notations, we define the TPL-OSR problem as follows.

Problem 2 (TPL-OSR Problem): Given a single row placement, we seek a legal placement and cell color assignment, so that the HPWL of all nets and the total stitch number are minimized.

Compared with the traditional OSR problem, the 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 problems, if the sum of all cell widths is less than row capacity, it is guaranteed that there would be one legal placement solution.

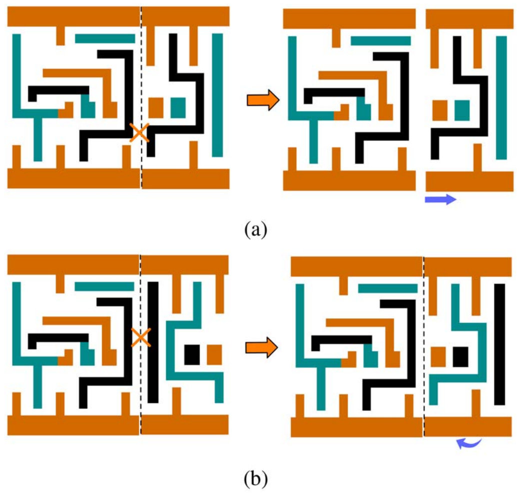


Fig. 8. Two techniques for removing conflicts during placement. (a) Shift the cell. (b) Flip the cell.

However, for TPL-OSR problems, 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 the 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 and 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)].

B. Unified Graph Model

In this section, 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. $V = \{\{0, \dots, m\} \times \{0, \dots, N\}, t\}$, 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 pth coloring solution. The edge set E is composed of three sets of edges: 1) horizontal edges E_h; 2) ending edges E_e; and 3) diagonal edges E_d

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

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 $\{(r(i, p), m) \to t\}$, it is labeled by the cost $(n - i) \cdot M$, where M is a large number.
3) For diagonal edge $\{(r(i-1, p), k) \to (r(i, q), k + w(i) + \text{LUT}(i-1, p, i, q))\}$, it is labeled by the cost as follows:

$$\alpha \cdot \Delta\text{WL} + s(i-1, p) + s(i, q)$$

where ΔWL is the HPWL increment of placing c_i in position $k + \text{LUT}(i-1, p, i, q)$.

Note that if position $k + \text{LUT}(i-1, p, i, q)$ is outside the range(i), we can simply ignore the diagonal edge. Here α is a user-defined parameter for assigning relative importance between the HPWL and the stitch number. In our framework, α is set to 10.

One example of the graph model is illustrated in Fig. 9, where two cells, c_1 and c_2, are to be placed in a row with five 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. 9(b)–(d), where each figure shows different part of diagonal edges. Cells c_1 and c_2 are connected with pins 1 and 2, respectively. Therefore, c_1 tends to be on the left side of the row, while c_2 tends to be on the right side. Fig. 10 gives two shortest path solutions with the same HPWL. Because the second has a smaller stitch number, it would be selected as the solution for the 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 precoloring 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.

C. Two Stage Graph Model

Although the unified graph model can be optimally solved through a shortest path method in $O(mnK)$, for practical design when each cell could allow many precoloring solutions, the proposed graph model may still suffer from long runtime penalty. Here, we present a new two-stage graph model for the TPL-OSR problem. The main idea is that the previous unified graph model is decomposed into two smaller graphs, one for color assignment and another for cell placement. Therefore, solving the new model can provide a fast solution to the TPL-OSR problem.

To solve the example in Fig. 9, the first stage graph model is illustrated in Fig. 11(a), where the cost of each edge $((i-1, p) \to (i, q))$ is defined as follows:

$$s(i, q) + \alpha \cdot \text{LUT}(i-1, p, i, q).$$

With this cost function, both stitch number and site number can be minimized simultaneously. A shortest path on the graph corresponds to a color assignment with optimized stitch number and site number.

Our second stage is for cell placement and the previous color assignment solutions are considered here. That is, if in previous

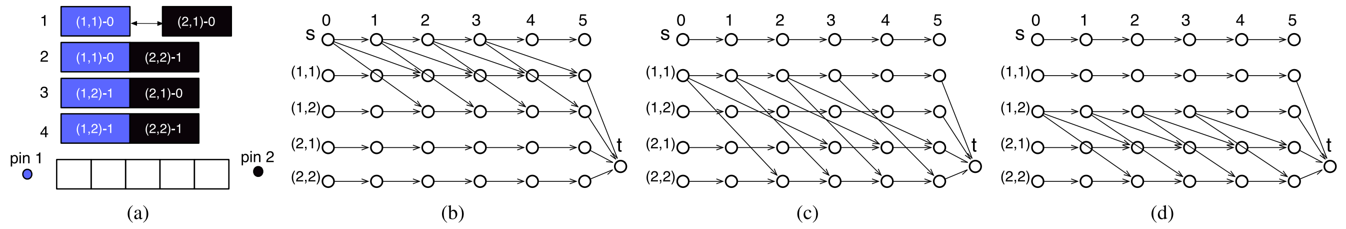


Fig. 9. Example for the TPL-OSR problem. (a) Two cells with different coloring solutions to be placed into a five sites row. Graph models with diagonal edges (b) from s vertex to first cell, (c) from c1_1 to second cell, and (d) from c1_2 to second cell.

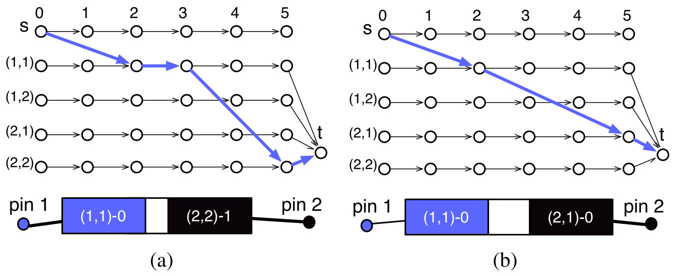


Fig. 10. Shortest path solutions on the graph model with (a) 1 stitch and (b) 0 stitch.

