Triple patterning lithography layout decomposition using end-cutting

Bei Yu
Subhendu Roy
Jhih-Rong Gao
David Z. Pan

Triple patterning lithography layout decomposition using end-cutting

Bei Yu,[a,*] Subhendu Roy,[a] Jhih-Rong Gao,[b] and David Z. Pan[a]
[a]University of Texas at Austin, ECE Department, Austin, Texas 78712, United States
[b]Cadence Design Systems, 12515-7 Research Boulevard, Austin, Texas 78759, United States

Abstract. Triple patterning lithography (TPL) is one of the most promising techniques in the 14-nm logic node and beyond. Conventional LELELE type TPL technology suffers from native conflict and overlapping problems. Recently, as an alternative process, TPL with end-cutting (LELE-EC) was proposed to overcome the limitations of LELELE manufacturing. In the LELE-EC process, the first two masks are LELE type double patterning, while the third mask is used to generate the end-cuts. Although the layout decomposition problem for LELELE has been well studied in the literature, only a few attempts have been made to address the LELE-EC layout decomposition problem. We propose a comprehensive study for LELE-EC layout decomposition. Layout graph and end-cut graph are constructed to extract all the geometrical relationships of both input layout and end-cut candidates. Based on these graphs, integer linear programming is formulated to minimize the conflict and the stitch numbers. The experimental results demonstrate the effectiveness of the proposed algorithms. *© 2015 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JMM.14.1.011002]

Keywords: triple/multiple patterning; end-cutting; layout decomposition; integer linear programming.

Paper 14072SSP received May 8, 2014; revised manuscript received Jul. 2, 2014; accepted for publication Jul. 17, 2014; published online Nov. 4, 2014.

1 Introduction

As the semiconductor process scales down to the 14-nm logic node and beyond, the industry encounters many lithography-related issues. Triple patterning lithography (TPL)[1] is one of the most promising candidates for next-generation lithography techniques, along with extreme ultraviolet lithography, electron beam lithography, directed self-assembly, and nanoimprint lithography.[2,3] However, all these new techniques are challenged by some problems, such as tremendous technical barriers, or low throughput. Therefore, TPL has recently earned more attention from both industry and academia.

One conventional process of TPL, so-called LELELE, has the same principle as litho-etch-litho-etch(LELE)-type double patterning lithography (DPL). Here, "L" and "E" represent one lithography process and one etch process, respectively. Although the LELELE process has been widely studied by industry and academia, two issues are derived. On one side, there are some native conflicts in LELELE, such as the 4-clique conflict.[4] For example, Fig. 1 illustrates a 4-clique conflict among features a, b, c, and d. No matter how the colors are assigned, there is at least one conflict. Since this 4-clique structure is common in advanced standard cell design, LELELE type TPL still suffers from this native conflict problem. On the other side, compared with LELE type double patterning, there are more serious overlapping problems in LELELE.[5]

To overcome all these limitations of LELELE, Lin[6] recently proposed a new TPL manufacturing process, called LELE-end-cutting (LELE-EC). As a TPL, this new manufacturing process contains three mask steps, namely the first, second, and trim masks. Figure 2 illustrates an example of the LELE-EC process. To generate target features in Fig. 2(a), the first and second masks are used for pitch splitting, which is similar to the LELE type DPL process. These two masks are shown in Fig. 2(b). Finally, a trim mask is applied to trim out the desired region as in Fig. 2(c). In other words, the trim mask is used to generate some end-cuts to further split feature patterns. Although the target features are not LELELE-friendly, they are LELE-EC process friendly so that with LELE-EC the features can be decomposed without any conflict. In addition, if all cuts are properly designed or distributed, LELE-EC can introduce a better printability than a conventional LELELE process.[6]

For a design with four short features, Figs. 3 and 4 present its simulated images using LELELE and LELE-EC processes, respectively. The lithography simulations are computed based on the partially coherent imaging system, where the 193-nm illumination source is modeled as a kernel matrix given by Ref. 7. To model the photoresist effect with the exposed light intensity, we use the constant threshold model with a threshold of 0.225. We can make several observations from these simulated images. First, there are some round-offs around the line ends [see Fig. 3(c)]. Second, to reduce the round-off issues as illustrated in Fig. 4(b), in the LELE-EC process short lines can be merged into longer lines, then the trim mask is used to cut off some spaces. Note that there might be some corner roundings due to the edge shorting of trim mask patterns. However, since line shortening or rounding is a strong function of the linewidth,[8] we observe that trim mask patterns can usually be much longer than the line-end width; thus, we assume that the rounding caused by the trim mask is insignificant. This assumption is demonstrated in Fig. 4(c).

Much research has been carried out to solve the corresponding design problems for an LELELE type TPL. The

*Address all correspondence to: Bei Yu, E-mail: bei@cerc.utexas.edu

The preliminary version of this work appeared in Proc. SPIE. Design for Manufacturability through Design-Process Integration VII.

0091-3286/2015/$25.00 © 2015 SPIE

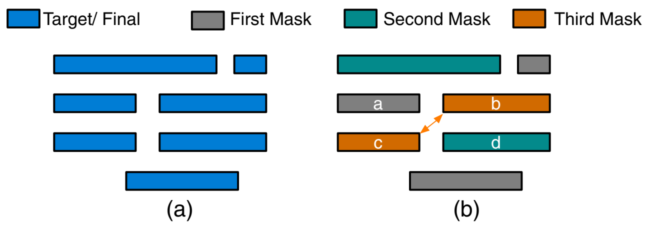

Fig. 1 Process of LELELE type triple patterning lithography (a) target features, (b) layout decomposition with one conflict introduced.

layout decomposition problem has been well studied.[4,9–15] In addition, the related constraints have been considered in early physical design stages, such as routing,[16,17] standard cell design,[18,19] and detailed placement.[19] However, only a few attempts have been made to address the LELE-EC layout decomposition problem. Note that although the trim mask can bring about better printability, it does introduce more design challenges, especially in the layout decomposition stage.

