Triple Patterning Aware Detailed Placement Toward Zero Cross-Row Middle-of-Line Conflict

Yibo Lin, Bei Yu, *Member, IEEE*, Biying Xu, and David Z. Pan, *Fellow, IEEE*

***Abstract*—Triple patterning lithography (TPL) is one of the most promising lithography technology in sub-14-nm technology nodes, especially for complicated low metal layer manufacturing. To overcome the intracell routability problem and improve the cell regularity, recently middle-of-line (MOL) layers are employed in standard cell design. However, MOL layers may introduce a large amount of cross-row TPL conflicts for row-based design. Motivated by this challenge, in this paper we propose the first TPL aware detailed placement toward zero cross-row MOL conflict. In standard cell precoloring, Boolean-based look-up table is proposed to reduce solution space. In detailed placement stage, three powerful techniques, i.e., local reordered single row refinement, min-cost flow-based conflict removal, and local cell interleaving, are proposed to provide zero TPL conflict solution. The experimental results demonstrate the effectiveness of our proposed methodologies.**

***Index Terms*—Design for manufacture, detailed placement, middle-of-line (MOL), triple patterning lithography (TPL).**

I. Introduction

With the scaling of the feature size in sub-14-nm technology nodes, semiconductor industry is challenged by the manufacturability with conventional 193-nm wavelength immersion (193i) lithography. Although under intensive research and development, the next generation lithography techniques, such as extreme ultraviolet lithography, directed self-assembly, electron beam lithography, and nanoimprint lithography, are postponed due to yield and throughput issues [1], [2]. Multiple patterning lithography (MPL), which reuses conventional 193i lithography, currently has been heavily utilized in industry [3]–[5]. For instance, double patterning lithography has been introduced in 14-nm technology node [6]. In emerging sub-14-nm technology nodes, LELELE type triple patterning lithography (TPL) is a very promising option for complicated low metal layer manufacturing [7].

As a fundamental and critical problem, layout decomposition for TPL or even general MPL has been studied extensively in [8]–[16]. Due to the $\mathcal{NP}$-hardness [17], most layout decomposers apply heuristic methods to search for near-optimal solutions, thus there may be a large amount of conflicts left in decomposed layout. To overcome this limitation, several studies integrate patterning constraints in early design stages. For example, how to introduce TPL friendly design in detailed routing has been discussed in [18]–[21]. Besides, there are several studies proposing different methodologies for TPL aware detailed placement [22]–[27]. For ordered single row (OSR) problem, Yu *et al.* [22], [23] proposed a unified graph model to cell placement and coloring assignment, which was further improved by a technique based on dynamic programming for quality and efficiency. The dynamic programming algorithm to solve OSR problem can achieve further speedup if the cost of each cell is independent to other cells in a row [28], [29], but it is not applicable to TPL awareness because the conflict cost is determined by two adjacent cells. Kuang *et al.* [24] and Chien *et al.* [25] recently further improved the detailed placement solutions in [22]. A special case of TPL aware single row placement was considered in [26] and [27], where each type of standard cell has only one final coloring solution. Lin *et al.* [30] summarized recent detailed placement challenges and techniques for advanced nodes.

Since the 20-nm technology node and beyond, to overcome the intracell routability problem and improve the cell regularity, a Tungsten-based middle-of-line (MOL) structure is employed for standard cell design [31]. MOL structure is made up of two different local interconnection layers, namely CA and CB (also called IM1 and IM2 [32]). An example of MOL enabled standard cell structure is shown in Fig. 1(a), where CA and CB layer features are labelled as blue and red, respectively. We can see from Fig. 1(b), if placement is not carefully designed, on CA layer there are several native TPL conflicts cross different standard cell rows (see the red edge which denotes a coloring conflict). Therefore, for advanced technology nodes where MOL layers are employed, standard cell coloring and placement should take the cross-row conflicts into account. However, all existing TPL aware placement works assume there is no conflict between cells in different rows.

In this paper, motivated by the particular structures of MOL layers, we propose a comprehensive study to TPL aware detailed placement to overcome cross-row TPL conflicts. To the best of our knowledge, this paper is the first TPL aware

Manuscript received July 5, 2016; revised October 2, 2016; accepted December 8, 2016. Date of publication January 5, 2017; date of current version June 16, 2017. This work was supported in part by the National Science Foundation under Project CCF-1218906, in part by the Semiconductor Research Corporation under Project 2414.001, and in part by the Chinese University of Hong Kong Direct Grant for Research. A preliminary version has been presented at the International Conference on Computer-Aided Design in 2015. This paper was recommended by Associate Editor E. Young.

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

B. Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

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.2017.2648843

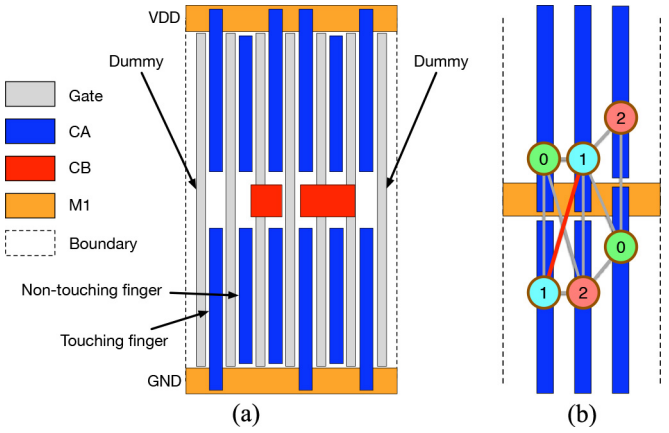

Fig. 1. Example of MOL-based standard cell structure. (a) MOL consists of CA and CB layers. (b) If placement is not carefully designed, CA layer may involve lots of TPL cross-row conflicts. The circles denote the abstraction of CA layer features in the conflict graph and an edge between two circles denote that the corresponding features should be printed by different masks. The number in each circle represents the mask/color used for the feature.

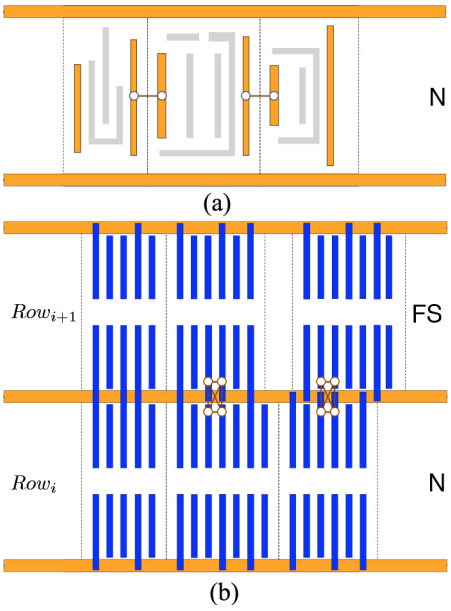

Fig. 2. Example of conflicts on (a) intrarow metal-1 layer and (b) cross-row MOL layer.

detailed placement targeting at cross-row conflict removal. The contributions of this paper can be highlighted as follows.

1) We carry out a comprehensive study on standard cell level coloring strategy and Boolean-based look-up table (BLUT) construction for MOL structures.
2) We propose single row placement techniques that allow local reordering for conflict and stitch minimization.
3) We develop a concurrent approach for multirow conflict removal.
4) We propose a dynamic programming algorithm to solve general q-partition interleaving for post wirelength and stitch refinement.
5) Experimental results demonstrate the effectiveness of our proposed framework.

The rest of this paper is organized as follows. Section II shows the definitions of related concepts and our overall flow. Sections III and IV propose the cell level decomposition and TPL aware detailed placement, respectively. Section V gives the experimental results, followed by the conclusion in Section VI.

II. Preliminaries and Overall Flow

A. MOL Structure

As shown in Fig. 1(a), MOL layers typically include CA and CB [32]. In this paper, we focus on CA layer colorability since this layer is more likely to cause TPL conflicts across different rows. In a row-based layout structure, standard cells are aligned to placement rows with identical height. Horizontal power and ground (PG) rails are shared by neighboring rows. Thus neighboring rows have to be aligned in a back-to-back manner; i.e., a row orientated to *N* must have neighbors orientated to *FS*, as shown in Fig. 2(b), vice versa. CA features can touch the PG rails. For simplicity, we define a *touching finger* as a CA feature that touches the PG rails, and a *non-touching finger* as a CA feature that does not hit any PG rail. All fingers are aligned to specific grids since they are aligned between features of gate layer according to the self-aligned manufacturing process. Due to the existence of touching fingers, it is very likely to introduce conflicts between cross-row CA features. Fig. 1(b) shows an example of coloring failure caused by cross-row conflicts if only three masks are available. We also assume that there is no conflict between MOL fingers within a standard cell.

B. Colorability Analysis for Advanced Standard Cell

In advanced standard cell design, the standard cell colorability is different from conventional one (e.g., 45-nm technology node). On one hand, due to the employment of MOL layers for local connection near PG rails, there is no cross-row conflict on metal-1 layers and the metal-1 layer conflict can only happen between two abutting cells in the same row, as shown in Fig. 2(a). On the other hand, if cells are not carefully placed, there would be a large amount of cross-row conflicts on MOL layers. Fig. 2(b) shows such conflicts.