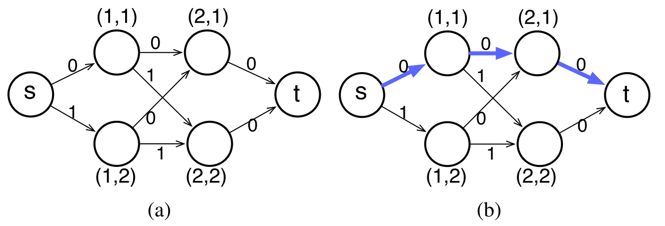


Fig. 11. (a) First stage to solve color assignment. In this example edge cost only considers the stitch number minimization. (b) One shortest path solution, where both cell 1 and 2 are assigned to coloring solution 1.

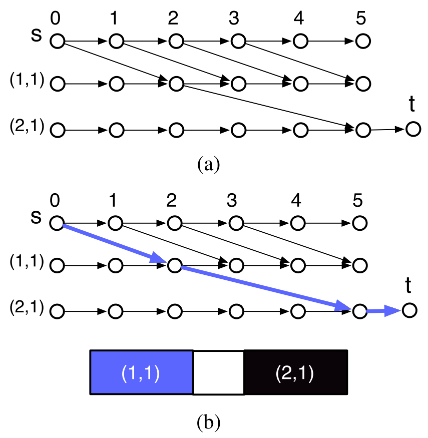


Fig. 12. (a) Second stage to solve detailed placement. (b) One shortest path solution corresponds to a cell placement.

color assignment cells, c_{i-1} and c_i, are assigned its pth and qth coloring solutions, then the width of cell c_i is changed from $w(i)$ to $w(i)+\text{LUT}(i-1, p, i, q)$. This way, the extra sites to resolve coloring conflicts are prepared for cell placement. Based on the updated cell widths, the graph model in [34] can be directly applied here. For instance, the second stage graph model for the example in Fig. 9 is illustrated in Fig. 12. It shall be noted that all cells have been assigned a coloring solution, thus the graph size is much smaller than that in Fig. 9. As shown in Fig. 12(b), the shortest path on the graph corresponds to a cell placement. The maximum displacement constraint can be easily considered through ignoring some diagonal edges.

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

VII. TPL-OSR With Maximum Displacement

In this section, we consider another single row placement problem, which is similar to the TPL-OSR, where the initial cell orders are determined. The slight difference here is that each cell is forbidden from moving more than distance M from its original location. The new problem is called TPL-OSR with maximum displacement. The motivation to study this new problem is twofold. First, in the previous TPL-OSR problem, although the two stage graph model can provide fast solutions due to the nature that the color assignment and cell placement are solved separately, its solution qualities may not be good enough. For the new problem, we are able to propose a fast but high performance optimization algorithm. Second, from the design perspective, a detailed placement technique with maximum displacement constraints is robust and important in practical situations. For example, if the initial placement is optimized toward other design metrics, e.g., timing, pin density, or routability, limiting cell displacements can help to maintain these metrics.

A. Problem Formulation

The problem we solve is finding new locations for all cells that preserve their relative order. Meanwhile, each cell has maximum moving distance, M, from its original location. In other words, for each cell c_i, its displacement range is defined as range(i) overlaps with $[x(i) - M, x(i) + M]$. Here $x(i)$ is the original position of cell c_i, while M is a user-defined parameter.

B. Linear Dynamic Programming Algorithm

Inspired by [22], our algorithm is based on linear dynamic programming, which means the optimal solution can be searched in linear time. The main idea is that we process cells starting from c_1 and explore cell pair locations for (c_1, c_2), followed by (c_2, c_3), etc. Once the optimal placements and color

TABLE II
NOTATIONS USED IN LINEAR DYNAMIC PROGRAMMING

$a(i)$	Color assignment value for c_i.
$d(i)$	Displacement value for c_i.
$t[i][d][a]$	Best cost for $c_1, \dots, c_i$ with $d(i) = d$ and $a(i) = a$.
$d[i][d][a]$	Displacement of c_{i-1} in an optimal sub-solution of $\{c_1, \dots, c_i\}$ with $d(i) = d$ and $a(i) = a$.
$a[i][d][a]$	Color assignment of c_{i-1} in an optimal sub-solution of $\{c_1, \dots, c_i\}$ with $d(i) = d$ and $a(i) = a$.
$r[i][d][a]$	whether $t[i][d][a]$ is inferior.

Algorithm 4 Linear Dynamic Programming

```
Require: Cells $C$ in row sites $R$;
1:  Initialize matrices $r, t, d$, and $a$; $F \leftarrow \infty$;
2:  for all $i = 2$ to $n$ do
3:      for all $a1 = 1$ to $v_{i-1}, a2 = 1$ to $v_i$ do
4:          for all $d1 = -M$ to $M, d2 = -M$ to $M$ do
5:              if $r[i-1][d1][a1] = 1$ or $x(i) + d1$ outside $range(i)$ or $x(j) + d2$ outside $range(j)$ then
6:                  continue;
7:              end if
8:              $y = t[i-1][d1][a1] + F_{i-1}(d1, a1, d2, a2)$;
9:              if $y < t[i][d2][a2]$ then
10:                 $t[i][d2][a2] \leftarrow y$;
11:                 $d[i][d2][a2] \leftarrow d1$;
12:                 $a[i][d2][a2] \leftarrow a1$;
13:             end if
14:         end for
15:     end for
16:     Mark inferior ones with $r[i][d][a] \leftarrow 1$;
17: end for
18: for $dn = -M$ to $M, an = 1$ to $v_n$ do
19:     if $t[n][dn][an] < F$ then
20:         $F \leftarrow t[n][dn][an]$;
21:         $d(n) \leftarrow dn$;
22:         $a(n) \leftarrow an$;
23:     end if
24: end for
25: for $i = n$ downto 2 do
26:     $d(i-1) \leftarrow d[i][d(i)][a(i)]$;
27:     $a(i-1) \leftarrow a[i][d(i)][a(i)]$;
28: end for
```

assignments for $c_1, \dots, c_{i-1}$ are computed we search the optimal placement and color assignment simultaneously for c_i. For convenience, Table II lists some additional notations used in the linear dynamic programming.

