Journal of
Micro/Nanolithography,
MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Directed self-assembly cut mask assignment for unidirectional design

Jiaojiao Ou
Bei Yu
Jhih-Rong Gao
David Z. Pan

SPIE.

Directed self-assembly cut mask assignment for unidirectional design

Jiaojiao Ou,[a,*] Bei Yu,[a,b] Jhih-Rong Gao,[c] and David Z. Pan[a]
[a]University of Texas at Austin, Department of Electrical and Computer Engineering, Austin, Texas 78712, United States
[b]Chinese University of Hong Kong, Department of Computer Science and Engineering, Shatin, Hong Kong
[c]Cadence Design Systems, 12515-7 Research Boulevard, Austin, Texas 78759, United States

Abstract. Recently, directed self-assembly (DSA) has emerged as a promising lithography solution for cut manufacturing. We perform a comprehensive study on the DSA aware mask optimization problem to provide a DSA friendly design on cut layers. We first formulate the problem as an integer linear programming (ILP) to assign cuts to different guiding templates, targeting both conflict minimization and line-end extension minimization. As ILP may not be scalable for very large size problems, we then propose two speed-up strategies. The first one is to decompose the initial problem into smaller ones and solve them separately, followed by solution merging without much loss of quality. The second one is using the set cover algorithm to decide the DSA guiding pattern assignment, and then legalize the template placement. Our approaches can be naturally extended to handle arbitrary DSA guiding template patterns with complicated shapes. Experimental results show that our methodologies can significantly improve the DSA friendly, i.e., both the unresolved pattern number and the line-end extensions can be reduced. [DOI: 10.1117/1.JMM.14.3.031211]

Keywords: directed self-assembly; cut mask; linear programming.

Paper 15053SS received Apr. 15, 2015; accepted for publication Jul. 8, 2015; published online Aug. 7, 2015.

1 Introduction

As the feature size of semiconductor transistors has been further scaled down in emerging technology nodes, a unidirectional design has attracted more and more attention.[1] On one hand, unidirectional design can provide a large process window and improved manufacturability.[2] With line-cut technology, unidirectional design can also simplify the fabrication process. On the other hand, compared with conventional two-dimensional design where design rules are dramatically increased along with highly scaled technology nodes, unidirectional design enables an effective and simplified design flow.[3]

Due to the physical limitation of the conventional 193i lithography system, it is still challenging to print randomly distributed patterns for highly unidirectional layout.[4] For example, the 193i lithography system might generate a large cut across adjacent metal lines. The situation is even worse for some tip to tip layout patterns, where some extra lines are cut off, thus affecting the connectivity of the contact. Therefore, the semiconductor industry is looking for next generation lithography technologies, such as multiple patterning lithography,[5] E-beam lithography,[6] extreme ultra violet,[7] and directed self-assembly (DSA).

Recently, DSA has earned more and more attention due to its low cost and high-manufacturing throughput.[8,9] In a unidirectional design, DSA has demonstrated its potential to generate dense patterns for cut.[10] The actual size of a DSA cut is much smaller than a traditional cut since DSA can formulate cylinders inside the guiding template. Therefore, by assigning cuts to different guiding templates, it can still print cut patterns when the size of the targeted cut is beyond the limit of traditional lithography.

Much work has been done on the investigation of DSA contact layer fabrication and DSA aware physical design. In the fabrication stage, Yi et al.[11] demonstrated the fabrication of DSA contacts for unidirectional standard cells. They showed that different contacts can be formulated by adjusting the size and shape of the guiding templates. In the physical design stage, Du et al.[12] took DSA related constraints into account in a standard cell design. In addition, they proposed the assignment of a cost function to different DSA templates based on manufacturability, thereby developing a DSA aware routing method that considers both cost reduction and throughput improvement.[13]

How to integrate DSA awareness in the layout optimization stage is a critical problem, thus Xiao et al.[10] proposed a DSA cut guiding templates' redistribution method for a unidirectional design layout. However, there are some issues in their method which may reduce the solution quality. For example, the proposed method selects a large template to avoid an unmarked template. However, since it neglects the relative positions of these templates, which leads to a lot of overlapping conflicts, more wire extensions and template merging are required to remove these conflicts. In the worst case, a lot of conflicts cannot be resolved by just extending the metal lines or merging templates.