C. Overview of Proposed Flow

The overall flow of our methodologies is shown in Fig. 3. There are two main phases, standard cell phase and detailed placement phase. In the standard cell phase, given standard cell library as the input, we perform precoloring for metal layers in standard cells, generate look-up table (LUT) for metal-1 layer and extract MOL features for next phase. In the second phase, TPL aware detailed placement is performed to optimize wirelength, assign coloring solution, and minimize conflicts and stitches. This phase consists of different placement approaches with TPL constraint consideration and post placement color assignment for MOL layer. The density-driven global move is very similar to [33], so we skip it for brevity. The output of the framework is decomposed layouts with optimized placement solution and color assignment for both metal-1 and MOL layer.

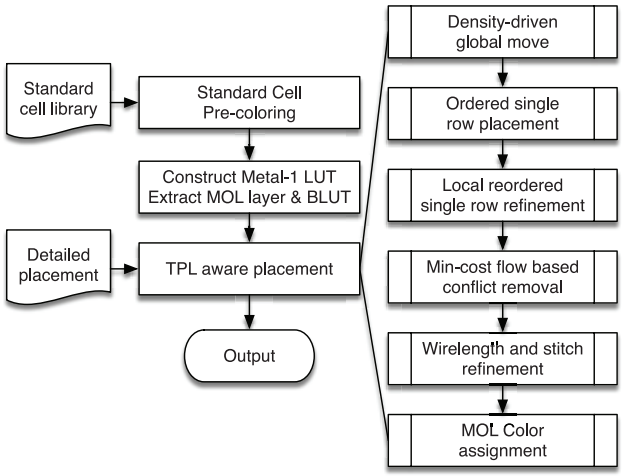

Fig. 3. Overall flow of the methodologies for TPL detailed placement.

III. Standard Cell Layout Decomposition

In TPL, features of a certain layer are decomposed into three masks. Two features that are too close to each other cannot be assigned to the same mask; otherwise, it will introduce a conflict. For standard cells, we consider metal-1 layer and MOL layer for TPL layout decomposition.

A. Standard Cell Precoloring

For metal-1 layer, the cell precoloring problem in a row-based layout structure is the same as that in [22]–[24]. That is, given a standard cell and coloring distance for metal-1 layer, generate all candidate coloring solutions where no two solutions are redundant to each other [23]. The redundancy is defined by the fact that one candidate coloring solution has the same color assignment to left and right boundary features as another solution of the cell. Although the solution space for a whole standard cell can be very large, there is no need to enumerate all possible coloring solutions for placement. Due to the fact that conflicts come from boundary features of cells in the TPL detailed placement problem, only boundary conditions need to be considered. In our metal-1 layer decomposition flow, we apply the backtracking algorithm from [23] and enumerate all color combinations for boundary features. Features not belong to the boundaries are assigned with any color combination that carries out a legal coloring solution. Due to limited number of boundary features, it is applicable to store the coloring solutions as an input for detailed placement stage.

For MOL layer precoloring, similar to metal-1 layer, we only consider coloring solutions of features that are close to cell top or cell bottom. As suggested in [34], we forbid stitching for CA layer. At first glance, the precoloring problems of metal-1 layer and MOL layer are very similar. However, the difference lies in the problem size. In advanced technology node, for metal-1 layer, there are usually fewer than five boundary features for left or right side of a standard cell, while the number of fingers for MOL layer is much larger. Usually large cells have even more fingers; e.g., a D flip-flop has more than twenty fingers at top boundary.

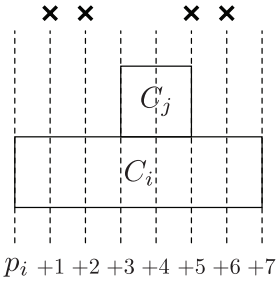

Fig. 4. Example of discrete conflict-free offset values between two vertically stacked cells for LUT of MOL layer.

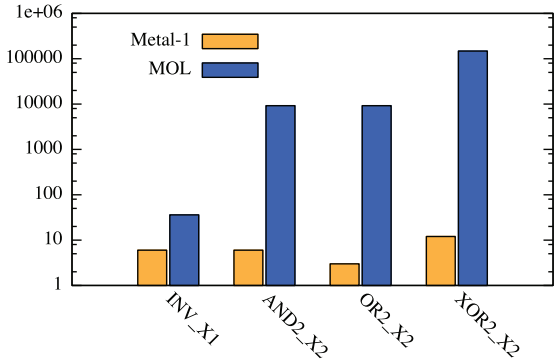

Fig. 5. Example of LUT sizes for metal-1 and MOL layers.

B. BLUT Construction

According to previous analysis, we generate precoloring solutions of metal-1 layer for each cell and precompute the minimum colorable distance for every cell pair with different coloring solutions. The distances are stored in an LUT.

For MOL layer, the LUT is discrepant to that of metal-1. Fig. 4 gives an example to explain the difference. Suppose we have cell C_i at bottom and cell C_j is on top of cell C_i. Conflicts may occur at discrete values of the position pairs. For simplicity, let the horizontal position of cell C_i be p_i. Possible situation is that conflicts occur if cell C_j is placed to position p_i+1, p_i+2, p_i+5, and p_i+6, while positions like p_i, p_i+3, p_i+4, and p_i+7 are safe for cell C_j. Therefore, the LUT for MOL should contain a set of conflict ranges instead of a single required distance like metal-1.

Nevertheless, as mentioned in Section II-A, such kind of approach may suffer from large problem sizes. Fig. 5 gives an example of solution spaces of four particular cells. The maximum amount of candidate coloring solutions for metal-1 comes from XOR2_X2, which is 12, but its MOL layer has more than 100 000 coloring solutions. It is no longer feasible to store all the solutions in the LUT. One way is to select partial solutions, but it will increase the difficulty to resolve conflicts for fewer coloring options.

Due to the regularity of MOL features, we can solve the problem more efficiently. All possible topological patterns of four abutting fingers can be represented by the patterns listed in Fig. 6 with proper flipping and rotation. Most of these patterns are TPL friendly except pattern F which results in a four-clique structure (K4). It is impossible to resolve a K4 with only three colors. As the topological patterns depend on

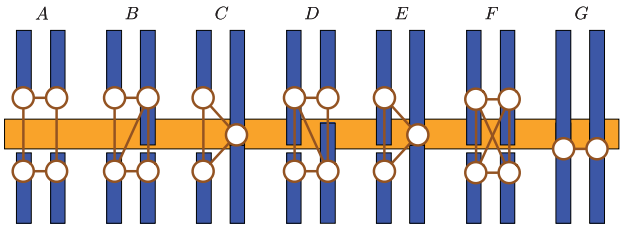

Fig. 6. Possible patterns of four abutting MOL fingers.

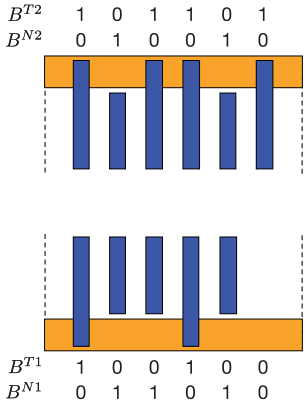

Fig. 7. Example of Boolean representation for MOL fingers.

cross-row cell positions, it has to be avoided during placement stage. If we can produce a K4-free placement solution for MOL layer, it should not be difficult to find a legal coloring solution by post placement color assignment due to the regularity. Actually we will show later that it is guaranteed to find a legal coloring solution if K4 is avoided. Hence, there is no need to explore the coloring solutions for MOL layer before placement.

The checking for K4 can be dynamically performed in placement instead of keeping an LUT. To further simplify the problem, we classify MOL fingers according to the vertical distances to cell boundary. Without loss of generality, we assume all touching fingers have the same distance to cell boundary, so do nontouching fingers. A BLUT is applied to represent the existence of different finger types. For each standard cell, four bitsets are required to store the fingers, as shown in Fig. 7. Bottom fingers are represented by B^{T1} and B^{N1} which denote the existence of touching fingers and nontouching fingers, respectively. If no finger exists at a certain grid j, both $B^{T1}(j)$ and $B^{N1}(j)$ are set to 0, where the grid is defined by tracks of gate layer. Top fingers are represented by B^{T2} and B^{N2}. The bottom and top concepts here are determined when a cell is orientated to *N*. Similar to cells, each row needs four bitsets R^{T1}, R^{N1}, R^{T2}, and R^{N2}. Note that all the four bitsets are necessary, because if there is neither touching nor nontouching finger at a grid, like that in the bottom right of Fig. 7, we need to set both corresponding touching and nontouching bitsets to zero at that grid.

The checking for K4 between two neighboring row i and row $i+1$ is summarized in Algorithm 1. As the orientation of a placement row is fixed, we assume row i is orientated to *N* and row $i+1$ is orientated to *FS*. So bitset R_i^{T2} and R_i^{N2} of row i interact with bitset R_{i+1}^{T2} and R_{i+1}^{N2} of row $i+1$.

Algorithm 1 Row K4 Checking

Require: Bitsets R_i^{T2}, R_i^{N2}, R_{i+1}^{T2}, R_{i+1}^{N2}.
Ensure: Whether there exists K4 between rows i and $i+1$.
1: $A_1 = R_i^{T2} \& R_{i+1}^{N2}$;
2: $A_2 = R_i^{N2} \& R_{i+1}^{T2}$;
3: $n =$ length of A_1;
4: **for** $j = 1$ *to* n **do**
5: **if** $A_1(j-1)A_1(j)$ or $A_2(j-1)A_2(j)$ **then**
6: **return** true;
7: **end if**
8: **end for**
9: **return** false;

The basic idea is to perform bitwise *and* operations for bitsets and check consecutive 1 s. Same approach can be applied to check K4 between a cell and its neighboring rows. Considering that the size of bitsets for a cell is smaller than that for a row, we can truncate the row bitsets by *shifting* before performing the checking.