The details of the linear dynamic programming are shown in Algorithm 4. Line 1 initializes the solution costs. The main algorithmic computation takes place in the loops (lines 2–17). We iteratively explore all cell pairs (c_{i-1}, c_i), with different displacement values and color assignment solutions (lines 2–4). For cell pair (c_{i-1}, c_i) and different combinations of $(d1, a1, d2, a2)$, the best cost is stored in $t[i][d2][a2]$, while $d1$ and $a1$ are stored in $d[i][d2][a2]$ and $a[i][d2][a2]$, respectively (lines 9–13). $F_{i-1}(d1, a1, d2, a2)$ is the cost considering wire-length impact and stitch number, defined as follows:

$$\alpha \cdot \Delta\text{WL} + s(i-1, a1) + s(i, a2)$$

where ΔWL is the HPWL improvement of placing c_{i-1} and c_i in $x(i-1) + d1$ and $x(i) + d2$, respectively. $s(i-1, a1)$ and $s(i, a2)$ are used to calculate the stitch numbers. Here α is a user-defined parameter for assigning relative importance between the HPWL and the stitch number.

Different from the method in [22], we propose pruning techniques to speed-up the dynamic programming process. For any two solutions $t[i][d1][a]$ and $t[i][d2][a]$, if $t[i][d1][a] >= t[i][d2][a]$ and $d1 >= d2$, we can say $t[i][d1][a]$ is inferior to $t[i][d2][a]$. Then $r[i][d1][a]$ is assigned 1 to label the inferiority (line 16). Therefore, one can exit early when checking the r value (lines 5–7). Lines 18–24 compute the end case of the last cell in the row, and the solution is recovered at last (lines 25–28).

Theorem 1: The linear dynamic programming runs in $O(nK^2M^2)$ time to optimally solve the problem.

The complexity analysis results from the for loops (lines 2–17). Since both K and M are constants, the runtime complexity of Algorithm 4 is linear. Optimality stems from the fact that $t[i][\;][\;]$ explores all possible displacement values and color assignment solutions. It shall be noted that the runtime complexity using unified graph model in Section VI-B is $O(nmK)$. Since usually m is larger than n, the complexity of a unified graph model is quadratic and may be slower than linear dynamic programming.

VIII. EXPERIMENTAL RESULTS

A. Experimental Setup

We implement our standard cell precoloring and TPL aware detailed placement in C++, and all the experiments are performed on a Linux machine with 3.0 GHz CPU. Nangate 45 nm library [28] is scaled down to 16 nm technology node, as our initial standard cell library. We apply standard cell compliance and precoloring on the scaled library. During standard cell precoloring, each cell's maximum allowed stitch number, max$S$, is set to 2. Note that for some complex cells, the minimum stitch number required may be larger than max$S$. For these complex cells, max$S$ is set as the minimum stitch number required. We use Design Compiler [38] to synthesize OpenSPARC T1 designs based on the modified cell library. For each benchmark, we perform placement with Cadence SoC Encounter [39] to generate initial placement results. 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 0.85. Generally speaking, the higher utilization rate, the more difficult of the detailed placement. In our implementation, the α value is set to 10.

The benchmark statistics are listed in Table III. Column "K" is the maximum cell precoloring solution number among all standard cell types, which is related to the LUT size. Columns "cell #" and "row #" are the total cell module number and the total row number for each placement test case, respectively. Both "cell #" and "row #" reflect the placement problem size. To demonstrate the problem size of each single row placement, columns "max cell # per row" and "max m per row" are used.

TABLE III
BENCHMARK STATISTICS

bench	K	cell #	row#	max cell # / row	max m / row
alu-70	74	2110	43	67	330
alu-80	74	2110	41	68	302
alu-85	74	2110	39	67	299
byp-70	32	4416	81	75	597
byp-80	32	4416	75	84	564
byp-85	32	4416	73	89	545
div-70	74	3758	65	92	489
div-80	74	3758	61	94	456
div-85	74	3758	59	109	444
ecc-70	32	1322	42	42	322
ecc-80	32	1322	40	47	296
ecc-85	32	1322	38	45	293
efc-70	74	1183	40	45	300
efc-80	74	1183	37	43	283
efc-85	74	1183	36	44	274
ctl-70	74	1694	48	45	363
ctl-80	74	1694	45	54	339
ctl-85	74	1694	44	53	326
top-70	74	14793	123	146	919
top-80	74	14793	115	157	860
top-85	74	14793	112	158	832

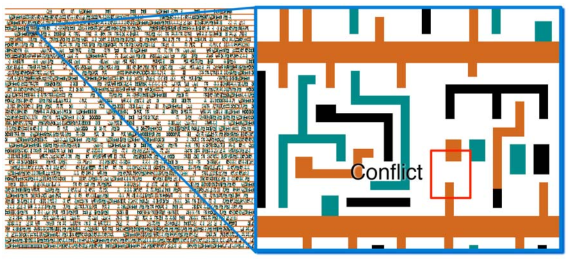
TABLE IV
COMPARISONS WITH POST-LAYOUT DECOMPOSITION

bench	Decomposor [14]		Our flow	
	CN#	ST#	CN#	ST#
alu-70	49	721	0	906
alu-80	62	750	0	945
alu-85	60	745	0	1004
byp-70	83	1534	0	1471
byp-80	84	1817	0	1612
byp-85	89	1707	0	1716
div-70	136	1908	0	1612
div-80	125	1488	0	1732
div-85	188	1867	0	1776
ecc-70	0	263	0	288
ecc-80	0	264	0	302
ecc-85	0	266	0	318
efc-70	89	647	0	794
efc-80	148	1178	0	862
efc-85	113	947	0	868
ctl-70	25	326	0	378
ctl-80	57	573	0	421
ctl-85	38	352	0	423
top-70	227	5704	0	5828
top-80	739	11995	0	6263
top-85	302	6192	0	6391
Average	**124.5**	1964.0	**0**	1710.0