In this work, we perform a comprehensive investigation on the DSA aware cut mask optimization problem, where cuts on the line ends are assigned to different DSA guiding templates, targeting wire extension minimization and conflict number minimization. We first formulate the problem to an integer linear programming (ILP). Since for a large scale ILP problem the runtime would increase exponentially, we divide a large layout into several smaller pieces and solve them separately.

*Address all correspondence to: Jiaojiao Ou, E-mail: jiaojiao.ou@utexas.edu

1932-5150/2015/$25.00

In addition, we propose a set cover-based method to solve this problem more efficiently. Since DSA guiding templates may have different shapes, we then perform the experiments with an extended DSA guiding template set.

Our contributions in this work can be summarized as follows:

- We propose an ILP formulation for the DSA-based end-cutting problem, which assigns cuts to different DSA templates with minimum wire extensions and minimum conflict number;
- We present several speed-up methods: ILP speed-up method and set cover based method, to efficiently solve the problem;
- Our algorithms can be adapted to arbitrary types of DSA guiding template shapes, which will be beneficial to the development of DSA manufacturing technique.

The rest of the paper is organized as follows. Section 2 introduces the background of the DSA-based end-cutting process and the problem formulation. Section 3 proposes the ILP formulation and speed-up method. Section 4 shows the experimental results, and Sec. 5 concludes the paper.

2 Preliminary and Problem Formulation

2.1 *End-Cutting Process*

For a unidirectional layout, lines-cut is an effective approach for metal lines' connectivity implementation.[14–16] For a unidirectional layout of metal wires, in order to achieve the circuit connectivity, some parts of the lines should be blocked or cut when printing the unidirectional lines using a self-aligned double patterning or other lithography techniques. This cut process can be done by using a trim mask or cut mask. The cut mask can be optimized via an end-cutting approach [see Fig. 1(b)],[17,18] which means cutting the connectivity of the wire through fixed cuts rather than removing the entire section of unwanted wire. In either case, the logical connectivity of the circuit remains the same. By extending the line end of the wires, some conflicted layout patterns can be resolved for the end-cutting approach. For patterns which cannot be resolved by wire extension, they will be marked or be printed by an alternative lithography technique, such as e-beam,[19,20] or an additional 193i cut mask.[21] However, with the scaling down of metal pitch, traditional lithography may not be able to print a mask which is small enough to cut just one metal line. In addition, the problem worsens for tip to tip layout since the design rule is even more strict. These problems can be resolved by DSA end-cutting as metal lines are cut by the cylinders inside the template, as is depicted in Fig. 1(b).

2.2 *Directed Self-Assembly Guiding Template Type*

The shape and the size of DSA guiding templates can be adjusted to generate different DSA patterns.[22] However, DSA cylinder pitch variations will increase with the growth of its guiding template size and complexity. For some complex guiding templates, some unwanted cylinders can be generated inside the templates.[23] However, some regular DSA guiding templates [e.g., shown in Figs. 2(a)–2(e)] could have reasonably good variation controllability and manufacturability.[24] Therefore, in order to ensure the throughput and manufacturability of the DSA cut for metal lines, we will restrict the DSA patterns to the above regular patterns in this work.

Without increasing any fabrication difficulties, we assume that the two-hole template can be rotated to the horizontal direction and the peanut-shaped template can be rotated 90 deg. The final DSA guiding template set is shown in Fig. 2(f). Note that with the increased feasible guiding template set, it is possible to achieve a better cuts' assignment result.

2.3 *Problem Formulation*

The DSA-based cut mask optimization problem is defined as follows:

Problem 1 (DSA aware cut mask optimization). Given n unidirectional metal wires and m DSA guiding template candidates, we are going to assign each wire cut into one DSA guiding template. If applicable, one wire line end can be extended to fit one guiding template. The objective is to minimize the number of unpatternable cuts and the total wire extensions.

3 Algorithms

In this section, we first propose an ILP formulation to search for the optimal solution of the DSA aware cut mask optimization problem. It should be noted that the ILP formulation is adaptive to different DSA guiding template shapes. Since the ILP formulation is not scalable for a very large problem size, we then propose two speed-up methods to efficiently solve the problem.

