A Systematic Framework for Evaluating Standard Cell Middle-Of-Line (MOL) Robustness for Multiple Patterning

Xiaoqing Xu[a], Brian Cline[b], Greg Yeric[b], Bei Yu[a] and David Z. Pan[a]

[a]ECE Dept. University of Texas at Austin, Austin, TX 78712 USA
[b]ARM Inc., Austin, TX 78735 USA
Email: {xiaoqingxu, bei, dpan}@cerc.utexas.edu; {brian.cline, greg.yeric}@arm.com

ABSTRACT

Multiple patterning (triple and quadruple patterning) is being considered for use on the Middle-Of-Line (MOL) layers at the 10*nm* technology node and beyond.[1] For robust standard cell design, designers need to improve the inter-cell compatibility for all combinations of cells and cell placements. Multiple patterning colorability checks break the locality of traditional rule checking and N-wise checks are strongly needed to verify the multiple patterning colorability for layout interaction across cell boundaries. In this work, a systematic framework is proposed to evaluate the library-level robustness over multiple patterning from two perpectives, including illegal cell combinations and full chip interactions. With efficient N-wise checks, the vertical and horizontal boundary checks are explored to predict illegal cell combinations. For full chip interactions, random benchmarks are generated by cell shifting and tested to evaluate the placement-level efforts needed to reduce the quadruple patterning to triple patterning for the MOL layer.

Keywords: Multiple Patterning, Middle-Of-Line, Cell Compatibility Check

1. INTRODUCTION

Multiple patterning (MP) has been widely adopted in industry for technology scaling due to the resolution limits of the 193*nm* lithography tools.[1–4] To enable MP lithography, restrictive design rules or constraints have been introduced into the back-end design flow. To deal with these complex constraints, a wide range of research efforts have been focusing on standard cell (SC) design,[2,5] placement,[2,6,7] routing,[8–11] layout migration and decomposition.[12–16] From the library design perspective, it is important to provide quick feedback on the cell layout robustness over MP constraints so that SC designers can improve the cell layout design accordingly. However, there has been few works related to efficient SC library evaluation over MP design constraints. For SC library design, inter-cell compatibility is essential for achieving a robust library that can be used in any design implementation, no matter what kind of placement is implemented for that design. Historically, inter-cell placement compatibility could be guaranteed by checking all 2-cell combinations, because the lithographic interactions were small in comparison to the cell sizes and limited to single mask layers.

With the industry extending 193*nm* lithography via MP, the lithographic interactions now include complex layout design constraints that extend beyond 2-cell interactions. MP coloring and decomposition - especially on middle of line (MOL) layers like the Contact-to-Active (CA) layer[17] that routinely introduces interactions across cell boundaries - breaks the locality of traditional rule checking and makes historical pair-wise cell compatibility checking obsolete. Fig. 1 illustrates the cell interactions introduced by MP. Fig. 1(a) shows a 3-cell interaction that requires triple patterning (TP) to decompose the pattern at the boundary while Fig. 1(b) illustrates a 3-cell interaction example that requires quadruple patterning (QP). Specifically, for the dimensions in the representative 10*nm*/15*nm* technologies used in this work, TP or even QP is needed on the CA layer, depending on the SC boundary conditions and the strength of color-aware placement tools. In cutting-edge technology nodes like 10*nm* and beyond, N-wise checks are now strongly-needed to verify the MP colorability for cell interactions across boundaries, as demonstrated in Fig. 1. This means that what once was a tractable problem in pair-wise cell checking now becomes an exponential problem in the exhaustive N-wise checking. Fortunately, the extremely regular layout structure of layers that interact at the cell boundary leads to large amounts of redundancy among

Design-Process-Technology Co-optimization for Manufacturability IX, edited by John L. Sturtevant, Luigi Capodieci, Proc. of SPIE Vol. 9427, 942707 · © 2015 SPIE · CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2085918

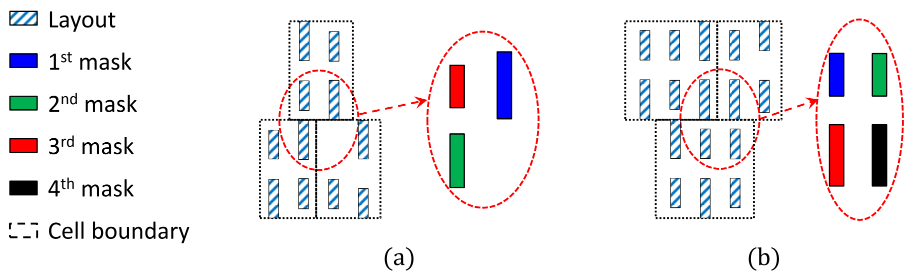


Figure 1. 3-cell interactions, (a) triple patterning, (b) quadruple patterning.

exhaustive inter-cell compatibility checks. For a typical SC library, it becomes worthwhile to explore and remove the redundancy among these inter-cell compatibility checks to make N-wise checks more tractable.

For the $10nm$ technology, the MOL layers, including CA and CB, are essential for the intra-cell routing.[18, 19] This work mainly focuses on the library robustness evaluation on the CA layer due to its complex MP design constraints. Specifically, for the representative $10m/15nm$ technologies used in this study, the CA layer has small dimensions/pitches that requires TP even QP. Compared to other MP layers, such as Metal-1, stitching is not allowed for the CA layer (which is enforced by a technology's Design Rule Checks - DRC's). In addition, the CA layer routinely introduces interactions across rows under MP constraints, which makes N-wise checks particularly important in the library evaluation stage. However, our evaluation framework can be easily adapted to other MP layers, such as Metal-1.

For library robustness evaluation, the inter-cell compatibility checks should report all the illegal cell combinations to the designers. Then, designers can improve the library cell layout and reduce the number of illegal cell combinations as much as possible. For the representative $10nm/15nm$ libraries we used, the inter-cell compatibility checks on the CA layer reveal that the TP DRC violations originate from the local coloring conflicts. Therefore, we further evaluate the robustness of the library over TP from the perspective of full chip interactions. In this work, the concepts of "cell sequence" and "multi-row cell compatibility checking" are proposed to enable efficient N-wise inter-cell compatibility checks. We manage the exponential growth of the N-wise checks by exploiting the pattern regularity of the CA layer and remove redundancy among various cell sequences. Vertical and horizontal boundary checks are explored in order to predict illegal cell combinations. Then, multiple rows of independent cell sequences are randomly combined to evaluate the full chip interactions on the CA layer over MP. Our main contributions are summarized as follows:

- A systematic framework is proposed to evaluate the library-level MOL robustness for MP.
- To enable efficient N-wise checks, the cell sequence size reduction problem is proposed and solved efficiently with a graph-based approach and divide-and-conquer technique.
- The illegal cell combinations are predicted with efficient horizontal and vertical boundary checks for a given SC library.
- For specific SC libraries, the full chip interactions are explored to evaluate the placement-level efforts in reducing the QP to TP for the MOL layer.