It should be noted that the application of BLUT involves both precoloring and placement stages, since both standard cells and placement rows keep their own BLUTs. The BLUT for each cell is extracted in the precoloring stage according to the positions and types of MOL fingers, while during placement each row keeps a BLUT for all the cells it holds. The BLUT for each row is updated dynamically with cell movement.

IV. TPL Aware Detailed Placement

In this section, we present our scheme of TPL aware detailed placement to remove conflicts and minimize wirelength. To maintain the properties of input placement solutions such as wirelength and routability, the maximum displacement constraint is introduced to limit the amount of movement for each cell in Manhattan distance. For simplicity, if not specially mentioned, we use *flipping* to represent flipping a cell in horizontal direction.

A. Single Row Placement Problem

Previous studies have shown that single row placement has impressive performance in simultaneous color assignment, conflict removal, and wirelength minimization for metal-1 layer [22], [24], [26], [27]. In our scheme, the cost of single row placement problem comes from both horizontally and vertically abutting cells. Hence the information of neighboring rows should also be considered.

Problem 1 (Single Row Placement Problem): Given a row of standard cells and its neighboring rows, determine the position, orientation, and coloring for cells to minimize cost

$$\text{cost} = \Delta WL + \alpha \cdot N_{\text{ST}} + \beta \cdot N_{\text{CF}} \tag{1}$$

where N_{ST} denotes the number of stitches and N_{CF} stands for the number of conflicts.

When an algorithm for the single row placement problem is performed in a row, all cells on the other rows are fixed. In our implementation, parameter α is set to 1 and β is set to a very large value, e.g., 32 times of perimeter of the layout.

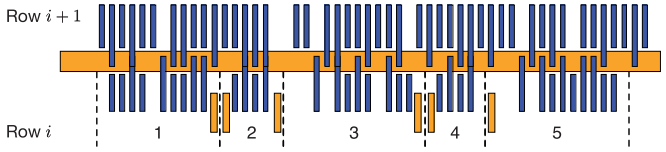

Fig. 8. Simplified example of single row placement problem.

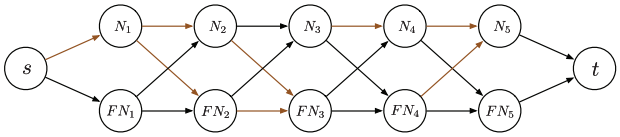

Fig. 9. Example of cell flipping graph.

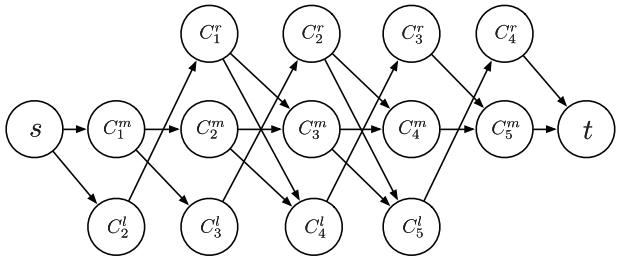

Fig. 10. Example of local cell reordering graph.

The cells in the neighboring rows are fixed when optimizing current row. Both metal-1 conflicts and MOL K4 structures are generalized to N_{CF}. An example for our single row placement problem is shown in Fig. 8. Blue rectangles stand for MOL fingers and orange rectangles represent boundary metal-1 features. Note that orientations of cells should correspond to their rows; i.e., if the row orientation is *N*, then cell orientation must be *N* or *FN*; if the row orientation is *FS*, then cell orientation must be *S* or *FS*. Although conflicts can be resolved by inserting whitespaces between abutting cells, flipping is also very effective due to the asymmetry of boundary features. For example, by flipping cell 1, both its vertical and horizontal conflicts are removed. Flipping is even more important when whitespaces are very limited in highly congested regions.

1) Ordered Single Row Placement: Single row placement problem with arbitrary order is well known $\mathcal{NP}$-hard [35]–[37]. However, optimal solutions can be found in polynomial time if the order of the cells is fixed [37]–[40]. Previous studies on TPL aware placement have already explored different algorithms for the application of intrarow conflicts [22]–[24], [26]. Here we adopt the linear dynamic programming algorithm in [23] as it is more compatible to maximum displacement constraints. The cost function for the algorithm is extended to check cross-row K4 which will be counted into conflicts, shown as (1), where N_{CF} includes the number of both metal-1 conflicts and cross-row K4.

2) Local Reordered Single Row Refinement: With only OSR, the performance is limited in terms of conflict removal owing to the high demands for whitespaces. If local reordering is available, larger solution space will contribute to fewer conflicts and stitches. In spite of the $\mathcal{NP}$-hardness of the arbitrary order single row placement, polynomial time algorithm is available for the problem with local reordering.

Inspired by Du and Wong [41], we construct a graph model to handle flipping, local reordering, local shifting, and metal-1 color assignment simultaneously. The difference of our method is that three vertices are introduced for each cell during local reordering to reduce number of edges. For simplicity, we explain the graph model for different techniques separately and then show the unified model. In the LRSR problem, the input has been optimized by OSR for wirelength, the main target is to further refine conflicts and stitches with small wirelength degradation. Therefore, displacement cost is applied instead of wirelength. The cost function can be rewritten as follows:

$$\text{cost} = D + \alpha \cdot N_{\mathrm{ST}} + \beta \cdot N_{\mathrm{CF}} \tag{2}$$

where *D* denotes the total movement of cells.

Consider the placement row shown in Fig. 8, whose the orientation is *N*. As only flipping is allowed within a row, a cell can be orientated to *N* or *FN*. Then for each cell, two vertices are introduced to represent its orientation, as Fig. 9 shows. For instance, vertex N_1 represents that cell 1 has an orientation of *N*, while vertex FN_1 denotes it is flipped to *FN*. If a flipping solution for two abutting cells results in metal-1 conflicts, the edge is assigned to a very large cost, such as the edge between N_1 and N_2. If a flipping solution of one cell leads to cross-row K4, all its input edges are assigned to a large cost like the edge connects *s* to N_1. These edges are marked in red. The problem of computing the best flipping solution is equivalent to finding the shortest path from *s* to *t* in the graph. In Fig. 9, we can find a legal path $s \to FN_1 \to N_2 \to N_3 \to FN_4 \to FN_5 \to t$.

The local reordering technique enables local swap of neighboring cells within a row. It can be stated by the following definition.

Definition 1 (P-Reordering): Given a row of standard cells, select *p* consecutive cells to switch positions between them, such that the cost defined in (2) is minimized.

Fig. 10 gives an example of the reordering graph when *p* is equal to 2. Each cell *i* corresponds to three vertices C_i^l, C_i^m, and C_i^r, as it can swap with its preceding cell or its succeeding cell. In Fig. 10, the edge from C_1^m to C_2^m stands for the condition that cells 1 and 2 are kept in original order. The edge from C_2^l to C_1^r represents the swap of cells 1 and 2. In this problem, if cell *i* has swapped with cell $i+1$, it is not allowed to further swap with cell $i+2$.

We can summarize the graph construction of two-reordering problem for a row with *N* cells as follows.

1) For each cell *i* ($2 \le i \le N$), vertex C_i^l has an output edge to vertex C_{i-1}^r.
2) For each cell *i* ($1 \le i \le N-1$), vertex C_i^m has an output edge to vertex C_{i+1}^m.
3) For each cell *i* ($1 \le i \le N-2$), vertex C_i^m has an output edge to vertex C_{i+2}^l.
4) For each cell *i* ($1 \le i \le N-2$), vertex C_i^r has an output edge to vertex C_{i+2}^m.
5) For each cell *i* ($1 \le i \le N-3$), vertex C_i^r has an output edge to vertex C_{i+3}^l.

As additional source vertex *s* and target vertex *t* are introduced, extra edges $s \to C_1^m$, $s \to C_2^l$, $C_{N-1}^r \to t$, and $C_N^m \to t$ are inserted.

For further flexibility, small movement at the original position and swapped positions for a cell is allowed. Let p_i^m be the original position of cell i, p_i^l denote the position if cell i is swapped with cell $i-1$, and p_i^r denote the position swapped with cell $i+1$. Additional vertices are inserted to enable positions such as $[p_i^l-d, p_i^l+d]$, $[p_i^m-d, p_i^m+d]$, and $[p_i^r-d, p_i^r+d]$, where d is the maximum shifting value for this step. At the same time, the graph model can also determine the coloring solutions for metal-1 layer.

The full algorithm uses a unified graph model that combines all the techniques mentioned above. In the unified graph model, each vertex has four attributes. Swapping attribute contains the swapping status (C_i^l, C_i^m, or C_i^r) of current cell. Shifting attribute d_i represents the shifting amount from current or swapped position. Color attribute k_i represents the metal-1 coloring solution and flipping attribute f_i represents the flipping status of a cell. The interconnection of the unified graph can be derived from previous cell flipping graph and local reordering graph. Edge cost is determined by all the attributes of two vertices. Unified graph model is used in the implementation to solve the LRSR problem optimally.