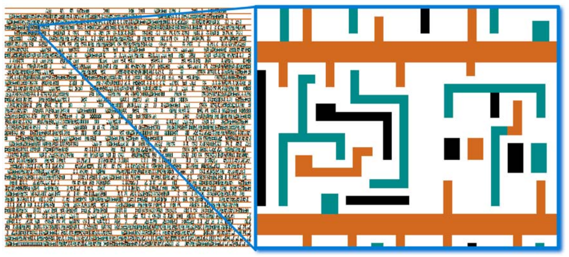
These columns represent maximum cell module number in one row, and maximum site number in one row, respectively.

B. Compare With Layout Decomposition Works

In the first experiment, we demonstrate the effectiveness of our TPL aware design flow compared to conventional TPL layout decomposition-based flow. The TPL layout decomposition-based flow consists of standard cell synthesis, placement, and TPL layout decomposition at post-stage. Our proposed flow integrates TPL constraints into standard cell synthesis and detailed placement, thus no layout decomposition is required on the whole chip layout. Table IV compares these flows for the M1 layers of all the benchmarks. In layout decomposition-based flow, Encounter is chosen as the placer. Column "decomposer [14]" lists the conflict number and the stitch number by an academic layout decomposers, while column "our flow" is the proposed TPL aware design flow. Layout modification and precoloring are carried out for each standard cell, and the unified graph model is applied to solve cell placement and color assignment simultaneously. Note that for each flow, the standard cell inner native conflicts have been removed through our compliance techniques (see Section III). In other words, theoretically the conflicts can only happen on the boundaries between standard cells.



(a)



(b)

Fig. 13. Examples of different design flows. (a) Even the layout is TPL friendly, layout decomposition may report unnecessary conflict. (b) Our TPL aware design flow can generate conflict-free output.

On one hand, we can see that in the layout decomposition-based flow, even input layout itself is TPL-friendly on average more than 120 unnecessary conflicts are reported. Due to the large number of conflicts, a lot of efforts may be required to manually modify or migrate the layout to resolve the conflicts. On the other hand, through considering TPL constraints in early design stages, our proposed TPL aware design flow can guarantee zero conflict. Fig. 13 compares the outputs from decomposer [14] and our flow. In Fig. 13(a), we can see that one unnecessary conflict is reported, while in Fig. 13(b) our flow can generate conflict-free layout.

C. Detailed Placement Algorithm Comparison

In Sections VI and VII, we have proposed several algorithms to solve TPL aware single row detailed placement. In the second experiment, we analyze the performances of the proposed algorithms and related speed-up techniques in Table V. Column "GREEDY" is a greedy detailed placement algorithm [20], which is implemented as our baseline. Although [20] is targeting the self-aligned double patterning, the proposed detailed placement algorithm can be modified to be integrated into our framework. The detailed placement in [20] is greedy-based. Rows of placement are processed one by one, left to right. For each pair of conflicting cells, we first check if the conflict can be resolved through cell flipping, if no then cell spreading would be carried out. Columns "TPLPlacer" and "TPLPlacer-2Stage" are detailed placement algorithms with different TPL-OSR engines. TPLPlacer utilizes the optimal unified graph model, while TPLPlacer-2Stage uses fast two-stage graph models to solve color assignment and cell placement iteratively.

TABLE V
COMPARISONS OF DETAILED PLACEMENT ALGORITHMS

bench	D_{disp}	GREEDY [20]			TPLPlacer			TPLPlacer-2Stage			TPLPlacer-MDP		
	(um)	ΔWL	ST#	CPU(s)	ΔWL	ST#	CPU(s)	ΔWL	ST#	CPU(s)	ΔWL	ST#	CPU(s)
alu-70	10	-0.30%	1016	3.50	-2.08%	906	7.44	-2.10%	1109	2.12	-1.61%	909	4.03
alu-80	10	-0.55%	1015	1.87	-1.88%	945	7.19	-1.89%	1107	2.01	-1.85%	945	3.90
alu-85	15	N/A	N/A	2.00	-1.14%	1004	15.34	-0.96%	1102	2.24	-1.20%	1005	5.50
byp-70	10	-0.37%	1756	1.14	-1.69%	1471	15.92	-1.68%	1967	1.84	-1.55%	1450	6.10
byp-80	10	-0.06%	1767	1.17	-1.25%	1612	15.51	-1.24%	1956	1.69	-1.16%	1598	5.82
byp-85	15	N/A	N/A	1.53	-1.33%	1716	35.59	-1.33%	1981	2.59	-1.29%	1713	10.14
div-70	10	-0.14%	1852	3.06	-1.96%	1612	11.49	-1.97%	1924	3.33	-1.92%	1604	5.58
div-80	10	0.10%	1867	3.13	-1.31%	1732	11.45	-1.32%	1974	3.49	-1.33%	1724	5.29
div-85	15	N/A	N/A	3.19	-1.31%	1776	22.58	-1.32%	1965	4.06	-1.26%	1771	7.58
ecc-70	10	-0.15%	356	0.60	-1.50%	288	3.14	-1.50%	333	0.69	-1.51%	288	1.63
ecc-80	10	0.19%	352	0.60	-0.67%	302	3.18	-0.67%	336	0.67	-0.72%	303	1.53
ecc-85	15	0.50%	351	0.61	-0.51%	318	3.61	-0.51%	343	0.72	-0.50%	316	1.46
efc-70	10	-0.85%	885	1.52	-3.79%	794	3.83	-3.86%	901	1.59	-3.78%	787	2.42
efc-80	10	-0.68%	896	1.50	-3.23%	862	3.76	-3.27%	933	1.59	-3.04%	859	2.27
efc-85	15	0.52%	905	1.57	-1.67%	868	4.13	-1.69%	928	1.76	-1.65%	862	2.23
ctl-70	10	-1.04%	487	2.09	-2.78%	378	5.48	-2.78%	505	2.33	-2.78%	373	3.43
ctl-80	10	0.55%	468	2.08	-1.42%	421	5.46	-1.43%	490	2.20	-1.38%	417	3.21
ctl-85	15	0.32%	460	2.12	-1.03%	423	6.15	-1.03%	481	2.26	-1.00%	425	3.31
top-70	10	-0.16%	6487	5.66	-1.60%	5828	81.02	-1.61%	7087	8.76	-1.65%	5776	17.35
top-80	10	0.45%	6530	5.82	-0.79%	6263	80.49	-0.80%	7238	8.62	-0.78%	6236	16.16
top-85	15	0.86%	6467	7.14	-0.57%	6391	86.41	-0.58%	7059	10.04	-0.56%	6388	16.55
Average		N/A	N/A	2.47	-1.595%	1710	20.44	-1.598%	1986.6	2.03	-1.549%	1702.3	4.36
Ratio					**-1.0**	**1.0**	**1.0**	**-1.002**	**1.162**	**0.10**	**-0.971**	**0.996**	**0.21**