The rest of this paper is organized as follows: Section 2 introduces the related background information, basic definitions and overall framework. Section 3 presents the formulation and solution of the cell sequence size reduction problem. Section 4 introduces the multi-row cell compatibility checking to enable efficient horizontal and vertical boundary checks with the elementary checker. Section 5 demonstrates the effectiveness of our framework with the results on two sets of benchmarks, including ARM $10nm$ representative library and modified Open NanGate $15nm$ library.[20] Several key observations are discussed in detail.

2. PRELIMINARIES AND OVERALL FLOW

2.1 Middle-Of-Line and Row Structure

In advanced technology nodes, the complex lithography and process constraints prevent simply using Metal-1 layer for intra-cell routing. Hence, the MOL layers have been adopted for intra-cell signal-wiring in the 20nm technology node and beyond.[18,19] As shown in Fig. 2(a), typical MOL layers include CA and CB.[17] In particular, there may be cross-coupled connections, i.e. special constructs, in the CA layer depending on the technological choices. In the placement row structure shown in Fig. 2(b), SC's are aligned horizontally and share the same height. The mirrored structure of the power and ground rails goes from the left to right of the design. For practical placement, there might be white spaces among neighboring cells on the same row. However, in the library design or evaluation stage, the exact amount of white spaces between any cell pair is unknown without placement information. A typical way to enable robust SC design is to estimate the white spaces conservatively. As illustrated in Fig. 2(c), the white spaces among cells on a single row are removed and cells are abutted horizontally. Thus, we have the following definition.

Definition 1 (Cell Sequence). *A set of cells abutting each other in the horizontal direction without overlapping or white space is called a cell sequence. If the sequence consists of n SC's, it is defined as an n-cell sequence.*

An example of the 4-cell sequence is shown in Fig. 2(c).

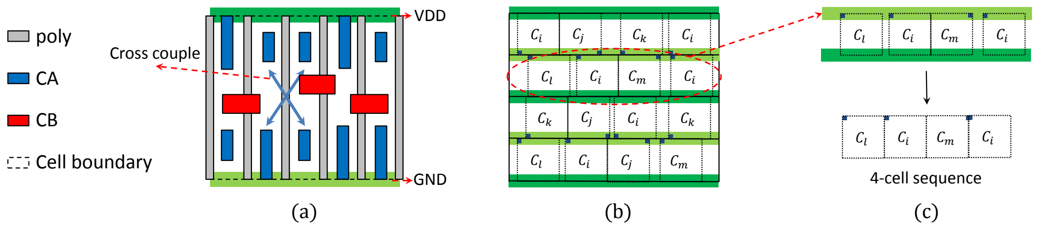


Figure 2. Standard cell and row structure, (a) MOL structures, (b) multiple rows, (c) one single row and 4-cell sequence.

2.2 Conflict Graph

To determine whether the target layout is free of MP DRC violations, i.e. k-patterning friendly, the first step is to built the conflict graph.[13] Fig. 3(a) and (b) shows a typical layout in the CA layer and the corresponding conflict graph, respectively. Then, the result of the k-patterning DRC is determined by the k-colorability of the conflict graph. For example, the target layout passes the quadruple ($k = 4$) patterning coloring check in Fig. 3(b) and the mask assignment is demonstrated in Fig. 3(c). The k-patterning design rule check is similar to the problem of layout decomposition. The MP layout decomposition problem is well-studied for double patterning,[13,21–23] triple patterning[14,24–28] and quadruple patterning.[16] Due to the timing criticality of the MOL layers for the SC design, we assume no stitches are allowed for the feasible coloring solution. In addition, MP DRC aims at deciding the existence of some coloring solution instead of finding an optimal one in layout decomposition. Then, an elementary DRC checker is presented to determine the k-colorability of the conflict graph for the CA layer.

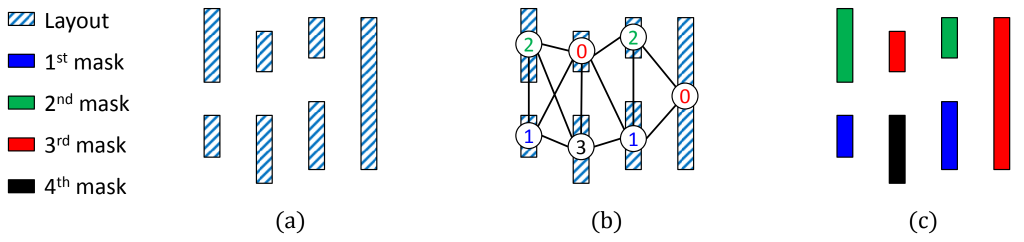


Figure 3. Layout graph for MP layout decomposition, (a) target layout, (b) graph and coloring, (c) mask assignment.

2.3 Overall Flow

The overall flow for our framework in demonstrated in Fig. 4. At the placement level, the TP or QP DRC involves 3-cell interactions as shown in Fig. 1. Thus, we need to enable N-wise checks across multiple rows to predict illegal cell combinations. Given a SC library, the framework starts with the cell sequence construction, which produces a set of independent k-cell sequences. Then, the two-row cells are exhaustively built from the independent cell sequences to compute the illegal cell combinations. For representative libraries we used, the cell compatibility checks reveal that the violations of TP DRCs come from the local conflicts on the CA layer. Random benchmarks are constructed to explore full chip interactions. This further produces the cell shift histogram to evaluate the placement-level efforts needed to reduce QP to TP. Reducing from QP to TP is potentially attractive because it may allow us to save one mask.

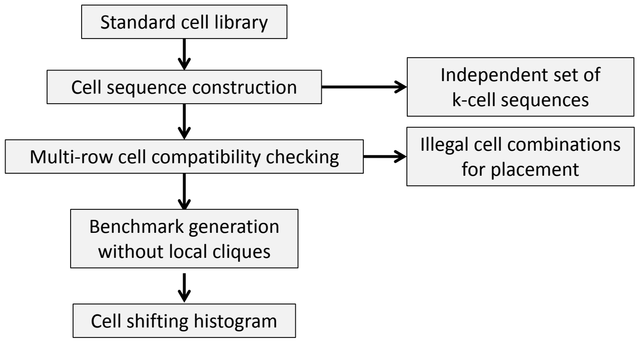

Standard cell library
Cell sequence construction
Independent set of k-cell sequences
Multi-row cell compatibility checking
Illegal cell combinations for placement
Benchmark generation without local cliques
Cell shifting histogram

Figure 4. The overall flow for our framework.