Let N be the total number of cells, d be the maximum shifting value for LRSR and K be the maximum number of coloring solutions. Since three types of vertices are introduced for local reordering and there are $2d+1$ candidate positions in local shifting for each vertex, the total number of vertices in the graph is $3N \cdot (2d+1) \cdot K \cdot 2$. As the graph model turns out to be a directed acyclic graph, the shortest path problem can be solved with $\mathcal{O}(d^2K^2N)$ time complexity. In the experiment we set d to 1, so the time complexity is $\mathcal{O}(K^2N)$. It should be noted that although not explicitly shown in Figs. 8 and 10, the algorithm is not limited by any whitespace between cells. Because in the graph model shifting values are enumerated by vertices and we always know the vertices from the neighboring cells of each target cell. If any overlap occurs between any pair of vertices, no edge is inserted between them.

B. Multiple Row Conflict Removal Based on Min-Cost Flow

While it is true that single row placement can remove most conflicts, it tends to fail in very dense regions. Movement between multiple rows is necessary to resolve all the conflicts. Conventional approaches such as global move are able to iteratively move cells out of congested regions. But due to the greedy nature, the solution may not be good from a global view. When cells compete for whitespaces, greedy-based method may process the less "urgent" cells before the most urgent one. For example, cell i and cell j share one candidate whitespace; cell i has another candidate whitespace, while cell j does not. The desired procedure is to place cell j to its only whitespace and move cell i to the remaining candidate position. It is very difficult for greedy approaches to handle such kind of situations.

Problem 2 (Conflict Removal Problem): Given a set of cells with conflicts, move these cells to eliminate conflicts with minimum degradation of the solution quality, i.e., minimum displacement to previous placement solution.

This step aims at eliminating the metal-1 conflicts and MOL K4 during post single row placement stage. We need to move the conflict cells to whitespaces with minimum conflicts and displacement. To solve this problem, a min-cost flow-based algorithm is proposed to add concurrent characteristics by assigning conflict cells to whitespaces simultaneously. Fig. 11 shows an example of the constructed graph for min-cost flow algorithm. Vertex C_i indicates cells and W_j denotes whitespaces. Edges are marked with (cost, capacity) pairs. In current implementation, the capacity is set to 1, which means we only support assigning one cell to one whitespace, because it is difficult to guarantee free of conflicts within one whitespace if multiple cells are assigned. The basic procedure of this algorithm is summarized in Algorithm 2.

When selecting candidate cells, we detect *conflict chains* and group related cells; A *conflict chain* is a set of cells that form a connected component if we connect them with conflict edges. For example, if there is conflict between cells 1 and 2, and also conflict between cells 2 and 3, then these cells are grouped together, called a conflict chain. For a conflict chain with n cells, only smallest $n-1$ cells will be selected for movement, because it is easier for small cells to fit into whitespaces. The whitespaces are collected in the following way.

Algorithm 2 Min-Cost Flow-Based Conflict Removal

Require: A set of cells $C_1, C_2, ..., C_n$ and a set of whitespaces $W_1, W_2, ..., W_m$.
Ensure: Assign cells to whitespaces with minimum costs
1: Detect conflicts and select candidate cells;
2: Collect whitespaces near candidate cells;
3: Construct graph G and candidate solution map S:
4: Add source vertex s and target vertex t to G;
5: Add edge $s \rightarrow C_i$ and edge $W_j \rightarrow t$ with cost 0 and capacity 1 for all i and j;
6: Add edge $C_i \rightarrow W_j$ iff C_i can be placed to W_j legally:
7: Calculate the best cost b with Eq. (2) when C_i is placed to W_j by enumerating all combinations of position, flipping and coloring solutions;
8: Assign b to the edge cost and 1 to edge capacity;
9: Add the corresponding solution of b to S;
10: Solve min-cost flow for G;
11: Assign cell C_i to whitespace W_j if the edge $C_i \rightarrow W_j$ has nonzero flow and apply corresponding flipping and coloring solutions stored in S;

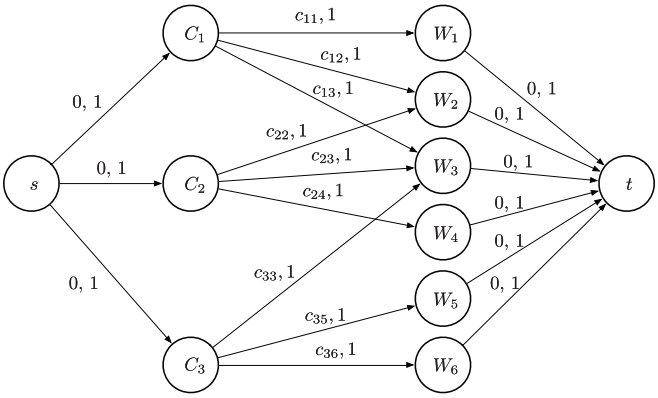

Fig. 11. Example of min-cost flow-based whitespace assignment.

For each candidate cell at row i, scan from rows $i-3$ to $i+3$ and collect all whitespaces without any repeated one. For any pair of whitespaces, if the distance between them is smaller than coloring distance, only the larger one is kept to avoid potential conflicts after cell assignment. Once the min-cost flow is solved, we iterate through the edges from all C_i to W_j to check the amount of flow; if the flow on the edge is 1, C_i is assigned to W_j with the position, flipping and coloring solution the same as that computed in the cost c_{ij}.

C. Stitch and Wirelength Refinement With Interleaving

With previous approaches to remove conflicts in Sections IV-A and IV-B, it is likely to result in wirelength and stitch degradation since they try to resolve conflicts while minimizing displacement. Inspired by Mongrel [42], we try to refine wirelength and stitch by interleaving a sequence of cells within a row. The interleaving problem defined by Mongrel only allows two partitions. We extend it to general q-partition interleaving that allows any arbitrary number of partitions q. Whitespaces are also integrated as dummy cells for interleaving such that cells are able to shift around.

Definition 2 (Q-Partition Interleaving): Given a sequence of standard cells and whitespaces in a row, extract q subsequences (partitions) with the relative order of cells in each subsequence fixed and interleave the subsequences to minimize the cost defined in (1).

Fig. 12 gives an example of two-partition interleaving and three-partition interleaving. Although the subsequence in each partition can be randomly picked, we classify consecutive cells into different partitions periodically; i.e., cell i goes to partition $(i-1 \mod q)+1$, where i starts from 1. A whitespace can be split into several small pieces and incorporated into interleaving as dummy cells. The subsequence of cells in each partition must preserve their relative order, while there is no ordering requirement for cells from different partitions. It can be seen that with larger q, cells have more freedom to move around.

Let $S_{i_1,i_2,\ldots,i_q}$ denote the optimal arrangement consisting of i_1 cells from partition 1, i_2 cells from partition 2, $\ldots$, i_q cells from partition q. Let c_j^i denote the jth cell in the subsequence of partition i. Let $C(S_{i_1,i_2,\ldots,i_q})$ denote the cost of partial placement defined by $S_{i_1,i_2,\ldots,i_q}$. The wirelength cost in $C(S_{i_1,i_2,\ldots,i_q})$ only computes the total horizontal wirelength up to the right boundary of $S_{i_1,i_2,\ldots,i_q}$. The interleaving problem in Mongrel is a special case of two-partition interleaving problem with only wirelength cost, which can be solved by dynamic programming with following recurrence relation:

$$S_{0,0} = \emptyset \tag{3a}$$

$$C\left(S_{0,0}\right) = 0 \tag{3b}$$

$$S_{i,j} = \begin{cases} S_{i-1,j}c_i^1, & \text{if } C\left(S_{i-1,j}c_i^1\right) < C\left(S_{i,j-1}c_j^2\right) \\ S_{i,j-1}c_j^2, & \text{otherwise.} \end{cases} \tag{3c}$$

The process of solving (3) is equivalent to filling up the $((N/2)+1)\times((N/2)+1)$ entries for the dynamic programming table in Fig. 13 from top left corner to bottom right, where N is the total number of cells; i.e., $m \times n$ entries if we use the notations in Mongrel. Starting from empty set, the cells in different partitions are gradually inserted to the partial placement

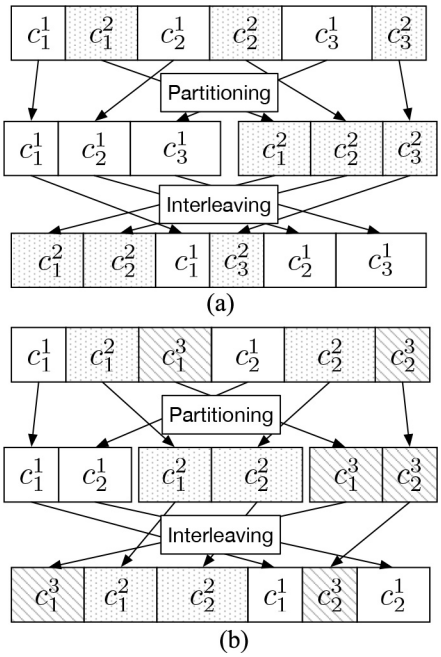

Fig. 12. Example of (a) two-partition interleaving and (b) three-partition interleaving.