In this paper, we propose a comprehensive study for LELE-EC layout decomposition. Given a layout which is specified by features in polygonal shapes, we extract the geometrical relationships and construct the layout graph. Furthermore, the compatibility of all end-cuts candidates is also modeled in the end-cut graph. Based on the graphs, integer linear programming (ILP) is formulated to assign each vertex into one layer. Our goal in the layout decomposition is to minimize the conflict number, and at the same time minimize the overlapping errors.

The rest of the paper is organized as follows. Section 2 provides some preliminaries and discusses the problem formulation. Section 3 provides the overall flow of our layout decomposer. Section 4 explains the details of end-cut candidate generation. Section 5 presents a set of algorithms to solve the layout decomposition problem. Section 6 discusses

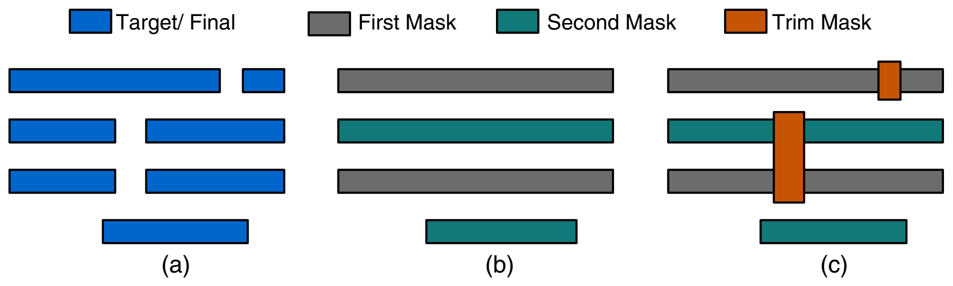

Fig. 2 Process of LELE-EC type triple patterning lithography (a) target features, (b) first and second mask patterns, (c) trim mask, and final decomposition without conflict.

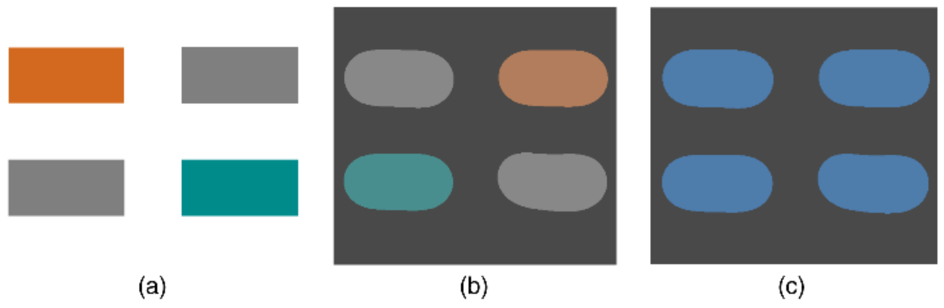

Fig. 3 LELELE process example. (a) Decomposed result, (b) simulated images for different masks, (c) combined simulated image as the final printed patterns.

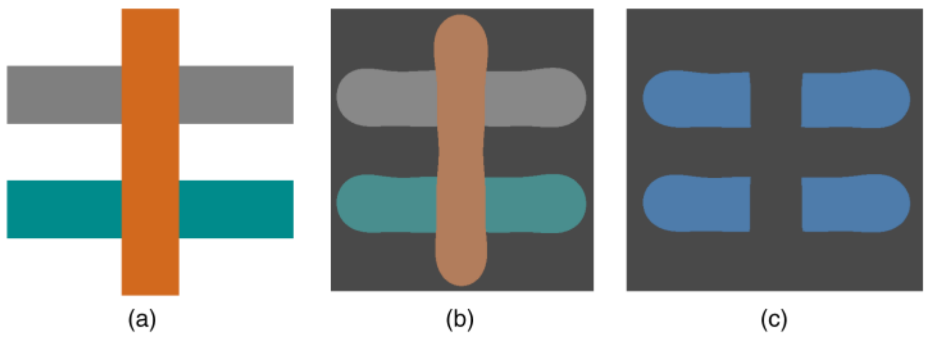

Fig. 4 LELE-EC process example. (a) Decomposed result, (b) simulated images for different masks, where orange pattern is trim mask, (c) combined simulated image as the final printed patterns.

several speed-up techniques. Section 7 presents the experimental results, followed by conclusions in Sec. 8.

2 Preliminary and Problem Formulation

2.1 *Layout Graph*

Given a layout which is specified by features in polygonal shapes, a layout graph[4] is constructed. As shown in Fig. 5, the layout graph is an undirected graph with a set of vertices V and a set of conflict edges (CEs). Each vertex in V represents one input feature. There is an edge in CE if and only if two features are within a minimum coloring distance dis_m of each other. In other words, each edge in CE is a conflict candidate. Figure 5(a) shows one input layout with the corresponding layout graph in Fig. 5(b). Here, the vertex set $V = \{1,2,3,4,5,6,7\}$, while the conflict edge set $\text{CE} = \{(1,2),(1,3),(1,4),(2,4),(3,4),(3,5),(3,6),(4,5),(4,6),(5,6),(5,7),(6,7)\}$. For each edge (conflict candidate), we check whether there is an end-cut candidate. For each end-cut candidate $i - j$, if it is applied, then features i and j will be merged into one feature. In this way, the corresponding CE can be removed. If the stitch is considered in the layout decomposition, some vertices in the layout graph can be split into several segments. The segments in one layout graph vertex are connected through "stitch edges". All these stitch edges are included in a set, called SE (Please refer to Ref. 20 for the details of stitch candidate generation.).