3. CELL SEQUENCE CONSTRUCTION

In this section, we first define the cell sequence redundancy given a pair of cell sequences. Then, the redundancy graph and divide-and-conquer technique are presented to remove the redundancy and compute the independent set of cell sequences. In the end, we demonstrate the efficient cell sequence construction algorithm.

3.1 Cell Sequence Size Reduction

For the k-patterning DRC, the pass or violation is decided by the k-colorability of the conflict graph constructed from the layout.[13, 14, 16] For the k-colorability of a graph, we have the following theorem.

Theorem 1. *If a graph is k-colorable, a subset of the graph is also k-colorable. The subset of a graph is achieved by removing some nodes or edges of the given graph.*

Proof. A graph is k-colorable means a feasible coloring solution exists for the graph. Pick one feasible solution and assign color to each node. Then, we have legal color assigned to each node of the subset of graph, which means the subset of the graph is also k-colorable. $\square$

With the regular layout structure on the CA layer, the conflict graph is determined by a set of rectangles and relative positions among them. Fig. 5(a) illustrates the layout of two 1-cell sequences , namely C_i and C_j, on the CA layer. As shown in Fig. 5(b), the layout of C_j is a **superset** of the layout of C_i on both the left and right boundary of C_j. Specifically, the left boundary layout of C_j exactly matches the layout of C_i. Moreover, the line ends of the right boundary layout of C_j are extended vertically compared to the layout of C_i, which may induce extra conflict edges during conflict graph construction. This means that the conflict graph for C_i is the subset of the conflict graph for C_j on both the left and right boundary. According to Theorem 1, the k-colorability of C_j guarantees that of C_i. Then, the k-patterning check for C_i will be redundant to that for C_j. As mentioned earlier, the MP DRC involves N-wise checks, which brings the necessity of k-cell sequence construction. Then, the redundancy of C_i to C_j is further demonstrated in Fig. 5(c). When we build the 2-cell sequences introducing another cell C_k, the layout of C_kC_j is a superset of the layout of C_kC_i and a similar relationship exists between C_jC_k and C_iC_k. If C_j is included in the independent set of cell sequences, C_i should be excluded from that independent set for both MP DRC and cell sequence construction, since including C_i would be redundant.

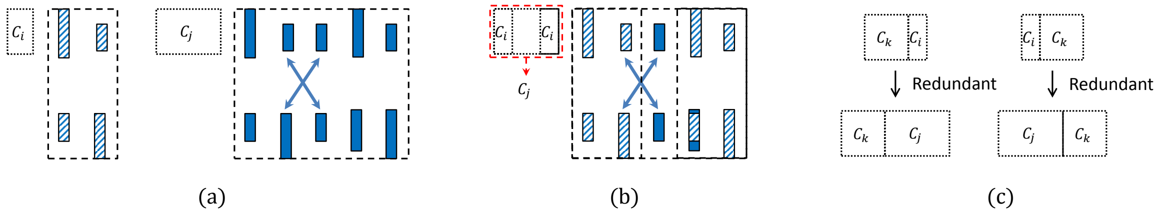


Figure 5. Sequence cell redundancy, (a) cell C_i and C_j, (b) C_i redundant to C_j, (c) C_kC_i redundant to C_kC_j, C_iC_k redundant to C_jC_k.

Redundancy Graph The ultimate goal for the cell sequence size reduction is to reduce the number of N-wise checks for MP. Given a set of cell sequences, we introduce the redundancy graph for removing redundancy and compute an independent set of cell sequences. The redundancy graph is a directed graph with each node representing one cell sequence. As shown in Fig. 6, there are n nodes and node C_i represents the cell sequence C_i. Now, we may use C_i to represent the cell sequence and the corresponding graph node interchangeably. In addition, an edge from node C_i to node C_j denotes that cell sequence C_i is redundant to cell sequence C_j. For example, in Fig. 6, an edge from C_3 to C_6 means cell sequence C_3 is redundant to C_6. In the end, with the redundancy graph constructed from the input set of cell sequences, the independent set consists of those nodes without successors in the graph. For instance, the dashed nodes $(C_2, C_5, C_6, C_{n-3}, C_{n-1})$ in Fig. 6 belong to the independent cell sequence set.

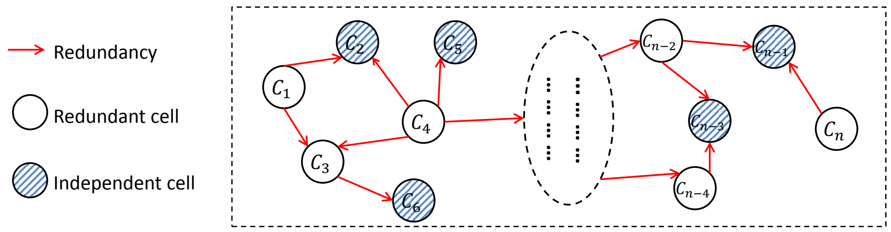


Figure 6. Cell sequence redundancy graph.

Given n cell sequences, the time complexity to build a redundancy graph is $O(n^2)$ because we need to determine the redundancy between each pair of cells. For practical implementations, the input size can be as large as $n \geq 10^6$, which would incur large runtime. Thus, a divide and conquer technique is deployed to optimize the set within an affordable runtime. Algorithm 1 illustrates the details on the recursive divide and conquer technique with the redundancy graph construction. Lines 5-10 explain the redundancy graph construction and independent set computation if the input size is within the graph node size bound (n_0). Otherwise, the input cell sequence set is segmented into various subsets and recursively computed. It can be observed that the inter-subset redundancy among cell sequences on different subsets are ignored in Algorithm 1. To account for that, Algorithm 2 demonstrates the iterative approach to implement the cell sequence size reduction. For each iteration from line 4 to line 9, the cell sequence set is optimized using the divide and conquer technique and the set is randomly shuffled for the next iteration. The exit condition is that no redundant cell is found or the iteration count exceeds the pre-set bound (m_0). A more intuitive interpretation of Algorithm 1 and Algorithm 2 is illustrated in Fig. 7. The input set is segmented into several subsets in Fig. 7(a) and each subset is optimized based on redundancy graph in Fig. 7(b) and the results are shown in Fig. 7(c). To prepare for the next iteration, the optimized set is randomly shuffled in Fig. 7(d).