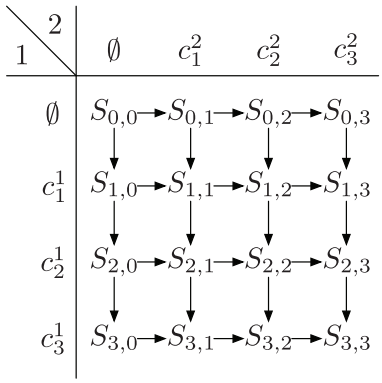

Fig. 13. Example of dynamic programming table for two-partition interleaving with wirelength cost only.

set of S. However, the q-partition interleaving problem tries to minimize the cost function in (1), which consists of wirelength, stitch and conflict costs, where the conflict cost comes from metal-1 and MOL K4 structure. Wirelength cost has already been discussed in Mongrel. Costs from stitch and MOL K4 structure do not involve interaction between cells; in other words, they only depend on the cell itself. metal-1 conflict cost has to be computed from two horizontally neighboring cells, which breaks the independency. Equation (3) cannot be applied to solve even the two-partition interleaving problem, because the metal-1 coloring solution of rightmost cell in $S_{i,j}$, i.e., cell c_i^1 or c_j^2, has interaction with the next cell c_{i+1}^1 or c_{j+1}^2 for computing $S_{i+1,j}$ or $S_{i,j+1}$. As a result, we are not able to prune the partial placement of either $S_{i-1,j}c_i^1$ or $S_{i,j-1}c_j^2$ for $S_{i,j}$.

In order to handle conflict costs, a high-dimensional dynamic programming table is necessary to record partial solutions from different coloring solutions. Considering the fact

that metal-1 conflict only involves interactions between current cell and previous cell, for any $S_{i,j}$ in a 2-D table for two-partition interleaving, we introduce two more dimensions to denote the partition of current rightmost cell and its metal-1 coloring solution. We use $S_{i,j,p,k}$ to represent the partial placement, where $p = 1, 2$ for the rightmost cell from either of the two partitions, and $k = 1, 2, \ldots, K$ for K candidate coloring solutions for metal-1. Let $c^p_{i,k}$ denote that the ith cell in partition p is assigned to kth candidate coloring solution. Then $S_{i,j,1,k}$ can be derived from $S_{i-1,j,1,k_1}c^1_{i,k}$ or $S_{i-1,j,2,k_2}c^1_{i,k}$, where k_1 and k_2 indicate the coloring solutions of the rightmost cells in the previous partial placement. Similarly, $S_{i,j,2,k}$ can be derived from $S_{i,j-1,1,k_1}c^2_{j,k}$ or $S_{i,j-1,2,k_2}c^2_{j,k}$.

With the notations above, the new recurrence relation for two-partition interleaving can be written as

$$S_{0,0,p,k} = \emptyset \tag{4a}$$
$$C\big(S_{0,0,p,k}\big) = 0 \tag{4b}$$
$$S_{i,j,1,k} = \underset{k_1,k_2}{\operatorname{argmin}}\Big\{C\Big(S_{i-1,j,1,k_1}c^1_{i,k}\Big) C\Big(S_{i-1,j,2,k_2}c^1_{i,k}\Big)\Big\} \tag{4c}$$
$$S_{i,j,2,k} = \underset{k_1,k_2}{\operatorname{argmin}}\Big\{C\Big(S_{i,j-1,1,k_1}c^2_{j,k}\Big) C\Big(S_{i,j-1,2,k_2}c^2_{j,k}\Big)\Big\} \tag{4d}$$
$$p \in \{1, 2\} \tag{4e}$$
$$k, k_1, k_2 \in \{1, 2, \ldots, K\} \tag{4f}$$
$$i, j \in \left\{1, 2, \ldots, \frac{N}{2}\right\} \tag{4g}$$

where N is the total number of cells in the sequence including whitespaces as dummy cells, K is the total number of candidate coloring solutions for each cell. In (4c), partial placement $S_{i,j,1,k}$ must be the one in the set $\{S_{i-1,j,1,k_1}c^1_{i,k}, S_{i-1,j,2,k_2}c^1_{i,k}\}$ that results in minimum cost by function C, similar is for $S_{i,j,2,k}$. If cell $c^1_{i,k}$ is a standard cell, (4c) holds, since it isolates partial placement $S_{i-1,j,1,k_1}$ and $S_{i-1,j,2,k_2}$ to be independent problems. If cell $c^1_{i,k}$ is a dummy cell from a whitespace, we need to ensure it is wide enough to guarantee the independency of subproblems; i.e., the whitespace should be able to resolve metal-1 coloring conflicts between its left cell and right cell. To ensure each whitespace is wide enough such that no metal-1 conflict is possible between the cells at its left and right, we split that whitespace into pieces that can resolve conflicts between any pair of cells in the row. Similar argument holds for (4d). The high-dimensional table for two-partition interleaving consists of $((N/2)+1) \times ((N/2)+1) \times 2 \times K$ entries.

Now we generalize to q-partition interleaving from two-partition interleaving. We represent the vector space defined by $\{i_1, i_2, \ldots, i_q\}$ with $\vec{v}$. Two functions f and g are introduced for space transformation

$$f_i(j, p) = \begin{cases} j-1, & \text{if } i = p \\ j, & \text{otherwise} \end{cases} \tag{5a}$$
$$g(\vec{v}, p) = \big\{f_1(i_1, p), f_2(i_2, p), \ldots, f_q\big(i_q, p\big)\big\}. \tag{5b}$$

Functions f and g help find the previous vector space from $\vec{v}$ and p. If $p = 1$, then $g(\vec{v}, p) = \{i_1 - 1, i_2, \ldots, i_q\}$; if $p = 2$, then $g(\vec{v}, p) = \{i_1, i_2 - 1, \ldots, i_q\}$, and so forth. The assignment operation $\vec{w} \leftarrow g(\vec{v}, p)$ is equivalent to

$$\vec{w} \leftarrow \vec{v}$$
$$\vec{w}(p) \leftarrow \vec{w}(p) - 1 \tag{6}$$

where $\vec{w}(p)$ indicates the pth dimension of $\vec{w}$. The recurrence relation for q-partition interleaving can be written as

$$S_{\vec{v},p,k} = \emptyset \tag{7a}$$
$$C\big(S_{\vec{v},p,k}\big) = 0 \tag{7b}$$
$$S_{\vec{v},p,k} = \underset{p',k'}{\operatorname{argmin}}\Big\{C\Big(S_{g(\vec{v},p),p',k'}c^p_{i_p,k}\Big)\Big\} \tag{7c}$$
$$p, p' \in \{1, 2, \ldots, q\} \tag{7d}$$
$$k' \in \{1, 2, \ldots, K\} \tag{7e}$$
$$i_1, i_2, \ldots, i_q \in \left\{1, 2, \ldots, \frac{N}{q}\right\} \tag{7f}$$

where we assume (N/q) is an integer for simplicity. Eventually $S_{\vec{v},p,k}$ has $q+2$ dimensions and there are $((N/q)+1)^q \times q \times K$ entries in the dynamic programming table.

The details about the algorithm are shown in Algorithm 3. The dynamic programming table is filled by calling the recursive function `SolveDP` from lines 13 to 31. To avoid repeating computation, each entry in the table has a variable to record whether it is visited, which is shown from lines 14 to 17. When the recursion reaches to $\vec{v}' = \vec{0}$, cell $c^p_{\vec{v}(p),k}$ is inserted to $S_{\vec{v},p,k}$ as the first cell from lines 19 to 21; otherwise, $S_{\vec{v},p,k}$ is constructed from the best one among all the combinations of $S_{\vec{v}',p',k'}c^p_{\vec{v}(p),k}$ from lines 22 to 29. The best solution S^* has to be selected from $q \times K$ last entries where $\vec{v} = \{(N/q), (N/q), \ldots, (N/q)\}$, which corresponds to lines 4 to 10.

The runtime complexity is highly related to the total number of entries in the dynamic programming table. A straightforward analysis is assuming the computation of each entry takes $\mathcal{O}(q \times K)$ runtime complexity because each entry has to iterate through all the $q \times K$ combinations of previous entries with current cell. So the overall runtime complexity can be $\mathcal{O}(((N/q))^q \times q^2 \times K^2)$ considering all the entries in the dynamic programming table. In real implementation, it is more efficient to store indices or pointer of previous entry and current cell inserted in each entry such that the best solution can be obtained from backtracking from the best last entry. Further speedup can be achieved by avoiding computing some parts of cost repeatedly; e.g., $S_{\vec{v}',p',k'}c^p_{\vec{v}(p),k}$ and $S_{\vec{v}',p',k'}c^p_{\vec{v}(p),k+1}$ should have the same cost on wirelength and MOL. As the runtime increases exponentially with the number of cells, we apply the algorithm to small sliding windows within each row.

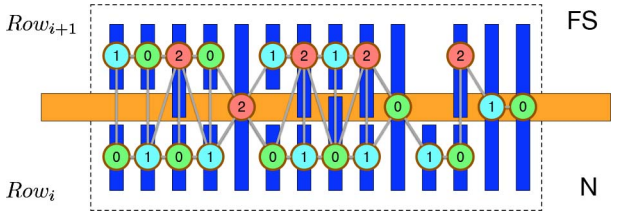

Fig. 14. Example of MOL coloring solutions.