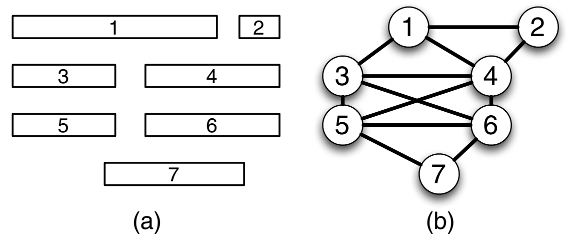

Fig. 5 Layout graph construction. (a) Input layout, (b) layout graph with conflict edges.

2.2 *End-Cut Graph*

Since all the end-cuts are manufactured with one single exposure process, they should be distributed far away from each other. That is, two end-cuts have conflict if they are within the minimum end-cut distance dis_c of each other. Note that these conflict relationships among end-cuts are not available in the layout graph; therefore, we construct an end-cut graph to store the relationships. Figure 6(a) gives an input layout example, with all end-cut candidates pointed out in Fig. 6(b). The corresponding end-cut graph is shown in Fig. 6(c). Each vertex in the end-cut graph represents one end-cut. There is a solid edge if and only if the two end-cuts conflict with each other. There is a dashed edge if and only if they are close to each other and they can be merged into one larger end-cut.

2.3 *Problem Formulation*

Here, we give the problem formulation of layout decomposition for triple patterning with end-cutting (LELE-EC).

Problem 1 (LELE-EC Layout Decomposition) *Given a layout which is specified by features in polygonal shapes, the layout graph and the end-cut graph are constructed. The LELE-EC layout decomposition assigns all vertices in the layout graph into one of two colors, and selects a set of end-cuts in the end-cut graph. The objectives is to minimize the number of conflicts and/or stitches.*

With the end-cut candidates generated, the LELE-EC layout decomposition is more complicated since more constraints are derived. Even though there is no end-cut candidate, LELE-EC layout decomposition is similar to the LELE type DPL layout decomposition. Sun et al. in Ref. 21 showed that LELE layout decomposition with minimum conflict and minimum stitch is NP-hard, and thus it is not hard to see that the LELE-EC layout decomposition is NP-hard as well. An NP-hard problem is a set of computational search problems that are difficult to solve.[22] No NP-hard problem can be solved in polynomial time in the worst case under the assumption that $\text{P} = \text{NP}$.

3 Overall Flow

The overall flow of our layout decomposer is illustrated in Fig. 7. First, we generate all end-cut candidates to find out all possible end-cuts. Then, we construct the layout and the end-cut graphs to transform the original geometric problem into a graph problem, and thus the LELE-EC layout decomposition can be modeled as a coloring problem in the layout graph and

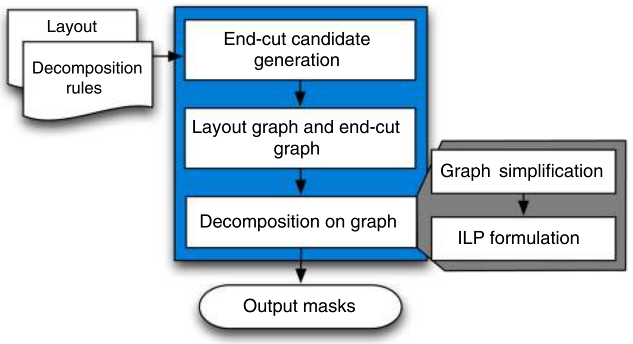

Fig. 7 Overall flow of our layout decomposer.

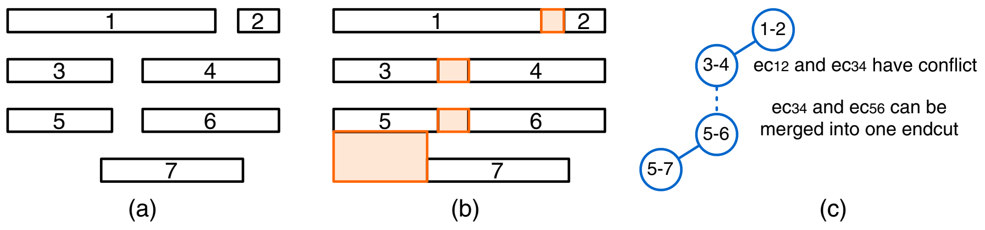
1
2
3
4
5
6
7
(a)
1
2
3
4
5
6
7
(b)
1-2
3-4
5-6
5-7
ec12 and ec34 have conflict
ec34 and ec56 can be merged into one endcut
(c)

Fig. 6 End-cut graph construction. (a) Input layout, (b) generated end-cut candidates, (c) end-cut graph.

the end-cut selection problem in the end-cut graph. Both the coloring and the end-cut selection problems can be solved through one ILP formulation. Since the ILP formulation may suffer from being a runtime overhead problem, we propose a set of graph simplification techniques. Besides, to further reduce the problem size, some end-cut candidates are preselected before ILP formulation. All the steps in the flow are detailed in the following sections.

4 End-Cut Candidate Generation

In this section, we will explain the details of our algorithm to generate all end-cut candidates. An end-cut candidate is generated between two conflicting polygonal shapes. It should be stressed that compared with the end-cut generation in Ref. 23, our methodology has the following two differences.

- An end-cut can be a collection of multiple end-cut boxes depending on the corresponding shapes. For instance, two end-cut boxes (ecb_1 and ecb_2) need to be generated between shapes S_1 and S_2, as shown in Fig 8. We propose a shape-edge-dependent algorithm to generate the end-cuts with multiple end-cut boxes.
- We consider the overlapping and variations caused by end-cuts. Two lithography simulations are illustrated in Figs. 9 and 10, respectively. In Fig. 9, we find some bad patterns or hotspots due to the cuts between two long edges. In Fig. 10, we can see that the final patterns are in better shape. Therefore, to reduce the manufacturing hotspot from the trim mask, during end-cut candidate generation we avoid the cuts along the two long edges.