In addition, column "TPLPlacer-MDP" is to apply the linear dynamic programming method (see Section VII). Here the maximum displacement value, M, is set to 8. Column "D_{disp}" lists the maximum displacement constraint for each test case. For each algorithm we list several metrics "ST#," "ΔWL," and "CPU(s)." "ST#" is the stitch number on the final decomposed layout. ΔWL is the total wire-length difference before and after our TPL aware placement, where HPWL is applied to calculate the total wire-length. Column CPU(s) gives the detailed placement process runtime in seconds.

From column GREEDY we can see that the greedy method is very fast. However, in 3 out of 21 cases it cannot find legal placement solutions. For each illegal result "N/A" is labeled in Table V. The main reason for these illegal solutions is that GREEDY only shifts the cells right. Therefore, due to the greedy nature, for a benchmark case with high cell utilization it may cause final placement violation. Meanwhile, since the color assignment is solved through greedy method as well, it loses the global view to minimize the stitch number. We can observe that more stitches are reported for those cases where it finds out legal results.

We further compare two TPL-OSR algorithms: TPLPlacer and TPLPlacer-2Stage. Comparing these two columns we can see that both of them can yield very similar wire-length improvement (around 1% wire-length reduction). In TPLPlacer-2Stage the unified graph is divided into two independent graphs, so the graph size can be reduced. Due to the smaller graph size, TPLPlacer-2Stage can get 10× speed-up against TPLPlacer. However, TPLPlacer-2Stage introduces 16% more stitch numbers. The possible reason is that under the 2-stage graph model, placement and color assignment are optimized separately, and then this speed-up technique may lose some optimality in terms of stitch number.

From column TPLPlacer-MDP we can see that the linear dynamic programming technique has a better trade-off to optimize wire-length and stitch number together. That is, TPLPlacer-MDP achieves nearly the same wire-length and stitch number results comparing with TPLPlacer. Meanwhile, TPLPlacer-MDP is around 5× faster than the unified graph model in TPLPlacer. The reason is that TPLPlacer-MDP is a linear runtime algorithm in terms of n, while TPLPlacer has nearly a quadratic runtime complexity.

The TPLPlacer-MDP is implemented with $M = 8$. In other words, each cell c_i has $2M+1$ possible new positions between $[x(i) - M, x(i) + M]$. Since the M value determines the placement solution space, it impacts the performance of detailed placement a lot. Therefore, to demonstrate the robustness of TPLPlacer-MDP, it would be interesting to analyze the performance with different M settings. Fig. 14 gives such analysis for test cases *alu_70, alu_80*, and *alu_85*. From Fig. 14(a) and (b), we can see that with different M values, TPLPlacer-MDP can achieve similar stitch number and wire-length improvement. Note that in Fig. 14(b), we observe sometimes smaller M value may lead to little better wire-length. The possible reason is that when M is smaller, we cannot find legal placement solution in a single row, thus global moving would be carried out. Due to the additional global moving effort, the total wire-length can be improved. It is not hard to see from Fig. 14(c) that the runtime is related to the M value. Although the runtime complexity is quadratic to M value, due to pruning technique the practical runtime is nearly a linear function of M. Therefore, we can conclude that TPLplacer-MDP is very robust and insensitive to the M value. In our implementation, M is set as a small value 8, to maintain both good speed-up and good performance.

α is a user-defined parameter for assigning relative importance between the HPWL and the stitch number. In Table V,