3.2 Cell Sequence Construction

Given a standard cell library consisting of m cells, i.e. m 1-cell sequences, the exhaustive construction scheme gives the number of n-cell sequences as $(2 * m)^n$ (the factor of 2 accounts for horizontal cell flipping). A simple calculation with $m = 100$ yields that the number of 4-cell sequences will be 1.6 billion. To manage the size to an acceptable level, cell sequences are built in a bottom-up manner as shown in Algorithm 3. We iterate through

Algorithm 1 Divide and Conquer

```
Require: A set of cell sequences ($CS$), graph node size bound ($n_0$);
 1: Define $ICS$ as the independent cell sequence set;
 2: Define $RG(CS)$ as the redundancy graph for set $CS$;
 3: function DivideConquer($CS$)
 4:     Compute $size$ = size of $CS$;
 5:     if $size \leq n_0$ then
 6:         for each node $v_i$ in $RG(CS)$ do
 7:             if $v_i$ has no successors then
 8:                 add $v_i$ to $ICS$;
 9:             end if
10:         end for
11:     else
12:         Divide $CS$ into $\sqrt{size}$ subsets;
13:         for each $subset$ for $CS$ do
14:             DivideConquer($subset$);
15:         end for
16:     end if
17: end function
```

Algorithm 2 Cell Sequence Size Reduction (CSSR)

```
Require: A set of cell sequences ($CS$), iteration bound ($m_0$);
 1: Define iteration count $m = 0$;
 2: Define $ICS$ as the independent cell sequence set;
 3: while $m \leq m_0$ do
 4:     $ICS$ = DivideConquer($CS$);
 5:     if size of $ICS <$ size of $CS$ then
 6:         $CS$ = random-shuffle($ICS$);
 7:     else
 8:         break;
 9:     end if
10:     $m = m + 1$;
11: end while
```

the number of cells, denoted as n, in one sequence in line 4. The independent n-cell sequences are computed in line 5. Then, in lines 6-9, the $(n+1)$-cell sequences are constructed from the independent sets, including the 1-cell sequence set to the n-cell sequence set. The iteration ends with the pre-set maximum number of cells (n_0) in one sequence. With Algorithm 3, the n-cell sequences are constructed from the independent cell sequences in a bottom up manner. For practical implementations, this method helps to achieve the complete set of 3-cell sequences in reasonable amount of runtime despite of the exponential growth of the cell sequences.

Algorithm 3 Cell Sequence Construction

```
Require: The set of 1-cell sequences ($CS_1$) in the library and the maximum number of cells in one sequence ($n_0$);
 1: Define $CS_n$ as the set of $n$-cell sequences;
 2: Define $ICS_n$ as the set of independent $n$-cell sequences;
 3: Define initial number of cells in one sequence as $n = 1$;
 4: while $n <= n_0$ do
 5:     $ICS_n$ = CSSR($CS_n$);
 6:     for $m = 1; m \leq \lfloor (n+1)/2 \rfloor; m = m + 1$ do
 7:         Construct the $(n+1)$-cell sequence set ($set_{n+1}$) from $ICS_m$ and $ICS_{n+1-m}$;
 8:         $CS_{n+1} = CS_{n+1} \cup set_{n+1}$;
 9:     end for
10:     $n = n + 1$;
11: end while
```

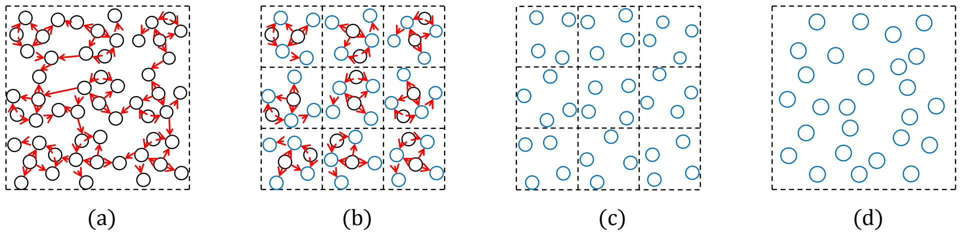


Figure 7. Divide and conquer, (a) divide the cell sequences into several subsets, (b) remove redundancy in each subset, (c) random shuffle, (d) for next iteration.

4. MULTI-ROW CELL COMPATIBILITY CHECKING

In this section, we build two-row cells and multi-row cells, to predict illegal cell combinations and explore full chip interactions, respectively. An elementary checker is presented predict the local conflicts and non-local conflicts for TP.

4.1 Two-Row Cells

Ideally, we need to exhaustively build cell combinations across as many rows as possible to explore long-range interactions for the MP DRC. However, exhaustive two-row cell checks are sufficient for the local conflict detections for QP and TP. In addition, the exponential growth of the cell combinations beyond 2 rows makes it unaffordable in practice. Hence, we only exhaustively build two-row cells from the independent cell sequences. In addition, we also enable the cell sequence shifting to account for the relative position changes of two cell sequences on neighboring rows. As shown in Fig. 8(a), one type of two-row cell consists of an independent 1-cell sequence and an independent 3-cell sequence on top of it, which is denoted as ics_1 on ics_3. For practical placement, the 1-cell sequence in Fig. 8(a) can be anywhere on top of the 3-cell sequence cell. Therefore, we allow for the cell sequence shifting from left to right in the unit of placement pitch and the k-pattering colorability is checked for each placement pitch. Moreover, Fig. 8(b) shows another type of two-row cell denoted as ics_2 on ics_2 and the cell shifting is also supported for the DRC. For practical implementations, we build three types of two-row cells, including ics_1 on ics_2, ics_1 on ics_3 and ics_2 on ics_2.

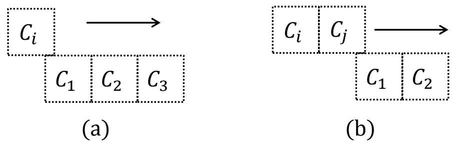


Figure 8. Two-row cells, (a) independent 1-cell sequence on independent 3-cell sequence (ics_1 on ics_3), (b) independent 2-cell sequence on independent 2-cell sequence (ics_2 on ics_2).