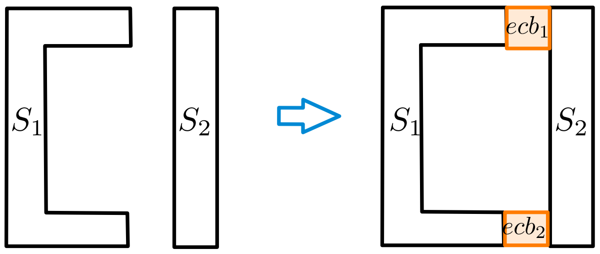

Fig. 8 An end-cut can have multiple end-cut boxes.

Algorithm 1 presents the key steps of generating an end-cut between two polygonal shapes S_1 and S_2. A polygonal shape consists of multiple edges. For each of the shape-edge-pair, one taken from S_1 and another from S_2, the possibility of the generation of an end-cut box (ecBox) is explored and we store all such end-cut boxes in ecBoxSet (Lines 2–9). The function "generateEndCutBox (se_1, se_2)" generates an end-cut box ecBox between the shape edges se_1 and se_2.

Figure 11 shows how end-cut boxes are generated between two shape edges under different situations. In Fig. 11(a), the end-cut box is between two shape edges, which are oriented in same direction and overlap in their x-coordinates. This type of end-cut box is called an edge-edge end-cut box. For Fig. 11(b), the shape edges are in same direction but do not overlap. In Fig. 11(c), the shape edges are oriented in different directions. The end-cut boxes generated in these two cases are corner-corner end-cut boxes. No end-cut box is generated in the case of Fig. 11(d). In addition, end-cut boxes are not generated for the following cases: (1) the end-cut box overlaps with any existing polygonal shape in the layout, (2) the height (h) or width (w) of the box is not within some specified range, i.e., when h, w do not obey the following constraints $h_{\text{low}} \le h \le h_{\text{high}}$ and $w_{\text{low}} \le w \le w_{\text{high}}$.

Then all generated end-cut boxes between two shapes are divided into independent components IC (Line 10) based on finding connected components of a graph $G = (V, E)$ with $V = \{v_i\} =$ the set of all end-cut boxes and $(v_i, v_j) \in E$, if v_i overlaps v_j. The overlap between two end-cut boxes is classified into type_1 and type_2 overlaps. When two boxes

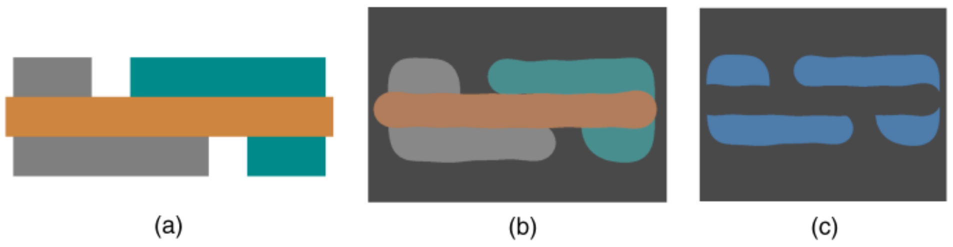

Fig. 9 (a) Decomposition example where cuts are along long edges, (b) simulated images for different masks, (c) combined simulated image with some hotspots.

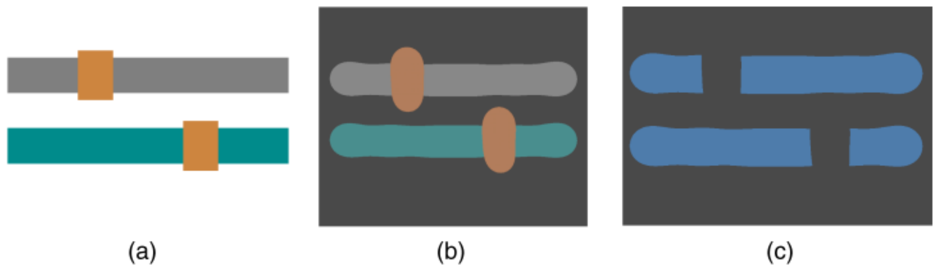

Fig. 10 (a) Decomposition example where cuts are between line ends, (b) simulated images for different masks, (c) combined simulated image with good printability.

Algorithm 1 Shape-edge dependent end-cut generation algorithm between two shapes S_1 and S_2

```
1: Procedure generateEndCut ($S_1$, $S_2$);
2: for all $se_1 \in$ edges($S_1$) do
3:     for all $se_2 \in$ edges($S_2$) do
4:         ecBox = generateEndCutBox ($se_1$, $se_2$);
5:         if ecBox $\neq$ NULL then
6:             Store ecBox in ecBoxSet;
7:         end if
8:     end for
9: end for
10: Divide ecBoxSet into independent components (IC);
11: if $|ecBoxSet| = |V|$ then
12:     Print all boxes;
13:     return true;
14: end if
15: for all $ic \in IC$ do
16:     Remove corner-corner end-cut boxes overlapping with
        edge-edge end-cut box;
17:     if $\exists$ set of $type_2$ overlaps then
18:         Generate minimum area box;
19:     else
20:         Generate all end-cut boxes
21:     end if
22: end for
23: return true;
24: end Procedure
```

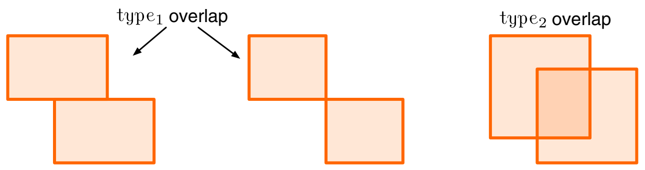

Fig. 12 Types of overlaps between end-cut boxes.

overlap only in an edge or in a point but not in space, we call this a type_1 overlap, whereas the overlap in space is termed as a type_2 overlap as shown in Fig. 12. Each of the ic $\in$ IC may contain multiple end-cut boxes. If the total number of end-cut boxes ($|V|$) is equal to $|\mathrm{IC}|$, that implies there is no overlap between the end-cut boxes and we generate all of them (Lines 11–14).