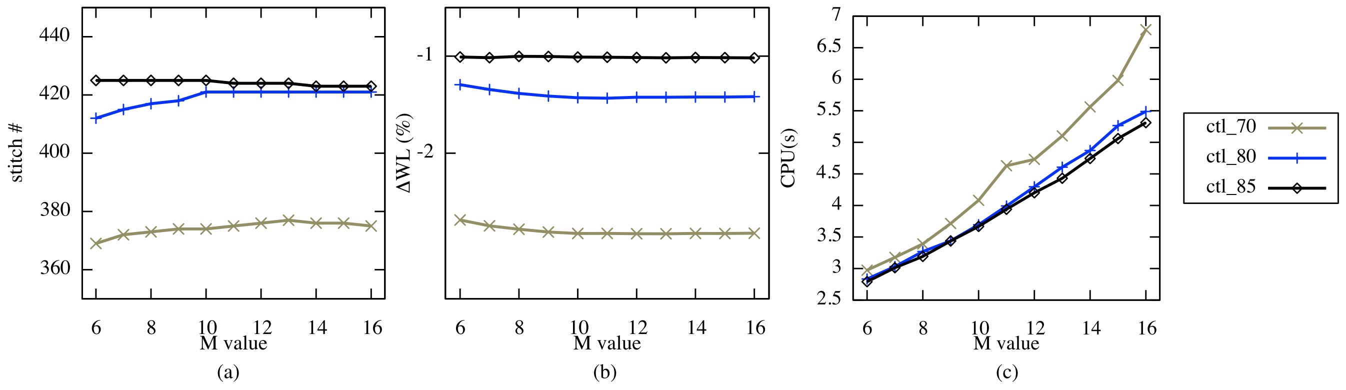


Fig. 14. TPLPlacer-MDP performance analyzes with different M values for alu design cases. (a) Impact on stitch numbers. (b) Impact on wire-length improvements (ΔWL). (c) Impact on runtimes.

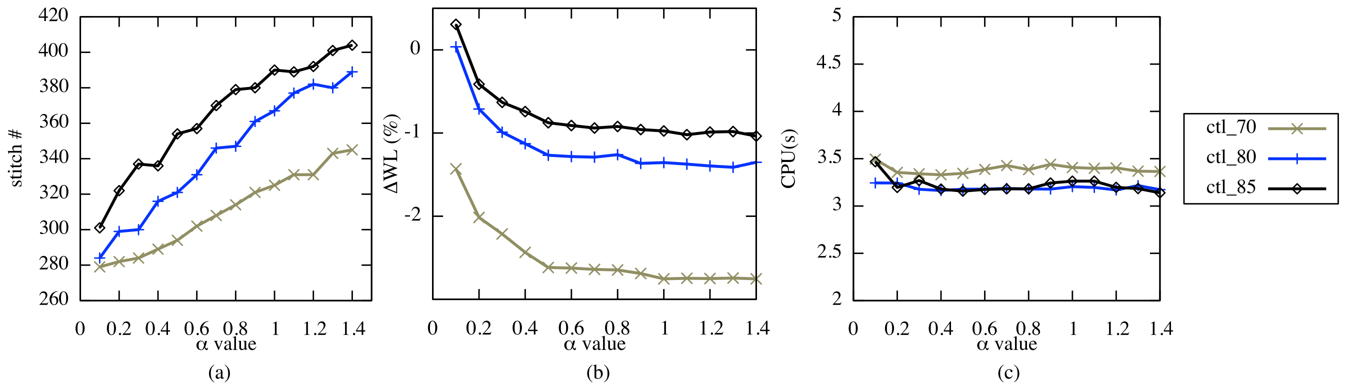


Fig. 15. Performance analyzes with different α values for ctl design cases. (a) Impact on stitch numbers. (b) Impact on wire-length improvements (ΔWL). (c) Impact on runtimes.

α is set to 10. Fig. 15 analyzes the TPLPlacer-MDP performance with different α settings for test cases *ctl_70, ctl_80*, and *ctl_85*. From Fig. 15(a) and (b), we can see that with smaller α value, TPLPlacer-MDP can achieve smaller stitch number, but longer wire-length. In other words, if smaller α is selected, stitch minimization would be emphasized during our TPL aware detailed placement; if larger α is applied, wire-length optimization would be more important.

IX. Conclusion

In this paper, we have proposed a coherent framework to seamlessly integrate the TPL aware optimizations into early design stages, e.g., standard cell compliance, standard cell precoloring, and detailed placement. To the best of our knowledge, this is the first work for TPL compliance at both standard cell and placement levels. For the TPL-OSR problem, an optimal graph model to simultaneously solve cell placement and color assignment is proposed. Then, a two-stage graph model is applied to speed-up the algorithm. To alleviate the impact on solution quality, another speed-up technique is presented to solve the TPL-OSR with maximum displacement constraints. The results show that considering TPL constraints in early design stages can dramatically reduce the conflict number and stitch number in the final layout.

As continuing shrinking of technology node to sub-16 nm, TPL turns out to be a definitely promising lithography solution. A dedicated design flow that integrates TPL constraints is necessary to assist in the whole process. We believe this paper will stimulate more research on TPL and TPL aware design. Future works would include how the coloring and placement methodologies can be extended to resolve conflicts across cell rows or multirow height cells. Another line of research would explore the parallelization of detailed placement for different rows, though the overall placement scheme may have to be modified to accommodate global moving during parallelization.

Acknowledgment

The authors would like to thank Dr. L. Liebmann at IBM for helpful discussions on this problem.

References

[1] (Feb. 2015). *ITRS*. [Online]. Available: http://www.itrs.net
[2] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1453–1472, Oct. 2013.
[3] B. Yu *et al.*, "Dealing with IC manufacturability in extreme scaling," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2012, pp. 240–242.
[4] K. Yuan, B. Yu, and D. Z. Pan, "E-beam lithography stencil planning and optimization with overlapped characters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, pp. 167–179, Feb. 2012.
[5] C. Wagner and N. Harned, "EUV lithography: Lithography gets extreme," *Nat. Photon.*, vol. 4, no. 1, pp. 24–26, 2010.
[6] K. Lucas *et al.*, "Implications of triple patterning for 14 nm node design and patterning," *Proc. SPIE*, vol. 8327, Mar. 2012, Art. ID 832703.

[7] C. Cork, J.-C. Madre, and L. Barnes, "Comparison of triple-patterning decomposition algorithms using aperiodic tiling patterns," *Proc. SPIE*, vol. 7028, May 2008, Art. ID 702839.

[8] R. S. Ghaida, K. B. Agarwal, L. W. Liebmann, S. R. Nassif, and P. Gupta, "A novel methodology for triple/multiple-patterning layout decomposition," *Proc. SPIE*, vol. 8327, Mar. 2012, Art. ID 83270M.

[9] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2011, pp. 1–8.