3.1 *Integer Linear Programming Formulation*

3.1.1 *Objective*

Given a unidirectional layout, we are trying to minimize the wire extensions and the number of conflicts when applying

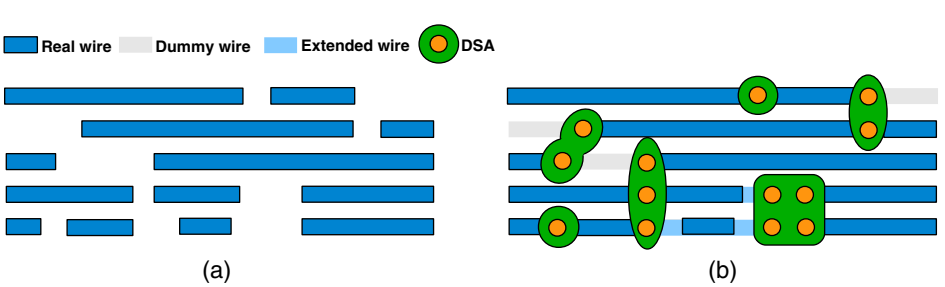

Fig. 1 (a) Target layout and (b) directed self-assembly (DSA) guiding template assignment.

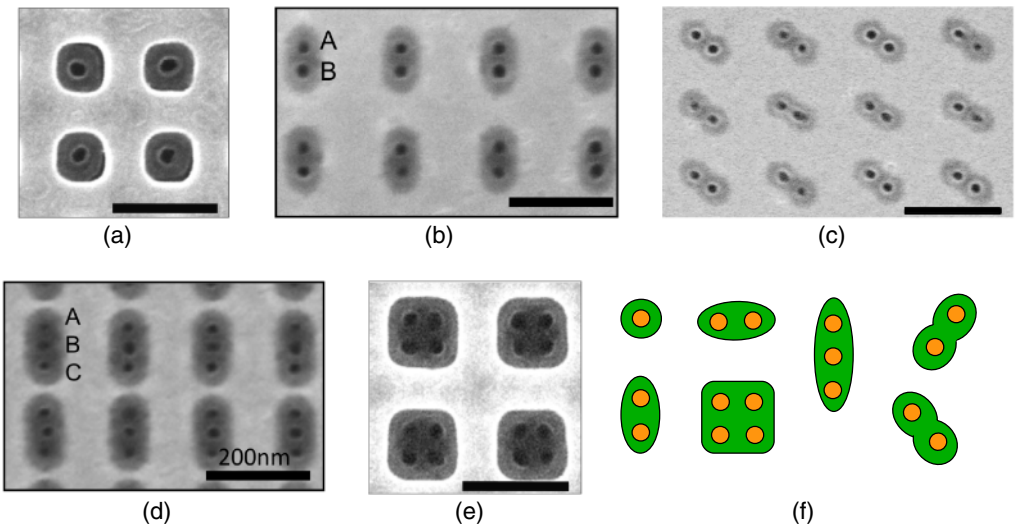

Fig. 2 Regular DSA guiding templates SEM images[24] and possible DSA guiding template set: (a) single-hole template, (b) two-hole template, (c) peanut-shaped template, (d) three-hole template, (e) four-hole template, (f) DSA guiding template set.

DSA guiding templates to the wire cuts. In addition, we also assign weights to different DSA guiding template types based on their manufacturability and variations. Thus, the total weight of the DSA guiding templates is also taken into account in the objective function. Because the guiding patterns are fixed and the metal wires can be extended, it is more difficult than a traditional end-cutting problem

$$\text{Minimize} \sum_{i=1}^{n}(x_{2i} - x_{2i-1} - l_i) + W\sum_{i=1}^{2n}\varepsilon_i + \sum_{i=1}^{m} w_i t_i,$$

where W is a constant which denotes the weight of the conflicted cut pattern. If a cut is marked as a conflict, we should either redesign the layout or adopt other complimentary lithography to remove the conflict. Therefore, in our implementation W is set to a very high value to effectively reduce the conflict number. ε_i is a binary variable indicating whether a conflict is introduced to cut i. That is, if cut i cannot be assigned to any DSA template, ε_i is set to 1. x_{2i-1} and x_{2i} are continuous variables indicating the left and right sides of wire i. They can also be considered as the horizontal positions for cuts $2i-1$ and $2i$. l_i is a constant which indicates the original length of wire i. w_i indicates the weight of template t_i.