For multiple boxes in each IC, if there is an overlap between corner-corner and edge-edge end-cut boxes, the corner-corner end-cut box is removed (Line 16). After doing this, either there will be a set of type_1 overlaps or a set of type_2 overlaps in each IC. In the case of type_2 overlaps, the end-cut box with the minimum area is chosen as shown in Fig. 13. For type_1 overlaps in each IC, all end-cut boxes are generated (Line 20).

5 Integer Linear Programming Formulations

After the construction of layout and end-cut graphs, the LELE-EC layout decomposition problem can be transformed to a coloring problem on a layout graph and a selection problem on the end-cut graph. At the first glance, the coloring problem is similar to that in the LELE layout decomposition. However, since the conflict graph cannot be guaranteed to be planar, the face graph-based methodology[24] cannot be applied here. Therefore, we formulate ILP to simultaneously solve both coloring and selection problems. For convenience, some notations in the ILP formulation are listed in Table 1.

5.1 *Integer Linear Programming Formulation Without Stitch*

In this subsection, we discuss the ILP formulation when no stitch candidate is generated in the layout graph. Given a set of input layout features $\{r_1, \ldots, r_n\}$, we construct layout and end-cut graphs. Every conflict edge e_{ij} is in CE, while each end-cut candidate ec_{ij} is in SE. x_i is a binary variable representing the color of r_i. c_{ij} is a binary variable for conflict edge $e_{ij} \in \mathrm{CE}$. To minimize the conflict number, our objective function is to minimize $\sum_{e_{ij}\in \mathrm{CE}} c_{ij}$.

To evaluate the conflict number, we provide the following constraints:

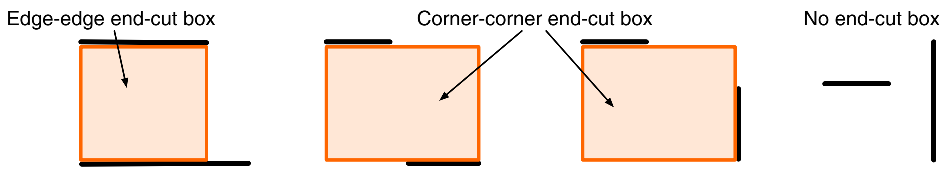

Fig. 11 End-cut box generation between any two shape edges.

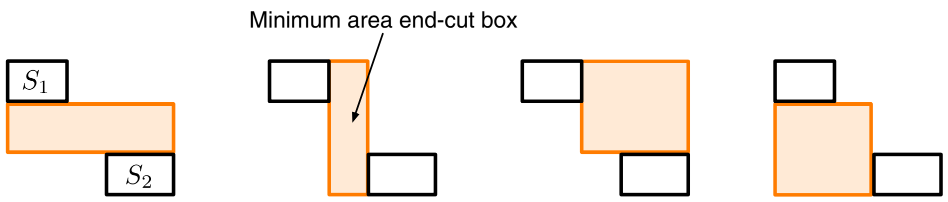

Fig. 13 Minimum area end-cut box is chosen for type$_2$ overlaps.

$$\begin{cases} x_i + x_j \le 1 + c_{ij} + \mathrm{ec}_{ij} & \text{if } \exists\ \mathrm{ec}_{ij} \in \mathrm{EE} \\ (1-x_i) + (1-x_j) \le 1 + c_{ij} + \mathrm{ec}_{ij} & \text{if } \exists\ \mathrm{ec}_{ij} \in \mathrm{EE} \\ x_i + x_j \le 1 + c_{ij} & \text{if } \nexists\ \mathrm{ec}_{ij} \in \mathrm{EE} \\ (1-x_i) + (1-x_j) \le 1 + c_{ij} & \text{if } \nexists\ \mathrm{ec}_{ij} \in \mathrm{EE} \end{cases}. \tag{1}$$

Here, ec_{ij} is a binary variable for an end-cut candidate. If there is no end-cut candidate between adjacent features r_i and r_j, if $x_i \neq x_j$ then one conflict would be reported ($c_{ij} = 1$). Otherwise, we will try to enable the end-cut candidate ec_{ij} first. If the end-cut candidate ec_{ij} cannot be applied ($\mathrm{ec}_{ij} = 0$), then one conflict will be also reported.

If end-cuts ec_{ij} and ec_{pq} are in conflict with each other, at most one of them will be applied. To enable this, we introduce the following constraint:

$$\mathrm{ec}_{ij} + \mathrm{ec}_{pq} \le 1, \quad \forall\ e_{ijpq} \in \mathrm{EE}. \tag{2}$$

To forbid a useless end-cut, we introduce the following constraints. That is, if features x_i and x_j are in different colors, $\mathrm{ec}_{ij} = 0$.

$$\begin{cases} \mathrm{ec}_{ij} + x_i - x_j \le 1 & \forall\ e_{ij} \in \mathrm{CE} \\ \mathrm{ec}_{ij} + x_j - x_i \le 1 & \forall\ e_{ij} \in \mathrm{CE} \end{cases}. \tag{3}$$

Therefore, without the stitch candidate, the LELE-EC layout decomposition can be formulated as shown in Eq. (4)

$$\min \sum_{e_{ij} \in \mathrm{CE}} c_{ij} \quad \text{s.t. } (1), (2), (3). \tag{4}$$

Table 1 Notations in LELE-EC layout decomposition.