[10] S.-Y. Fang, Y.-W. Chang, and W.-Y. Chen, "A novel layout decomposition algorithm for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 397–408, Mar. 2014.

[11] H. Tian, H. Zhang, Q. Ma, Z. Xiao, and M. Wong, "A polynomial time triple patterning algorithm for cell based row-structure layout," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2012, pp. 57–64.

[12] J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–6.

[13] B. Yu, J.-R. Gao, and D. Z. Pan, "Triple patterning lithography (TPL) layout decomposition using end-cutting," *Proc. SPIE*, vol. 8684, Mar. 2013, Art. ID 86840G.

[14] B. Yu *et al.*, "A high-performance triple patterning layout decomposer with balanced density," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 163–169.

[15] Y. Zhang, W.-S. Luk, H. Zhou, C. Yan, and X. Zeng, "Layout decomposition with pairwise coloring for multiple patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 170–177.

[16] L. Liebmann, D. Pietromonaco, and M. Graf, "Decomposition-aware standard cell design flows to enable double-patterning technology," *Proc. SPIE*, vol. 7974, Apr. 2011, Art. ID 79740K.

[17] T.-C. Chen, M. Cho, D. Z. Pan, and Y.-W. Chang, "Metal-density-driven placement for CMP variation and routability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 12, pp. 2145–2155, Dec. 2008.

[18] S. Hu, P. Shah, and J. Hu, "Pattern sensitive placement perturbation for manufacturability," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 6, pp. 1002–1006, Jun. 2010.

[19] M. Gupta, K. Jeong, and A. B. Kahng, "Timing yield-aware color reassignment and detailed placement perturbation for bimodal CD distribution in double patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 8, pp. 1229–1242, Aug. 2010.

[20] J.-R. Gao, B. Yu, R. Huang, and D. Z. Pan, "Self-aligned double patterning friendly configuration for standard cell library considering placement," *Proc. SPIE*, vol. 8684, March 2013, Art. ID 868406.

[21] K. B. Agarwal, C. J. Alpert, Z. Li, G.-J. Nam, and N. Viswanathan, "Multi-patterning lithography aware cell placement in integrated circuit design," U.S. Patent 8 495 548, Jul. 23, 2013.

[22] T. Taghavi *et al.*, "New placement prediction and mitigation techniques for local routing congestion," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2010, pp. 621–624.

[23] Q. Ma, H. Zhang, and M. D. F. Wong, "Triple patterning aware routing and its comparison with double patterning aware routing in 14 nm technology," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 591–596.

[24] Y.-H. Lin, B. Yu, D. Z. Pan, and Y.-L. Li, "TRIAD: A triple patterning lithography aware detailed router," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2012, pp. 123–129.

[25] K. Yuan and D. Z. Pan, "WISDOM: Wire spreading enhanced decomposition of masks in double patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2010, pp. 32–38.

[26] S.-Y. Fang, S.-Y. Chen, and Y.-W. Chang, "Native-conflict and stitch-aware wire perturbation for double patterning technology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 5, pp. 703–716, May 2012.

[27] R. S. Ghaida *et al.*, "Layout decomposition and legalization for double-patterning technology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 2, pp. 202–215, Feb. 2013.

[28] (Feb. 2015). *NanGate FreePDK45 Generic Open Cell Library*. [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib

[29] *Calibre Verification User'S Manual*, Mentor Graph. Corp., Wilsonville, OR, USA, 2008.

[30] (Feb. 2015). *Predictive Technology Model Ver. 2.1*. [Online]. Available: http://ptm.asu.edu

[31] R. Neapolitan and K. Naimipour, *Foundations of Algorithms*. Sudbury, MA, USA: Jones and Bartlett, 2010.

[32] W.-K. Chow, J. Kuang, X. He, W. Cai, and E. F. Young, "Cell density-driven detailed placement with displacement constraint," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Petaluma, CA, USA, 2014, pp. 3–10.

[33] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 28, no. 1, pp. 12–18, Jan. 1981.

[34] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2005, pp. 891–898.

[35] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proc. IEEE/ACM Proc. Design Autom. Test Europe (DATE)*, Paris, France, 1998, pp. 321–324.

[36] A. B. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of linear placements for wirelength minimization with free sites," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Hong Kong, 1999, pp. 241–244.

[37] U. Brenner and J. Vygen, "Faster optimal single-row placement with fixed ordering," in *Proc. IEEE/ACM Design Autom. Test Europe (DATE)*, Paris, France, 2000, pp. 117–121.

[38] (Feb. 2015). *Synopsys IC Compiler*. [Online]. Available: http://www.synopsys.com

[39] (Feb. 2015). *Cadence SoC Encounter*. [Online]. Available: http://www.cadence.com

Bei Yu (S'11–M'14) received the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 2014.

He is a Post-Doctoral Research Scholar with the Department of Electrical and Computer Engineering, University of Texas at Austin. His current research interests include design for manufacturability and optimization algorithms.

Dr. Yu was the recipient of the two Best Paper Awards at the International Conference on Computer-Aided Design (ICCAD) in 2013, the Asia and South Pacific Design Automation Conference in 2012, the three other Best Paper Award Nominations at Design Automation Conference in 2014, Asia and South Pacific Design Automation Conference (ASPDAC)'13, and ICCAD'11, the Chinese Government Award for Outstanding Students Abroad in 2013, the Society of Photo-Optical Instrumentation Engineer's Education Scholarship in 2013, the Silver Medal in ACM Student Research Contest at ICCAD'13, and the IBM Ph.D. Scholarship in 2012.

Xiaoqing Xu (S'15) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree in electrical and computer engineering, University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include robust standard cell design, design for manufacturability, and physical design.

Mr. Xu was the recipient of the Microelectronics and Computer Development Fellowship from the University of Texas at Austin in 2012 and the Excellent Graduate Award from Peking University in 2012.