4.2 Multi-Row Cells

Next, we randomly build multi-row cells to explore the full chip interactions. Specifically, benchmarks with multiple rows are randomly generated from the independent 1-cell sequences as illustrated in Fig. 9. We pre-set a bound for the design width and height for placement. Independent 1-cell sequences are randomly chosen and abutted horizontally and vertically to fill in the placement region. In Fig. 9(a), 1-cell sequences are abutted horizontally without gaps among their placement and routing boundaries. However, for Fig. 9(b), the cell shifting is enabled to greedily remove the local conflicts for MP, which creates white spaces among cells in the same row. As discussed in Section 5, these types of multi-row cells help to evaluate the placement-level efforts required to reduce QP to TP. In particular, there exists white spaces at the end of each placement row when the random cell can not be inserted due to the fixed design width.

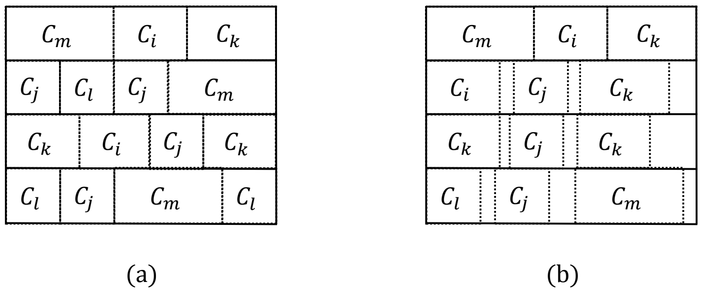


Figure 9. Multi-row cells, (a) Multiple rows of 1-cell sequences, (b) Multiple rows of 1-cell sequences with local conflict removed.

4.3 Multiple Patterning Compatibility Check

Given a cell combination, our elementary k-patterning checker starts with the conflict graph construction. With the conflict graph, the k-patterning DRC consists of three stages, including iterative vertex removal, clique detection and backtrack coloring. The iterative vertex removal means that any conflict graph node with degree $< k$ can be removed without impacting the colorability of the graph.[16, 24] The clique detection searches for a $(k+1)$-clique, which is not k-colorable. For the complex conflict graph without $(k+1)$-cliques, the backtrack coloring is deployed to determine the k-colorability of the conflict graph in a brute-force manner. Our elementary checker works for small scale conflict graph within reasonable runtime.

Local conflict and non-local conflict Next, we discuss why QP, instead of TP, is needed for the CA layer in the 10nm representative technology. With the cell combinations constructed and our elementary checker, we can enable the efficient 3-patterning checks. In general, we find that there are two types of conflicts, including local conflicts and non-local conflicts, which lead to the TP DRC violations. As show in Fig. 10(a) and (b), the local conflict originates from the local 4-clique involving neighboring rows. The non-local conflict is due to the coloring failure of the layout across three rows or beyond. An example is illustrated in Fig. 10(a), (c) and (d). For the 3-coloring of the graph, if we have adjacent 3-circles sharing one edge, the two nodes without edge connection should share the same color. As shown in the dashed rectangle in Fig. 10(d), the two nodes without edge connection share the same color, denoted as "1". The same color constraint propagates across the entire graph and introduces a coloring conflict in the dash ellipse in Fig. 10(d).

Furthermore, the graph nodes due to the special constructs, as shown in Fig. 10(c), play a critical role in the non-local conflicts. If we forbid or remove the special construct in the library design, non-local conflicts never happen in our 3-patterning checks. This means the TP DRC violations are simply due to the local conflicts, i.e. local 4-cliques for TP. This also leads to the possibility of removing the local 4-cliques in the placement level and reducing the QP to TP for the CA layer. A local 4-clique consists of 4 rectangles, which at most involves 4 standard cells neighboring each other, and greedy local 4-clique removal for multi-row cells can be achieved by shifting cells and adding white spaces, as shown in Fig. 9(b).

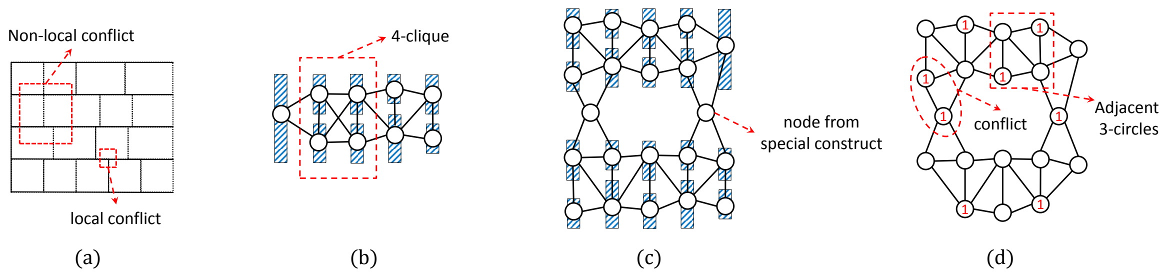


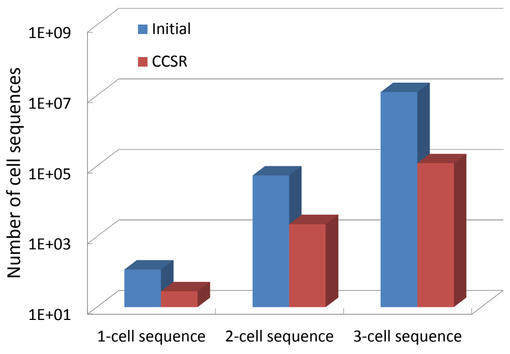
Figure 10. Violations of TP DRCs, (a) the local and non-local conflicts (b) the local conflict, (c) the non-local conflict, (d) the coloring failure for the non-local conflict.