CE	Set of conflict edges
EE	Set of end-cut conflict edges
SE	Set of stitch edges
n	Number of input layout features
r_i	The i-th layout feature
x_i	Variable denoting the coloring of r_i
ec_{ij}	0–1 variable, $\mathrm{ec}_{ij} = 1$ when a end-cut between r_i and r_j
c_{ij}	0–1 variable, $c_{ij} = 1$ when a conflict between r_i and r_j
s_{ij}	0–1 variable, $s_{ij} = 1$ when a stitch between r_i and r_j

5.2 *Integer Linear Programming Formulation With Stitch*

If the stitch insertion is considered, the ILP formulation is as in Eq. (5). Here, the objective is to simultaneously minimize both the conflict and the stitch numbers. The parameter α is a user-defined parameter for assigning relative importance between the conflict and the stitch numbers. The constraints (6) and (7) are used to calculate the stitch number.

$$\min \sum_{e_{ij} \in \mathrm{CE}} c_{ij} + \alpha \times \sum_{e_{ij} \in \mathrm{SE}} s_{ij}, \tag{5}$$

$$\text{s.t. } x_i - x_j \le s_{ij} \quad \forall\ e_{ij} \in \mathrm{SE}, \tag{6}$$

$$x_j - x_i \le s_{ij} \quad \forall\ e_{ij} \in \mathrm{SE}. \quad (1), (2), (3) \tag{7}$$

6 Graph Simplification Techniques

ILP is a classical NP-hard problem, i.e., there is no polynomial time optimal algorithm to solve it.[22] Therefore, for large layout cases, the solving of ILP may suffer from a long runtime penalty to achieve the desired results. In this section, we provide a set of speed-up techniques. Note that these techniques can keep optimality. In other words, with these speed-up techniques, ILP formulation can achieve the same results as compared to those that do not apply speed-up.

6.1 *Independent Component Computation*

The first speed-up technique is the so-called independent component computation. By breaking down the whole layout graph into several independent components, we partition the initial layout graph into several small ones. Then, each component can be independently resolved through ILP formulation. At last, the overall solution can be taken as the union of all the components without affecting the global optimality. Note that this is a well-known technique which has been applied in many previous studies (e.g., Refs. 20, 25, and 26).

6.2 *Bridge Computation*

A bridge of a graph is an edge whose removal disconnects the graph into two components. If the two components are independent, removing the bridge can divide the whole problem into two independent subproblems. We search for all bridge edges in the layout graph, then divide the whole layout graph through these bridges. Note that all bridges can be found in $O(|V| + |E|)$, where $|V|$ is the vertex number and $|E|$ is the edge number in the layout graph.

Table 2 Comparison between with stitch and without stitch.

			ILP without stitch				ILP with stitch			
Circuit	Wire#	Comp#	Conflict#	Stitch#	Cost	CPU(s)	Conflict#	Stitch#	Cost	CPU(s)
C1	1109	123	1	0	1	1.19	1	0	1	1.32
C2	2216	175	1	0	1	2.17	1	0	1	2.89
C3	2411	270	0	0	0	3.11	0	0	0	3.62
C4	3262	467	0	0	0	3.2	0	0	0	3.75
C5	5125	507	4	0	4	8.72	4	0	4	8.81
C6	7933	614	1	0	1	11.24	1	0	1	11.1
C7	10189	827	2	0	2	14.57	2	0	2	15.98
C8	14603	1154	2	0	2	21.2	2	0	2	23.07
C9	14575	2325	23	0	23	24.36	12	12	13.2	28.06
C10	21253	1783	7	0	7	28.42	7	0	7	32.02
S1	4611	272	0	0	0	6.23	0	0	0	7.04
S2	67696	5116	166	0	166	179.05	166	1	166.1	218.37
S3	157455	15176	543	0	543	506.55	530	13	531.3	563.65
S4	168319	15354	443	0	443	464.84	436	7	436.7	494.4
S5	159952	12626	419	0	419	464.11	415	6	415.6	514.56
Average	—	—	107.5	0	107.5	115.9	105.1	2.6	105.4	128.6
Ratio	—	—	—	—	**1.0**	**1.0**	—	—	**0.98**	**1.10**

Note: the bold values are used to highlight the comparison between two methodologies.

6.3 *End-Cut Preselection*

Some end-cut candidates have no conflict end-cuts. For the end-cut candidate ec_{ij} that has no conflict end-cut, it would be preselected in the final decomposition results. That is, the features r_i and r_j are merged into one feature. In this way, the problem size of the ILP formulation can be further reduced. End-cut preselection can be finished in linear time.

7 Experimental Results

We implement our algorithms in C++ and test on an Intel Xeon 3.0 GHz Linux machine with 32G RAM. In total, 15 benchmark circuits from Ref. 4 are used. GUROBI[27] is chosen as the ILP solver. The minimum coloring spacing $\min_s$ is set as 120 for the first 10 cases and as 100 for the last five cases, as in Refs. 4 and 10. The width threshold w_{th}, which is used in end-cut candidate generation, is set as dis_m.

7.1 *With Stitch or Without Stitch*

In the first experiment, we show the decomposition results of the ILP formulation. Table 2 compares two ILP formulations "**ILP w/o. stitch**" and "**ILP w. stitch**." Here "ILP w/o. stitch" is the ILP formulation based on the graph without SEs, while "ILP w. stitch" considers the stitch insertion in the ILP. Note that all speed-up techniques are applied to both. Columns "Wire#" and "Comp#" report the total feature number and the divided component number, respectively.