Jhih-Rong Gao received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

She was a Research and Development Engineer at Synopsys, Inc., Taipei, Taiwan, from 2007 to 2009. She is a Senior Technical Staff with Cadence Design Systems, Inc., Austin. Her current research interests include physical design and design for manufacturability.

Dr. Gao was the recipient of the BACUS Photomask Scholarship from the Society of Photo-Optical Instrumentation Engineers Education in 2013.

Yibo Lin received the B.S. degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include physical design and design for manufacturability.

Zhuo Li (S'01–M'05–SM'09) received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2005.

From 2006 to 2014, he was a Research Staff Member at IBM Watson T. J. Research Center, Yorktown Heights, NY, USA, and Austin Research Laboratory, Austin, TX, USA, where he developed IBM flagship physical design tools including physical synthesis, interconnect synthesis, placement, optimization, timing and noise-driven routing, and flow integration. He was a Co-Founder of Pextra Corporation, College Station, which was a start-up specializing in parasitic extraction and acquired by Mentor Graphics Corporation, Wilsonville, OR, USA, in 2009. In 2014, he joined Cadence Design Systems, Austin, where he is a Software Architect of next generation digital design flow. He has published over 70 conference and journal papers and holds 61 filed patents with 43 issued.

Dr. Li was the recipient of five IBM Outstanding Technical Achievement Awards, four IBM Research O-Level Accomplishment Awards, the ASPDAC Best Paper Award, the Circuits and Systems Society (CAS) Outstanding Young Author Award, the International Symposium on Physical Design (ISPD) Best Paper Nominee, the International Conference on Computer-Aided Design (ICCAD) Best Paper Nominee, and the International Symposium on Quality Electronic Design Best Paper Nominee, the IEEE/ACM Service Awards ACM Special Interest Group on Design Automation (SIGDA) Technical Leadership Award, the SRC Mahboob Khan Outstanding Industry Liaison/Associate Award, the Design Automation Conference (DAC) Service Award, the IEEE Region Five Outstanding Individual Member Achievement Award twice, and the IEEE Council on Electronic Design Automation Early (CEDA) Career Award in 2013 as the First Industry Winner. He was the Chapter Chair of the IEEE CAS/Solid-State Circuits Society Chapter of Central Texas Section, which won the IEEE Circuits and Systems Society in 2014 Chapter of the Year Award, the 2011 Region Seven Chapter of the Year Award, and the IEEE Solid State Circuits Society in 2011 Outstanding Chapter Award. He served as the Technical Program Committee (TPC) Sub-Committee Chair and as a Committee Member for several major conferences such as DAC, ICCAD, and Design Automation and Test in Europe in Electronic Design Automation area. He served as the Co-Chair of DAC Ph.D. from 2014 to 2015, the Chair of computer-aided design (CAD) Contest in 2013 and the Contest Chair of the 2011 and 2012 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU) Power Grid Simulation Contest. He was the Founding Chair of the IEEE CEDA Chapter of Central Texas Section. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He currently serves as the Secretary of the IEEE Central Texas Section.

Charles J. Alpert (F'05) received a dual degrees in history and math and computational sciences from Stanford University, Stanford, CA, USA, in 1991, and the Doctoratal degree in computer science from the University of California, Los Angeles, Los Angeles, CA, USA, in 1996.

He was at Design Productivity Group, IBMs Austin Research Laboratory, Austin, TX, USA, where he specializes in algorithmic innovations and develops physical design automation tools. In 2014, he joined Cadence Design Systems, Austin, where he is currently with the Advanced Technology Flow Team, whose mission is to drive innovations across teams from synthesis to sign-off. He has published over 100 conference and journal papers.

Dr. Alpert was the recipient of the Best Paper Award from the ACM/IEEE Design Automation Conference thrice. He served as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN for ten years, where he is currently the journal's Deputy Editor-in-Chief. He was active in the academic community, serving as the Chair of the TAU Workshop on Timing Issues and the International Symposium on Physical Design. He serves as the Vice Chair of the IEEE/ACM Design Automation Conference, and he will be the Chair of the conference when it is hosted in Austin, in 2016.

David Z. Pan (S'97–M'00–SM'06–F'14) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California, Los Angeles (UCLA), Los Angeles, CA, USA.

From 2000 to 2003, he was a Research Staff Member at IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He is currently the Engineering Foundation Endowed Professor with the Department of Electrical and Computer Engineering, University of Texas (UT) at Austin, Austin, TX, USA. His current research interests include cross-layer nanometer IC design for manufacturability/reliability, new frontiers of physical design, and CAD for emerging technologies such as 3-D-IC, bio, and nanophotonics. He has published over 200 papers in refereed journals and conferences. He holds eight U.S. patents.

Prof. Pan was the recipient of the SRC 2013 Technical Excellence Award, the Design Automation Conference (DAC) Top Ten Author in Fifth Decade, the DAC Prolific Author Award, 11 Best Paper Awards, several International CAD Contest Awards, the Communications of the ACM Research Highlights in 2014, the ACM/SIGDA Outstanding New Faculty Award in 2005, the National Science Foundation CAREER Award in 2007, the SRC Inventor Recognition Award thrice, the IBM Faculty Award four times, the UCLA Engineering Distinguished Young Alumnus Award in 2009, and the UT Austin RAISE Faculty Excellence Award in 2014. He served as a Senior Associate Editor of *ACM Transactions on Design Automation of Electronic Systems*, an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II, *Science China Information Sciences*, the *Journal of Computer Science and Technology*, and the IEEE CAS Society Newsletter. He served as the Chair of the IEEE Computer-Aided Network DEsign Committee and the ACM/SIGDA Physical Design Technical Committee, the Program/General Chair of ISPD, TPC Subcommittee Chair of DAC, International Conference on Computer-Aided Design (ICCAD), ASPDAC, ISLPED, ICCD, the Tutorial Chair of DAC'14, and the Workshop Chair for ICCAD'15, among others.