5. EXPERIMENTAL RESULTS

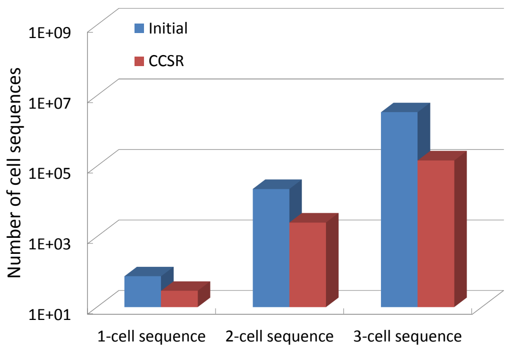
We have implemented our algorithms in C++ and tested them with two sets of benchmarks. One is an ARM representative $10nm$ standard cell library consisting of 116 logic cells. The other is the NanGate $15nm$ open cell library[20] consisting of 76 logic cells. All experiments on ARM representative $10nm$ library are performed on a Linux machine with 3.33GHz Intel(R) Xeon(R) CPU X5680. For experiments on the modified NanGate $15nm$ library, they are performed on a Linux machine with 3.4GHz Intel(R) Core and 32GB memory. For the NanGate $15nm$ open cell library, there exits three MOL layers for intra-cell routing. The CA layer is pre-designed as being explicitly decomposed into two masks on separate layers. In order to test our methodology, we recombine these two MOL layers into one single layer, and then run our algorithms on the modified layout using the combined MOL layer. In Algorithm 1, the graph size bound (n_0) is set as 100 and in Algorithm 2, the iteration bound (m_0) is set as 10. We first show the benefit of our cell sequence size reduction. After identifying independent cell sequences, the size reduction of two-row cells are demonstrated. Then, we discuss a set of observations from the multi-row cell compatibility checking. In particular, the cell shifting histogram is introduced to demonstrate the placement-level efforts required to reduce QP to TP for the CA layer with the representative libraries we used.

5.1 Cell Sequence Size Reduction

The cell sequence size reduction helps remove the redundancy among cell sequences and enables efficient cell sequence construction as discussed in Section 3. However, from the theoretical perspective, the number of cell sequences still grows exponentially with the length of the sequence. As shown in Fig. 11, the maximum number of cells in one sequence we can achieve is only 3. However, the 3-cell sequences are sufficient to explore the two-row cell interactions for TP and QP. Meanwhile, we see orders of magnitude reduction on the number of cell sequences from the brute-force construction for both the ARM and NanGate representative libraries. In Fig. 11, "Initial" denotes the brute-force construction while the "CSSR" denotes the cell sequence size reduction. Meanwhile, for each 3-cell sequence built from the library, it is either included in the independent set or redundant to some sequence within the independent set. Moreover, we discuss the necessity of the divide and conquer approximation and speedup. As mentioned earlier, the redundancy graph construction has the time complexity as $O(n^2)$, where n denotes the graph node size. Here, we give an example on the runtime for practical implementations. For ARM representative library, the number of 3-cell sequences before redundancy removal is $n = 120960$ and the average amount of time to compute the redundancy between a pair of 3-cell sequences is $t_0 = 0.36s$. Then, we can estimate the amount of runtime to compute the redundancy graph without the speedup will be $t = 0.5 * n^2 * t_0$, which is approximately 83 years. With the divide and conquer speedup, we achieve the results in Fig. 11 within several hours.



(a)

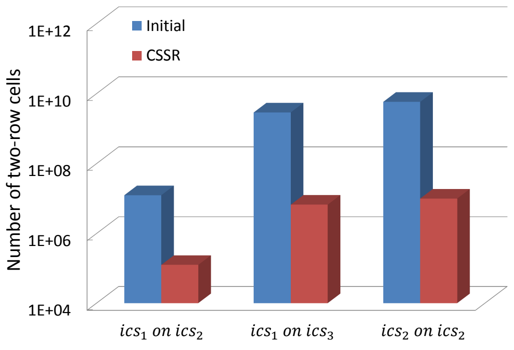


(b)

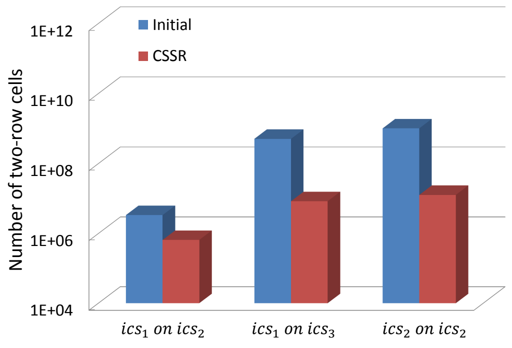
Figure 11. Cell sequence size reduction, (a) ARM $10nm$ representative library, (b) Modified NanGate $15nm$ library.

5.2 Size Reduction of Two-Row Cells

Fig. 12 demonstrates the size reduction of two-row cells. For the horizontal axis, the "ics_i *on* ics_j" denotes one type of two-row cells, which has an i-cell sequence on top of a j-cell sequence. As discussed in Section 4, the two-row cells are only exhaustively built from the independent cell sequences based on the cell sequence size reduction. Then, the size of "ics_i on ics_j" = 2 $*$ the size of i-cell sequences $*$ the size of j-cell sequences, where the factor of 2 accounts for the vertical flipping of cell sequences. Actually, the orders of magnitude reductions for two-row cells are due to the cell sequence size reduction in Fig. 11. For example, in Fig. 12(a), we observe that billions of "ics_1 on ics_3" are reduced to millions of them. In addition, any two-row cell constructed from the cell library is either included in the reduced sets or redundant to some two-row cell in the reduced sets.



(a)



(b)

Figure 12. Size reduction of two-row cells from the cell sequence size reduction, (a) ARM 10nm representative library, (b) Modified NanGate 15nm library.

5.3 Full Chip Interactions

Table 1 and Table 2 demonstrate the multi-row cells randomly generated from the set of independent 1-cell sequences for the NanGate 15nm library and the ARM 10nm representative library, respectively. Within the tables, the "Bench" denotes the benchmark name, the "size" is the area for placement, the "Util(%)" is the utilization rate representing the percentage of cell area over total area of the design, the "Cell#" is the number of cells, the "Rect#" is the number of rectangles, "Pass" denotes the DRC result and "CPU" denotes the amount of runtime. We have the utilization rate (Util(%)) less than 100% because white space is left at the end of a row when there is not enough to insert another random 1-cell sequence. As shown in both tables, all the multi-row cells pass the QP DRCs. In contrast, TP DRCs either lead to violations or can not be finished within an acceptable amount of runtime. As mentioned earlier, the violations of TP DRCs originate from two types of conflicts, including local conflicts and non-local conflicts. The non-local conflicts are due to the propagation of same color constraints across three rows or beyond as shown in Fig. 10(d). The conflict graph nodes from the special constructs prevents simplifying the graph into small components. Then, we may encounter large graph node sizes for the multi-row cells, which makes the 3-coloring problem intractable. For instance, the TP DRCs for the multi-row cells from ARM_4 to ARM_6 in Table 2 can not be finished within a reasonable amount of runtime due to the large graph sizes. Actually, the use of special constructs varies from technology to technology and the NanGate 15nm library does not have this cross-coupled structure. Thus, the 3-patterning checks for all the multi-row cells from the NanGate library can be achieved in Table 1 (without the special construct that merges the graph across a row, the graph naturally partitions itself by row, which leads to much more manageable graph sizes and runtimes).