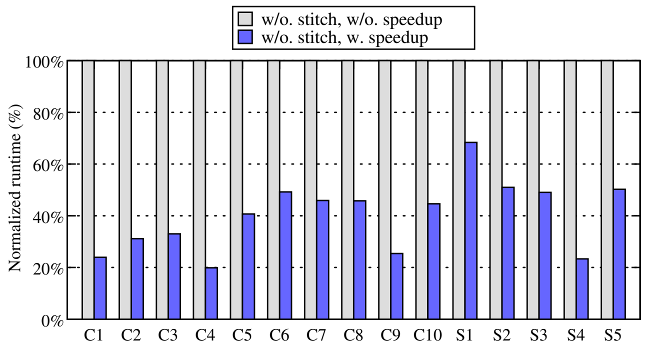

Fig. 14 Effectiveness of speed-up techniques when no stitch is introduced.

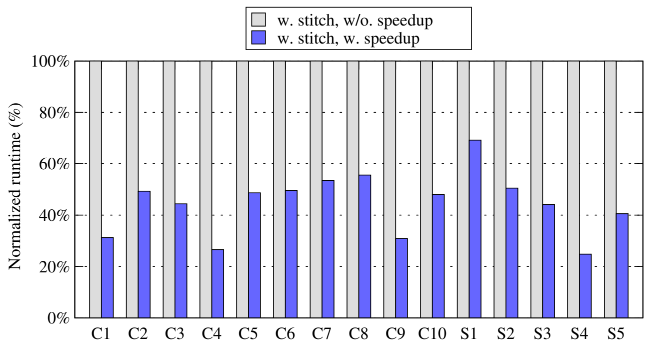

Fig. 15 Effectiveness of speed-up techniques when stitch is introduced.

For each method we report the conflict number, stitch number, and computational time in seconds("CPU(s)"). "Cost" is the cost function, which is set as conflict# $+0.1\times$ stitch#.

From Table 2, we can see that compared with "ILP w/o. stitch," when stitch candidates are considered in the ILP formulation, the cost can be reduced by 2%, while the runtime is increased by 5%. Note that stitch insertion has been known to be an effective method to reduce the cost for both LELE and LELELE layout decompositions. However, we can see that for LELE-EC layout decomposition, stitch insertion is not that effective. In addition, due to the overlap variation derived from stitch, stitch insertion for LELE-EC may not be an effective method.

7.2 *Effectiveness of Speed-up Techniques*

In the second experiment, we analyze the effectiveness of the proposed speed-up techniques. Figure 14 compares two ILP formulations "**w/o. stitch w/o. speedup**" and "**w/o. stitch w. speedup**," where "w/o. stitch w/o. speedup" only applies an independent component computation, while "w. speedup" involves all three speed-up techniques. However, none of them consider the stitch in the layout graph. From Fig. 14, we can see that with speed-up techniques (bridge computation and end-cut preselection), the runtime can be reduced by around 60%.

Figure 15 demonstrates the similar effectiveness of speed-up techniques between "**w. stitch w. speedup**" and "**w/o. stitch w. speedup**." Here, stitch candidates are introduced in the layout graph. We can see that for these two ILP formulations, the bridge computation and the end-cut preselection can reduce the runtime by around 56%.

7.3 *Conflict Analysis*

Figure 16 shows four conflict examples in a decomposed layout, where conflict pairs are labeled with red arrows. We can observe that some conflicts [see Figs. 16(a) and 16(c)] are introduced due to the end-cuts existing in neighboring. For these two cases, the possible reason is that the patterns are irregular; therefore, some end-cuts that are close to each other cannot be merged into a larger one. We can also observe some conflicts [see Figs. 16(b) and 16(d)] come from via shapes. For these two cases, one possible reason is that it is hard to find end-cut candidates around via, compared to long wires.

8 Conclusions

In this paper, we have proposed an improved framework and the algorithms to solve the LELE-EC layout decomposition problem. New end-cut candidates are generated considering potential hotspots. The layout decomposition is formulated as an ILP. The experimental results show the effectiveness of our algorithms. Note that our framework is very generic so that it can provide high quality solutions to both uni-directional and bi-directional layout patterns. However, if all the layout patterns are uni-directional, there might be

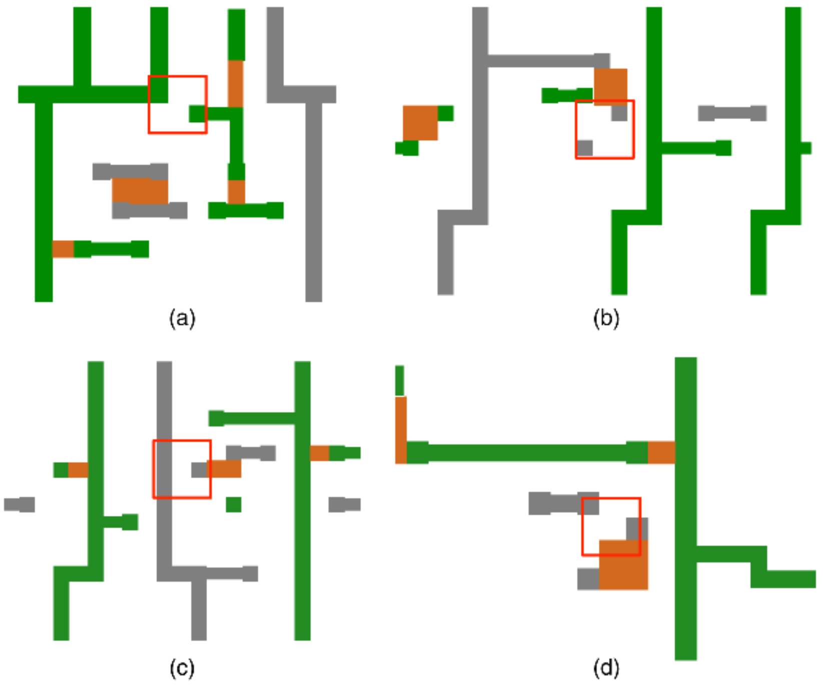

Fig. 16 (a) and (c) Conflicts because no additional end-cut can be inserted due to the existing neighboring end-cuts. (b) and (d) Conflicts because no end-cut candidates between irregular vias.

some faster solutions. Since end-cutting can provide a better printability than a traditional LELELE process, we expect to see more works on the LELE-EC layout decomposition and LELE-EC aware design.