Algorithm 3 Q-Partition Interleaving

Require: A sequence of N cells and number of partitions q.
Ensure: Interleave cells to minimize wirelength, conflict and stitch cost.

```
Partition cells into q subsequences;
Define S* as the placement with best cost, C(S*) ← ∞;
C(S_{0⃗,p,k}) ← 0, ∀p, k;
v⃗ ← {N/q, N/q, ..., N/q};
for each p ∈ 1, 2, ..., q, k ∈ 1, 2, ..., K do
    S_{v⃗,p,k} ← SolveDP(S_{v⃗,p,k});
    if C(S_{v⃗,p,k}) < C(S*) then
        S* ← S_{v⃗,p,k};
    end if
end for
return S*;

function SolveDP(S_{v⃗,p,k})
    if S_{v⃗,p,k} is visited then
        return S_{v⃗,p,k};
    end if
    Set S_{v⃗,p,k} as visited;
    v⃗' ← g(v⃗, p);
    if v⃗' = 0⃗ then
        S_{v⃗,p,k} ← {c^p_{1,k}};
    end if
    for each p' ∈ 1, 2, ..., q, k' ∈ 1, 2, ..., K do
        if v⃗'(p') > 0 then
            S_{v⃗',p',k'} ← SolveDP(S_{v⃗',p',k'});
            if C(S_{v⃗',p',k'} c^p_{v⃗(p),k}) < C(S_{v⃗,p,k}) then
                S_{v⃗,p,k} ← S_{v⃗',p',k'} c^p_{v⃗(p),k};
            end if
        end if
    end for
    return S_{v⃗,p,k};
end function
```

D. MOL Layer Color Assignment

After TPL aware placement, we obtain a placement solution without metal-1 conflicts or MOL K4. We still need to assign color to MOL layer. Due to the regularity of MOL features, we can decompose the MOL layer in a row-by-row approach. Layout decomposition for row-based layout structure has been well studied by Tian *et al.* [9] and Chien *et al.* [14]. The region we need to perform coloring algorithm is not the placement row, but the interaction region between two neighboring rows, shown in the dashed region of Fig. 14 where we consider the top of row i and bottom of row $i+1$ together in the example. According to the analysis for different types of patterns in Fig. 6, we scan the MOL fingers from left to right, construct a pattern with previous finger pair (left) and current finger pair, and perform pattern matching to find the coloring solution. For instance, if a pattern matches pattern B in Fig. 6, the bottom right finger should be assigned with the same color as the top left finger; then the top right finger should be assigned to last available color. The MOL finger color assignment can be solved in linear time without conflicts as long as no K4 exists.

TABLE I
BENCHMARK STATISTICS

Bench	cell#	net#	d_{max}	$K_{Metal-1}$
alu-70/80/85/90	2125	2524	5	12
byp-70/80/85/90	9116	10075	5	18
div-70/80/85/90	6050	6390	5	18
ecc-70/80/85/90	1286	1488	5	18
efc-70/80/85/90	1956	1964	5	48
ctl-70/80/85/90	2350	2578	5	48
top-70/80/85/90	21251	21608	5	48

V. EXPERIMENTAL RESULTS

A. Experimental Setup

Our algorithms are implemented in C++ and tested on an eight-core 3.40 GHz Linux server with 32 GB RAM. The min-cost flow problem is solved by the successive shortest path algorithm in Boost library [44]. We use Nangate 15-nm library [45] as our initial standard cell library. Due to different standard cell libraries, we cannot directly use the benchmark from [23]. Therefore, we synthesize new placement solutions for OpenSPARC T1 designs using the Nangate 15-nm library and Cadence Encounter [46]. For each design, we choose four different utilization rates, 0.7, 0.8, 0.85, and 0.9. Usually, the higher utilization rate, the harder to find legal placement and coloring solutions. Table I lists all the statistics of different test cases. Columns "cell#" and "net#" are the total number of cells and the total number of nets in each test case, respectively. Column "$d_{\max}$" is the maximum displacement constraint for each placement. Column "$K_{\text{metal-1}}$" is the maximum number of cell precoloring solutions among all standard cell types for metal-1 layer. The value of $K_{\text{metal-1}}$ is related to the LUT size. Related benchmarks and executables are released at link (http://www.cerc.utexas.edu/utda/download/TPLPlace/index.html).

During standard cell precoloring, the coloring distance for both metal-1 and MOL layers is set to 80. For metal-1 layer the upper bound of coloring solutions is set to 50, while for MOL layer the upper bound is set to 10. During standard cell placement, since Nangate 15-nm library inserts dummy poly on the cell boundaries, there is no metal-1 conflict within a row. Although such design can effectively provide legal single row coloring solution, the inserted dummy pitch would cause additional area penalty, as shown in Fig. 1(a). To better compare the performances of different placement techniques, we increase the coloring distance on metal-1 for neighboring cells to 110.

B. Detailed Placement Algorithm Comparison

Table II analyzes the performances of the proposed BLUT construction (see Section III-B) and different detailed placement algorithms (see Section IV). Here we compare four different design methodologies, as listed in four columns. Column "LUT+OSR" is based on conventional LUT construction and OSR placement. Here we implement the linear dynamic programming algorithm in [23] to search for optimal single row placement solution. Column "BLUT+OSR" is extended from LUT+OSR that instead of conventional LUT construction, the BLUT introduced in Section III-B is applied

TABLE II
PERFORMANCE EVALUATION OF LUT CONSTRUCTION AND DETAILED PLACEMENT ALGORITHMS

Bench	LUT+OSR [43]				BLUT+OSR [43]				BLUT+All [43]				Full Flow			
	ΔWL	CF	ST	Time(s)	ΔWL	CF	ST	Time(s)	ΔWL	CF	ST	Time(s)	ΔWL	CF	ST	Time(s)
alu-70	-1.53%	1778	75	1007.65	-3.71%	2	72	3.68	-3.66%	0	72	3.96	-4.14%	0	66	4.63
alu-80	-1.46%	2277	83	895.27	-2.91%	11	86	3.44	-2.36%	0	87	4.48	-3.35%	0	73	4.65
alu-85	-0.86%	2516	85	828.55	-2.19%	20	83	3.19	-1.65%	0	89	4.32	-1.97%	0	70	4.92
alu-90	0.13%	2823	87	767.20	-0.69%	30	90	3.10	-0.09%	0	89	4.08	-0.48%	0	70	5.33
byp-70	-0.78%	6335	806	4453.00	-2.09%	0	791	16.41	-2.09%	0	791	16.62	-2.32%	0	721	17.58
byp-80	-0.39%	8485	852	4058.07	-1.34%	2	853	15.50	-1.32%	0	854	16.00	-1.48%	0	741	16.66
byp-85	-0.39%	9736	931	3971.18	-1.39%	10	918	15.45	-1.26%	0	918	16.96	-1.58%	0	784	17.85
byp-90	-0.14%	10965	1010	3676.96	-0.96%	29	992	14.61	-0.79%	0	1000	18.12	-1.06%	0	855	18.92
div-70	-1.89%	4857	589	2807.06	-3.47%	7	583	10.40	-3.37%	0	583	11.56	-3.78%	0	550	11.95
div-80	-1.00%	6237	620	2570.82	-2.52%	12	605	9.98	-2.35%	0	604	11.25	-2.68%	0	554	11.91
div-85	-1.06%	6879	635	2474.66	-2.34%	24	631	9.85	-1.95%	0	631	12.09	-2.51%	0	561	12.38
div-90	-0.40%	7638	686	2353.77	-1.66%	22	676	9.38	-1.46%	0	673	11.33	-1.89%	0	596	11.49
ecc-70	-2.51%	1078	253	626.37	-4.45%	0	248	2.46	-4.45%	0	248	2.46	-5.30%	0	240	2.50
ecc-80	-1.39%	1382	270	565.63	-3.37%	1	262	2.27	-3.27%	0	262	2.48	-4.03%	0	245	2.45
ecc-85	-1.35%	1508	278	509.79	-2.82%	2	272	2.09	-2.22%	0	272	2.33	-3.46%	0	250	2.01
ecc-90	-0.72%	1835	294	384.69	-1.33%	23	295	1.79	-0.27%	0	303	2.25	-1.35%	0	270	2.48
efc-70	-3.96%	1641	235	1048.91	-6.51%	0	239	3.86	-6.51%	0	239	3.84	-7.29%	0	228	3.82
efc-80	-3.16%	2119	253	944.11	-4.98%	0	246	3.82	-4.98%	0	246	3.84	-5.93%	0	228	3.52
efc-85	-1.57%	2349	258	840.85	-3.47%	3	253	3.63	-3.32%	0	253	3.92	-4.32%	0	232	3.56
efc-90	0.09%	2534	261	832.88	-1.51%	23	264	3.68	-0.79%	0	267	4.64	-1.92%	0	245	4.35
ctl-70	-3.22%	2158	437	1243.38	-4.57%	1	439	5.04	-4.50%	0	439	5.08	-5.16%	0	428	4.74
ctl-80	-2.15%	2535	444	1139.64	-3.52%	2	447	4.82	-3.42%	0	447	5.11	-4.13%	0	431	4.82
ctl-85	-1.83%	2870	452	1046.36	-2.83%	10	458	4.60	-2.67%	0	458	5.32	-3.37%	0	439	4.78
ctl-90	-0.87%	3196	474	978.42	-1.68%	13	475	4.40	-1.34%	0	478	5.52	-1.91%	0	451	4.72
top-70	-0.94%	14435	1157	12402.91	-2.26%	8	1126	38.29	-2.23%	0	1125	41.17	-2.58%	0	1090	42.43
top-80	-0.49%	19295	1201	11269.44	-1.63%	6	1181	36.40	-1.58%	0	1185	39.08	-1.86%	0	1105	38.42
top-85	-0.17%	21870	1212	11057.75	-1.33%	21	1213	35.53	-1.08%	0	1213	40.13	-1.51%	0	1114	39.63
top-90	0.20%	24634	1257	10382.27	-0.83%	23	1270	34.40	-0.73%	0	1272	40.86	-0.99%	0	1178	35.86
avg.	-1.21%	**6284**	542	3040.63	-2.58%	**10**	538	10.79	-2.35%	**0**	539	12.10	-2.94%	**0**	493	12.08
ratio	-0.51	-	1.01	251.29	-1.08	-	1.00	0.89	-1.00	-	1.00	1.00	-1.25	-	0.91	1.00

here. Column "BLUT+All" employs BLUT and all the TPL aware detailed placement techniques in [43], including all techniques in Section IV except Section IV-C. Column "Full Flow" employs BLUT and all the TPL aware detailed placement techniques proposed in Section IV. For each design methodology, we list four different metrics: "ΔWL," "CF," "ST," and "Time(s)." ΔWL measures the total wirelength change after optimization. CF gives the total conflict number on metal-1 and MOL layer. ST gives the stitch number only on metal-1 layer as stitching is not allowed on MOL layer. Time(s) lists the process runtime in seconds.