Table 1. NanGate 15nm multi-row cells with local 4-cliques

					TP Checks		QP checks	
Bench	Size(um^2)	Util(%)	Cell#	Rect#	Pass	CPU(s)	Pass	CPU(s)
NanGate_1	0.8 x 0.8	91.0	77	1870	No	0.07	Yes	0.04
NanGate_2	1.5 x 1.5	96.5	288	7875	No	0.35	Yes	0.29
NanGate_3	31 x 31	98.0	1257	31783	No	3.37	Yes	2.83
NanGate_4	61 x 61	99.0	4948	12864	No	52.7	Yes	39.6
NanGate_5	115 x 115	99.5	17382	454671	No	584.9	Yes	449.1
NanGate_6	154 x 154	99.6	31057	809015	No	1823.9	Yes	1403.4

Table 2. ARM $10nm$ representative multi-row cells with local 4-cliques

					TP Checks		QP checks	
Bench	Size(um^2)	Util(%)	Cell#	Rect#	Pass	CPU(s)	Pass	CPU(s)
ARM_1	17.2 x 17.2	73.6	29	937	No	0.04	Yes	0.02
ARM_2	34.6 x 34.6	86.3	124	4411	No	24.4	Yes	0.15
ARM_3	69.1 x 69.1	93.4	444	19381	No	915.6	Yes	1.36
ARM_4	138 x 138	95.4	1831	79287	No	308605.0	Yes	16.8
ARM_5	259 x 259	97.5	6362	284933	n/a	n/a	Yes	200.0
ARM_6	346 x 346	98.3	11261	511888	n/a	n/a	Yes	638.4

For the NanGate library without special constructs, the TP DRC violations of multi-row cells are simply due to the local 4-cliques. As shown in Fig. 13(a), the conflict graph on the CA layer are split into independent components in a row by row manner. The local 4-cliques come from the cell interactions between neighboring rows. The multi-row cells without local 4-cliques are still built horizontally and vertically from the random 1-cell sequences. When local 4-cliques are detected, the current cell being inserted is shifted from left to right by one placement pitch until all local 4-cliques are eliminated.

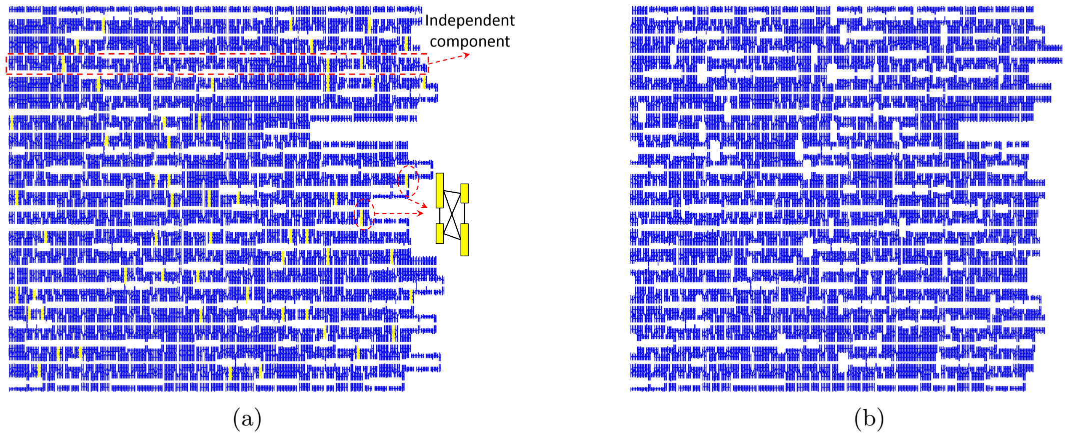
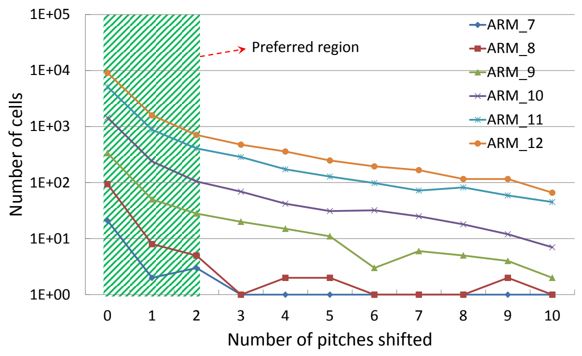


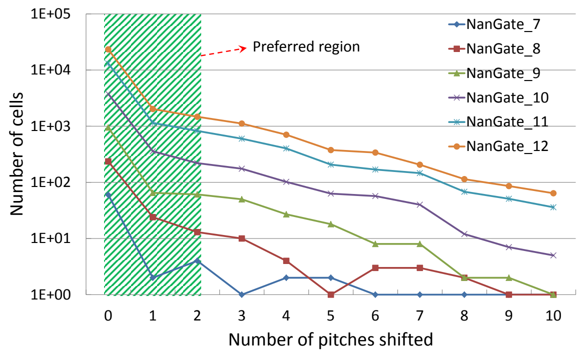
Figure 13. Benchmarks with multiple rows, (a) with local 4-cliques, (b) without local 4-cliques.

The non-local conflicts for TP are inevitable if special constructs are included in the SC design as discussed in Section 4.3. If the special constructs are excluded, it is possible to resolve the local conflicts in the placement level and reduce QP to TP. Therefore, we build two sets of multi-row cells, one is from the NanGate library in Table 3 and the other is obtained by modifying the ARM library and removing the special constructs within the standard cells in Table 4. We first discuss the **cell shifting histogram** in Fig. 14. In the cell shifting histogram, the horizontal axis is the number of placement pitches each inserted cell is shifted before achieving a placement site that is free of local 4-cliques and the vertical axis is the number of cells. In general, the cell shifting histogram is an initial attempt to capture the effort that might be needed by a placement algorithm in order to achieve a legal placement given the same scenario of cells. For most benchmarks, the number of cells is a non-increasing function of the number pitches shifted. To the first order, we prefer that cells are shifted less than 3 placement pitches, as shown in the shadowed region marked, "Preferred region", in Fig. 14. In the representative libraries we used, a minimum-sized inverter occupied 3 placement pitches (in width), so we chose to use the same width as the upper bound of our "Preferred Region". For each benchmark, we further calculate the percentage of cells in the preferred region and the utilization rate, which are two metrics to evaluate the placement-level efforts to reduce QP to TP. If TP on the CA layer for practical designs is desired, we expect to require that a high percentage of cells would be in the preferred region while the utilization rate of the placement is also high. A high percentage of cells in the preferred region means that minimum placement-level efforts will be needed to remove the local 4-cliques and achieve TP friendly design. The high utilization of each benchmark means that the cell shifting in the placement stage will introduce less area penalty. To improve the robustness at the standard cell library level, SC designers need to increase the utilization rate and the percentage of cells in the preferred region for the multi-row cells.