References

1. K. Lucas et al., "Implications of triple patterning for 14 nm node design and patterning," *Proc. SPIE*, **8327**, 832703 (2012).
2. D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **32**(10), 1453–1472 (2013).
3. B. Yu et al., "Dealing with IC manufacturability in extreme scaling," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 240–242, ACM, San Jose, California (2012).
4. B. Yu et al., "Layout decomposition for triple patterning lithography," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 1–8, ACM, San Jose, California (2011).
5. C. Ausschnitt and P. Dasari, "Multi-patterning overlay control," *Proc. SPIE*, **6924**, 692448 (2008).
6. B. J. Lin, "Lithography till the end of Moore's law," in *Proc. ACM Int. Symp. on Physical Design (ISPD)*, pp. 1–2, ACM, Napa, California (2012).
7. S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 271–274, ACM, San Jose, California (2013).
8. C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*, John Wiley & Sons, Hoboken, New Jersey (2008).
9. C. Cork, J.-C. Madre, and L. Barnes, "Comparison of triple-patterning decomposition algorithms using aperiodic tiling patterns," *Proc. SPIE*, **7028**, 702839 (2008).
10. S.-Y. Fang, W.-Y. Chen, and Y.-W. Chang, "A novel layout decomposition algorithm for triple patterning lithography," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, pp. 1185–1190, ACM, San Francisco, California (2012).
11. H. Tian et al., "A polynomial time triple patterning algorithm for cell based row-structure layout," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 57–64, ACM, San Jose, California (2012).
12. J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, pp. 1–69, ACM, San Francisco, California (2013).
13. B. Yu et al., "A high-performance triple patterning layout decomposer with balanced density," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 163–169, ACM, San Jose, California (2013).
14. Y. Zhang et al., "Layout decomposition with pairwise coloring for multiple patterning lithography," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 170–177, ACM, San Jose, California (2013).
15. B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, pp. 1–6, ACM, San Francisco, California (2014).
16. Q. Ma, H. Zhang, and M. D. F. Wong, "Triple patterning aware routing and its comparison with double patterning aware routing in 14 nm technology," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, pp. 591–596, ACM, San Francisco, California (2012).
17. Y.-H. Lin et al., "TRIAD: a triple patterning lithography aware detailed router," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 123–129, ACM, San Jose, California (2012).
18. H. Tian et al., "Constrained pattern assignment for standard cell based triple patterning lithography," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 57–64, ACM, San Jose, California (2013).
19. B. Yu et al., "Methodology for standard cell compliance and detailed placement for triple patterning lithography," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 349–356, ACM, San Jose, California (2013).
20. A. B. Kahng et al., "Layout decomposition for double patterning lithography," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 465–472, ACM, San Jose, California (2008).
21. J. Sun et al., "Post-routing layer assignment for double patterning," in *Proc. IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*, pp. 793–798, IEEE Press, Yokohama, Japan (2011).
22. T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts (1990).
23. B. Yu, J.-R. Gao, and D. Z. Pan, "Triple patterning lithography (TPL) layout decomposition using end-cutting," *Proc. SPIE* **8684**, 86840G (2013).
24. Y. Xu and C. Chu, "A matching based decomposer for double patterning lithography," in *Proc. ACM Int. Symp. on Physical Design (ISPD)*, pp. 121–126, ACM, San Francisco, California (2010).
25. K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *Proc. ACM Int. Symp. on Physical Design (ISPD)*, pp. 107–114, ACM, San Diego, California (2009).
26. J.-S. Yang et al., "A new graph-theoretic, multi-objective layout decomposition framework for double patterning lithography," in *Proc. IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*, pp. 637–644, IEEE Press, Taipei, Taiwan (2010).
27. Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com (2014).

Bei Yu is a postdoctoral scholar in the Department of Electrical and Computer Engineering, University of Texas at Austin, where he received his PhD degree in 2014. His research interests include design for manufacturability and optimization algorithms with applications in CAD. He received the SPIE Education Scholarship in 2013, IBM PhD Scholarship in 2012, Best Paper Awards at ICCAD'13 and ASPDAC'12, and three other Best Paper Award nominations.

Subhendu Roy is a PhD student in the Electrical and Computer Engineering Department in the University of Texas at Austin. His research interests include the area of design automation for logic synthesis, physical design, and cross-layer reliability. He has worked in Atrenta as a full-time employee, and in IBM T J Watson Research Center and Mentor Graphics as a summer intern. He received the Best Paper Award in ISPD'14.

Jhih-Rong Gao is a senior technical staff in Cadence Design Systems, Inc., Austin. She received her PhD degree from the Department of Electrical and Computer Engineering, University of Texas at Austin in 2014. Her current research interests include physical design and design for manufacturability. She was a research and development engineer with Synopsys, Inc., Taiwan, from 2007 to 2009. She was awarded the BACUS Photomask Scholarship from SPIE in 2013.

David Z. Pan is a professor at the Department of Electrical and Computer Engineering, University of Texas at Austin. His research interests include nanometer IC design for manufacturability/reliability, new frontiers of physical design, and CAD for emerging technologies. He has published over 200 technical papers, and is the holder of eight U.S. patents. He has received many awards, including the SRC Technical Excellence Award and 11 Best Paper Awards, among others. He is an IEEE Fellow.