We first compare methodologies LUT+OSR and BLUT+OSR, where both of them utilize the OSR placement in [23]. The difference is that the former one employs the conventional LUT, while the latter one uses the BLUT as in Section III-B. We can see that, on average LUT+OSR introduces more than 5000 conflicts, while BLUT+OSR only reports less than ten conflicts. The reason is that under MOL-based structure, conventional LUT may contain too many precoloring combinations. To control the LUT size, some precoloring solutions are predeleted in cell library. Therefore, LUT-based method is not able to fix most of the conflicts. Compared with conventional LUT, the proposed BLUT enables larger solution space, and thus is more powerful to fix cross-row conflicts.

We further compare BLUT+OSR and BLUT+All, where BLUT+All integrates OSR, LRSR, and min-cost flow techniques for detailed placement developed in this paper. From Table II, we can see that both BLUT+OSR and BLUT+All can achieve around 1% improvement in wirelength compared with LUT+OSR. Compared with BLUT+OSR, through some powerful techniques, such as local reordered single row refinement (LRSR) and min-cost flow-based conflict removal, BLUT+All can resolve all the TPL conflicts. It shall be noted that the runtime penalty of BLUT+All is not significant, i.e., BLUT+All only introduces less than 11% of runtime overhead against BLUT+OSR. The reason is that for each single row, OSR-based method is applied first. Only if there exist remaining conflicts, more expensive LRSR and min-cost flow-based methods are applied.

The Full Flow includes all the techniques in BLUT+All and the interleaving algorithm for post refinement which further optimizes wirelength and stitches without creating any conflict. Compared with BLUT+All, it produces 9% fewer stitches and further increases the average wirelength improvement from 2.35% to 2.94% with similar runtime. The nature of the interleaving algorithm for post refinement is a single row placement algorithm, which can also serve as conflict removal like OSR placement. However, we observe that it is not as effective as the OSR placement, since it subjects to strict constraints from partitions which limit its solution space. Hence we apply it in the post refinement stage where OSR placement has already been applied.

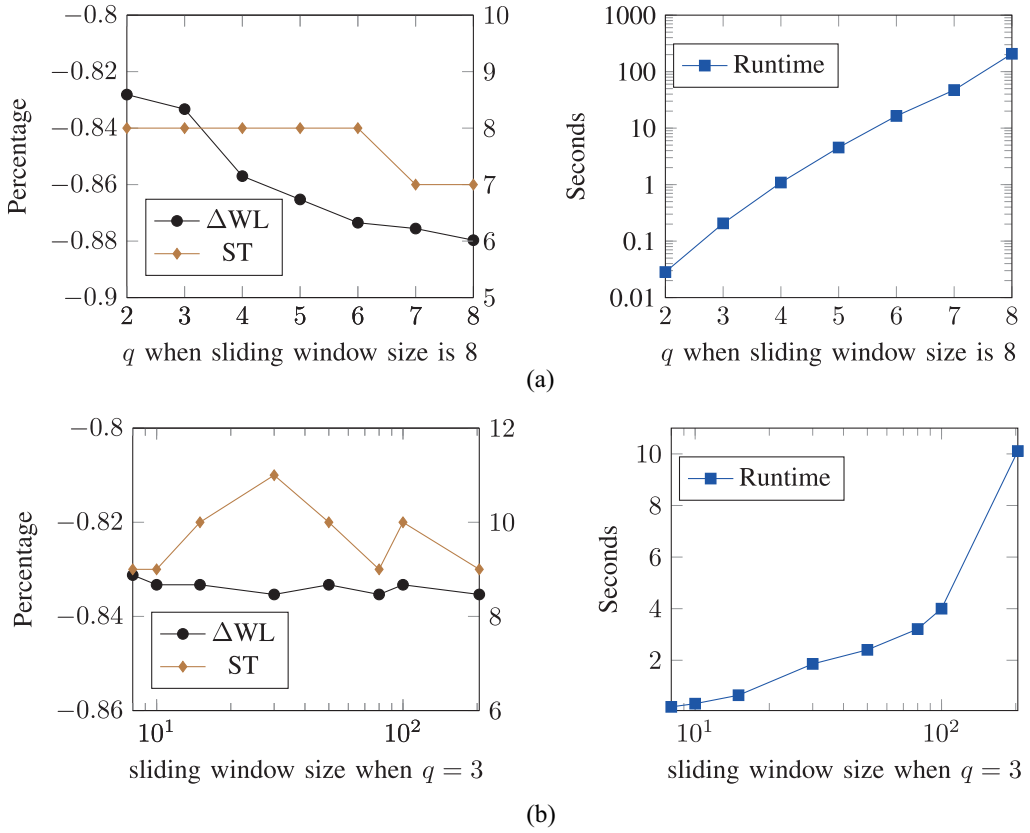

Fig. 15. Example of wirelength, stitch, and runtime tradeoffs for q-partition interleaving. Sweeping over (a) q when sliding window size is set to 8 and (b) sliding window size when $q = 3$.

C. Tradeoffs in Wirelength and Stitch Refinement

The performance and runtime of q-partition interleaving in Section IV-C are highly related to the number of partitions q and size of sliding window. We study the relation by sweeping q and sliding window sizes for one randomly picked row in benchmark "top-85," as shown in Fig. 15.

The size of sliding windows is set to 8 when sweeping over q, which means eight standard cells will be included in a sliding window regardless of whitespaces among them. The overall number of cells for interleaving including dummy cells from whitespaces is usually 2–3 times of the total standard cells in the window, as we split whitespaces into small pieces for interleaving. With the increase of q, cells have more freedom to move around and thus result in better cost including wirelength and stitches, as shown in Fig. 15(a). It is easier for wirelength to drop significantly when cells are able to have large movements, while the number of stitches reduces much slower. Sometimes, the stitch number may even increase a little bit to achieve better wirelength. According to the analysis of runtime complexity in Section IV-C, the runtime grows exponentially with q, which is also shown in the figure.

Fig. 15(b) shows the impact of sliding window sizes on wirelength and stitches. The smallest sliding window size in the samples is 8, and the largest size is the total number of standard cells in the row, i.e., 204. When the window size is set to largest, the interleaving algorithm gives the optimal cost in the row; otherwise, it achieves suboptimal solutions as we divide the problem. It can be seen that the gaps of wirelength and stitches between the suboptimal and optimal solutions are not very large, but the runtime increases quickly. To balance runtime and performance, we set the q to 3 and the size of sliding window to 8.

Another factor to affect runtime is the maximum number of pieces that each whitespace is split into. We split a whitespace into pieces such that each piece is wide enough to resolve potential metal-1 conflicts between standard cells in the row. However, whitespace pieces are regarded as dummy cells in interleaving, which is directly correlated to the problem size. It may result in runtime overhead if too many whitespace pieces are generated, especially in layout with low utilization. Thus we set an upper bound of pieces that can be generated from each whitespace. Fig. 16 shows the relation between runtime and maximum number of pieces for a randomly picked row for benchmark "top" with various utilizations. The runtime increases with the upper bounds and saturates at different saturating points where the saturating points of layouts with low utilizations come later than that with high utilizations. As a result, layouts with low utilizations may be slower than others with high utilizations, though they contain similar number of standard cells. The phenomenon is still quite significant in the runtime of "top-70" to "top-90" in Table II. We control the total number of pieces generated in each window to be no larger than twice the amount of standard cells for better consistency of runtime. The influences to performances from the upper bound vary from design to design in our experiment.

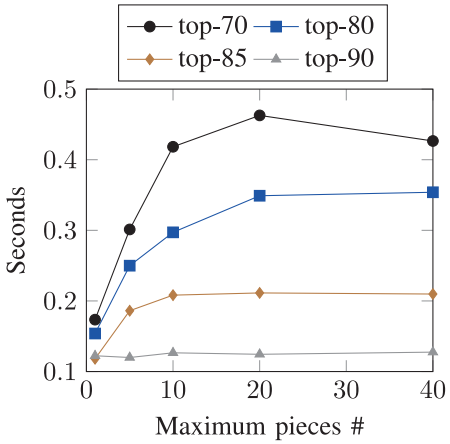

Fig. 16. Sweeping over maximum number of pieces for each whitespace when $q = 3$ and sliding window size is 8.

VI. Conclusion

Motivated by a large amount of cross-row TPL conflicts from MOL layers, in this paper we have proposed a comprehensive study to TPL aware detailed placement toward zero cross-row conflict. Several effective techniques, such as BLUT construction, LRSR, min-cost flow-based conflict removal, and q-partition interleaving for post refinement, are proposed. Our framework is verified through a set of placement test cases using 15-nm standard cell library.