In the end, the multi-row cell compatibility checking yields a unique set of observations for the representative $10nm$ technology. (1) The 4-clique exists due to the row to row interactions, which means QP needs to be deployed to enable implicit coloring for standard cells; (2) There is no local five-clique for the CA layer in the representative technology used; (3) For all the two-row cells and multi-row cells generated, the colorability for QP can be easily proved with our elementary checker.



(a)



(b)

Figure 14. The cell shifting histogram denoting the number of pitches shifted to remove local 4-cliques, (a) ARM $10nm$ representative library (special constructs removed), (b) Modified NanGate $15nm$ library.

Table 3. NanGate $15nm$ multi-row cells without local 4-cliques

						TP Checks	
Bench	Size(um^2)	Util(%)	Cell#	Rect#	C.i.p(%)	Pass	CPU(s)
NanGate_7	0.8 x 0.8	91.3	72	1865	91.7	Yes	0.02
NanGate_8	1.5 x 1.5	92.8	295	7499	92.5	Yes	0.19
NanGate_9	31 x 31	94.6	1191	30561	90.3	Yes	2.53
NanGate_10	61 x 61	95.6	4800	123621	90.4	Yes	37.9
NanGate_11	115 x 115	95.5	16757	433639	90.0	Yes	461.8
NanGate_12	154 x 154	95.7	29870	771964	89.9	Yes	1475.8

Table 4. ARM $10nm$ representative multi-row cells (special constructs removed) without local 4-cliques

						TP Checks	
Bench	Size(um^2)	Util(%)	Cell#	Rect#	C.i.p(%)	Pass	CPU(s)
ARM_7	17.2 x 17.2	89.7	29	913	89.7	Yes	0.09
ARM_8	34.6 x 34.6	86.9	117	4443	91.5	Yes	1.00
ARM_9	69.1 x 69.1	84.7	480	17300	86.3	Yes	2.23
ARM_10	138 x 138	92.3	2028	75405	88.4	Yes	1237.6
ARM_11	259 x 259	93.2	7338	267291	87.1	Yes	4904.2
ARM_12	346 x 346	93.5	13162	476000	86.8	Yes	1502.1

6. CONCLUSION

This paper proposes a comprehensive framework to evaluate the MOL robustness for a standard cell library over multiple patterning. To reduce the number of N-wise checks, the issues of cell sequence construction and size reduction are proposed and solved efficiently. The multi-row cell compatibility checking and an elementary checker are presented to enable efficient N-wise checks. For representative $10nm/15nm$ libraries we used, the cell shifting histogram is proposed to quantify the placement-level efforts required to reduce QP to TP for the CA layer.

REFERENCES

[1] R. Merritt, "Intel sees quad-patterned path to 10-nm chips." `http://www.eetimes.com/document.asp?doc_id=1262512`, September 2012.

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

[3] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* **32**(10), pp. 1453–1472, 2013.

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

[5] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan, "Self-aligned double patterning aware pin access and standard cell layout co-optimization," in *ACM International Symposium on Physical Design (ISPD)*, pp. 101–108, 2014.

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

[7] B. Yu, X. Xu, J.-R. Gao, and D. Z. Pan, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 349–356, 2013.

[8] M. Cho, Y. Ban, and D. Z. Pan, "Double patterning technology friendly detailed routing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 506–511, 2008.

[9] M. Mirsaeedi, J. A. Torres, and M. Anis, "Self-aligned double-patterning (SADP) friendly detailed routing," in *Proc. of SPIE*, **7974**, 2011.

[10] 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 *ACM/IEEE Design Automation Conference (DAC)*, pp. 591–596, 2012.

[11] Y.-H. Lin, B. Yu, D. Z. Pan, and Y.-L. Li, "TRIAD: A triple patterning lithography aware detailed router," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 123–129, 2012.

[12] C.-H. Hsu, Y.-W. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 595–600, 2009.

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

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

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

[16] B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2014.

[17] M. Rashed, N. Jain, J. Kim, M. Tarabbia, I. Rahim, S. Ahmed, J. Kim, I. Lin, S. Chan, H. Yoshida, *et al.*, "Innovations in special constructs for standard cell libraries in sub 28nm technologies," in *IEEE International Electron Devices Meeting (IEDM)*, pp. 248–251, 2013.

[18] G. Northrop, "Design technology co-optimization in technology definition for 22nm and beyond," in *IEEE Symposium on VLSI Technology (VLSIT)*, pp. 112–113, 2011.

[19] M. Rashed, S. Soss, J. Kye, I. Lin, J. Gullette, C. Nguyen, J. Kim, M. Tarabbia, Y. Ma, Y. Deng, *et al.*, "Semiconductor device with transistor local interconnects," Nov. 12 2013. US Patent 8,581,348.

[20] "NanGate FreePDK15 Open Cell Library." `http://www.nangate.com/?page_id=2328`.

[21] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *ACM International Symposium on Physical Design (ISPD)*, pp. 185–196, 2009.

[22] Y. Xu and C. Chu, "A matching based decomposer for double patterning lithography," in *ACM International Symposium on Physical Design (ISPD)*, pp. 121–126, 2010.

[23] X. Tang and M. Cho, "Optimal layout decomposition for double patterning technology," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 9–13, 2011.

[24] S.-Y. Fang, W.-Y. Chen, and Y.-W. Chang, "A novel layout decomposition algorithm for triple patterning lithography," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 1185–1190, 2012.

[25] H. Tian, H. Zhang, Q. Ma, Z. Xiao, and M. Wong, "A polynomial time triple patterning algorithm for cell based row-structure layout," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 57–64, 2012.

[26] J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 69:1–19:6, 2013.

[27] B. Yu, Y.-H. Lin, G. Luk-Pat, D. Ding, K. Lucas, and D. Z. Pan, "A high-performance triple patterning layout decomposer with balanced density," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 163–169, 2013.

[28] Y. Zhang, W.-S. Luk, H. Zhou, C. Yan, and X. Zeng, "Layout decomposition with pairwise coloring for multiple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 170–177, 2013.