In the future, we will try to further refine the techniques for conflict removal. For example, the min-cost flow-based conflict removal algorithm only supports assigning one cell to one whitespace, which is worthwhile for further exploration of assigning multiple cells into one whitespace. With further scaling of transistor feature size and MOL layers, the cross-row conflicts would be an emerging problem for TPL friendly design. We believe this paper will stimulate more future research on TPL aware standard cell design and physical design.

Acknowledgment

The authors would like to thank Dr. Y.-C. Ban at GLOBALFOUNDRIES for helpful comments.

References

[1] L. Liebmann, A. Chu, and P. Gutwin, "The daunting complexity of scaling to 7NM without EUV: Pushing DTCO to the extreme," in *Proc. SPIE*, vol. 9427. San Jose, CA, USA, 2015, Art. no. 942702.

[2] B. J. Lin, "Optical lithography with and without NGL for single-digit nanometer nodes," in *Proc. SPIE*, vol. 9426. San Jose, CA, USA, 2015, Art. no. 942602.

[3] 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.

[4] D. Z. Pan, L. Liebmann, B. Yu, X. Xu, and Y. Lin, "Pushing multiple patterning in sub-10nm: Are we ready?" in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.

[5] B. Yu *et al.*, "Design for manufacturability and reliability in extreme-scaling VLSI," *Sci. China Inf. Sci.*, vol. 59, no. 6, pp. 1–23, 2016.

[6] L. Liebmann, "Demonstrating production quality multiple exposure patterning aware routing for the 10nm node," in *Proc. SPIE*, vol. 9053. San Jose, CA, USA, 2014, Art. no. 905309.

[7] K. Lucas *et al.*, "Implications of triple patterning for 14nm node design and patterning," in *Proc. SPIE*, vol. 8327. San Jose, CA, USA, 2012, Art. no. 832703.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

[14] H.-A. Chien, S.-Y. Han, Y.-H. Chen, and T.-C. Wang, "A cell-based row-structure layout decomposer for triple patterning lithography," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Monterey, CA, USA, 2015, pp. 67–74.

[15] Y. Lin, X. Xu, B. Yu, R. Baldick, and D. Z. Pan, "Triple/quadruple patterning layout decomposition via novel linear programming and iterative rounding," in *Proc. SPIE Adv. Lithography*, vol. 9781. San Jose, CA, USA, 2016, Art. no. 97810M.

[16] H.-Y. Chang and I. H.-R. Jiang, "Multiple patterning layout decomposition considering complex coloring rules," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2016, pp. 1–6.

[17] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 433–446, Mar. 2015.

[18] 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 *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 591–596.

[19] 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.

[20] P.-Y. Hsu and Y.-W. Chang, "Non-stitch triple patterning-aware routing based on conflict graph pre-coloring," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Chiba, Japan, 2015, pp. 390–395.

[21] Z. Liu, C. Liu, and E. F. Y. Young, "An effective triple patterning aware grid-based detailed routing approach," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, Grenoble, France, 2015, pp. 1641–1646.

[22] B. Yu, X. Xu, J.-R. Gao, and D. Z. Pan, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 349–356.

[23] B. Yu *et al.*, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 726–739, May 2015.

[24] J. Kuang, W.-K. Chow, and E. F. Y. Young, "Triple patterning lithography aware optimization for standard cell based design," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2014, pp. 108–115.

[25] H.-A. Chien, Y.-H. Chen, S.-Y. Han, H.-Y. Lai, and T.-C. Wang, "On refining row-based detailed placement for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 778–793, May 2015.

[26] H. Tian, Y. Du, H. Zhang, Z. Xiao, and M. D. F. Wong, "Triple patterning aware detailed placement with constrained pattern assignment," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2014, pp. 116–123.

[27] T. Lin and C. Chu, "TPL-aware displacement-driven detailed placement refinement with coloring constraints," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Monterey, CA, USA, 2015, pp. 75–80.

[28] Y. Lin *et al.*, "Stitch aware detailed placement for multiple e-beam lithography," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2016, pp. 186–191.

[29] Y. Lin *et al.*, "MrDP: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2016, pp. 1–8.
[30] Y. Lin, B. Yu, and D. Z. Pan, "Detailed placement in advanced technology nodes: A survey," in *Proc. IEEE Int. Conf. Solid State Integr. Circuit Technol. (ICSICT)*, Hangzhou, China, 2016, pp. 25–28.
[31] T. Kauerauf *et al.*, "Reliability of MOL local interconnects," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Monterey, CA, USA, 2013, pp. 2F.5.1–2F.5.5.
[32] M. Rashed *et al.*, "Innovations in special constructs for standard cell libraries in sub 28nm technologies," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2013, pp. 9.7.1–9.7.4.
[33] S. Popovych *et al.*, "Density-aware detailed placement with instant legalization," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.
[34] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan, "A systematic framework for evaluating cell level middle-of-line (MOL) robustness for multiple patterning," in *Proc. SPIE*, vol. 9427. San Jose, CA, USA, 2015, Art. no. 942707.
[35] T. Ohtsuki, H. Mori, E. Kuh, T. Kashiwabara, and T. Fujisawa, "One-dimensional logic gate assignment and interval graphs," *IEEE Trans. Circuits Syst. I*, vol. 26, no. 9, pp. 675–684, Sep. 1979.
[36] S. Chowdhury, "Analytical approaches to the combinatorial optimization in linear placement problems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, no. 6, pp. 630–639, Jun. 1989.
[37] 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.
[38] 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.
[39] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, Paris, France, 1998, pp. 321–324.
[40] U. Brenner and J. Vygen, "Faster optimal single-row placement with fixed ordering," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, Paris, France, 2000, pp. 117–121.
[41] Y. Du and M. D. F. Wong, "Optimization of standard cell based detailed placement for 16 nm FinFET process," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, Dresden, Germany, 2014, pp. 1–6.
[42] S. W. Hur and J. Lillis, "MONGREL: Hybrid techniques for standard cell placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2000, pp. 165–170.
[43] Y. Lin, B. Yu, B. Xu, and D. Z. Pan, "Triple patterning aware detailed placement toward zero cross-row middle-of-line conflict," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 396–403.
[44] *Boost C++ Library, Version 1.55.0*. Accessed on Dec. 13, 2016. [Online]. Available: http://www.boost.org
[45] (2015). *NanGate FreePDK15 Open Cell Library*. [Online]. Available: http://www.nangate.com/?page_id=2328
[46] *Cadence SOC Encounter, Version v09.11-s084_1*. Accessed on Jul. 29, 2015. [Online]. Available: http://www.cadence.com

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 with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

He was an Intern with IMEC, Leuven, Belgium, Cadence, San Jose, CA, USA, and Oracle, Redwood City, CA, USA. His current research interests include physical design and design for manufacturability.

Mr. Lin was a recipient of the Franco Cerrina Memorial Best Student Paper Award at the SPIE Advanced Lithography Conference 2016, and the National Scholarship at Shanghai Jiaotong University in 2012.

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

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Dr. Yu was a recipient of three Best Paper Awards at the 2016 SPIE Advanced Lithography Conference, the 2013 International Conference on Computer Aided Design (ICCAD), and the 2012 Asia and South Pacific Design Automation Conference (ASPDAC), three other Best Paper Award Nominations at the 2014 Design Automation Conference, ASPDAC 2013, and ICCAD 2011, three ICCAD Contest Awards in 2012, 2013, and 2015, the European Design and Automation Association Outstanding Dissertation Award in 2014, the Chinese Government Award for Outstanding Students Abroad in 2014, the SPIE Scholarship in 2013, and the IBM Ph.D. Scholarship in 2012. He has served in the Editorial Boards of *Integration, the VLSI Journal* and *IET Cyber-Physical Systems: Theory and Applications*.

Biying Xu received the B.S. degree in electronics and information engineering from Zhejiang University, Hangzhou, China, in 2014. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

She was an Intern with Synopsys, Inc., Mountain View, CA, USA. Her current research interests include physical design automation for integrated circuits.

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 at Los Angeles (UCLA), Los Angeles, CA, USA.

He was a Research Staff Member with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, from 2000 to 2003. He is currently the Engineering Foundation Endowed Professor with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA. He has published over 250 papers in refereed journals and conferences, and holds eight U.S. patents. His current research interests include cross-layer nanometer IC design for manufacturability, reliability, security, new frontiers of physical design, and computer-aided design for emerging technologies.

Dr. Pan was a recipient of a number of awards, including the Semiconductor Research Corporation (SRC) 2013 Technical Excellence Award, the DAC Top 10 Author in Fifth Decade, the DAC Prolific Author Award, the Asia and South Pacific Design Automation Conference (ASPDAC) Frequently Cited Author Award, 13 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 three times, the IBM Faculty Award four times, the UCLA Engineering Distinguished Young Alumnus Award in 2009, and the University of Texas at Austin RAISE Faculty Excellence Award in 2014. He has served as a Senior Associate Editor for *ACM Transactions on Design Automation of Electronic Systems*, an Associate Editor for the IEEE DESIGN AND TEST, the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, *Science China Information Sciences*, and the *Journal of Computer Science and Technology*. He has served as the Program/General Chair of the ISPD 2007/2008, the TPC Chair for ASPDAC 2016, the Vice Program Chair for the 2017 International Conference on Computer Aided Design, the Tutorial Chair for DAC 2014, among others. He is a Fellow of SPIE.