3.1.2 *Constraints*

C1: line-end extension. The line ends of metal wires can only be extended to the left or right of the original design so as not to interrupt the circuit logical connectivity. However, the wires should not be extended outside the layout boundary. This constraint is similar to the one in a traditional end-cutting problem.[17]

$$\begin{cases} x_{2i-1} \le L_i \\ x_{2i} \ge R_i \\ x_{2i} - x_{2i-1} \le l_i + \sigma_i \quad i = 1,2,\ldots,n \\ x_{2i-1} \ge \text{LL} \\ x_{2i} \le \text{RR} \end{cases}$$

where σ_i indicates the wire extension limit for wire i. L_i and R_i are constant numbers indicating the original left and right x-coordinates of wire i. LL and RR are the left and the right bounds of the layout.

C2: constraints for templates. For the two line ends of a wire, we can consider them as two cuts. For each cut, it can be combined with other cuts to be fitted into one DSA guiding template pattern. There might be several template candidates for each cut, e.g., cut i can be printed by a single-hole template, or it can be combined with cut j and be printed with a two-hole template. Since each cut should be printed by only one template, the constraint is as follows:

$$\sum_{t_j \in T_i} t_j = 1, \quad \forall\, i \quad i = 1,2,\ldots,n \quad j = 1,2,\ldots,m.$$

T_i is the potential guiding template set for cut i. In other words, it includes all the possible DSA guiding templates to print cut i. t_j is a binary variable indicating one of the potential DSA guiding templates. For $t_j \in T_i$, if $t_j = 1$, cut i is printed by t_j. The sum of all potential templates for cut i should be 1. m indicates the total number of potential DSA guiding templates for cut i. The possible DSA guiding template assignments for an example layout are depicted in Fig. 3 (right). A conflict edge will be assigned to the line ends if they violate the design rule without any line extensions, as shown in Fig. 3 (left).

C3: minimum distance between adjacent templates. Because DSA guiding templates are printed by traditional 193i lithography, adjacent templates should be a minimum distance away from each other. If cut i and cut j are in different templates, without any loss of generality we use x_i and x_j to indicate the x-coordinate of the two cuts in adjacent templates.

1. Cuts on the same track

   If two cuts are on the same track, we assume that the initial cut i is on the right side of cut j, then x_i is always at least $\min_s$ larger than x_j. However, if the two

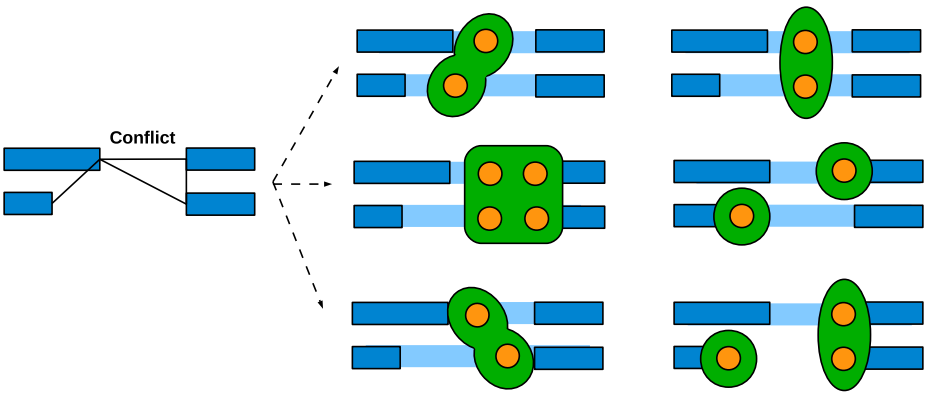

Fig. 3 Possible DSA guiding template assignments for the example layout.

cuts are in the same template t_k or one of the cuts is marked as conflict, then there is no such constraint.

$$x_i - x_j + B \times (\varepsilon_i + \varepsilon_j + \sum t_k) \geq \min_s,$$
$$t_k \in (T_i \cap T_j),$$

where B is a very large constant, and $\min_s$ is the minimum template spacing.

2. Cuts on adjacent track

If cut i and cut j are on adjacent tracks, since the line end can be extended, cut i can be on the left or right side of cut j.

$$\begin{cases} x_i - x_j + B \times \left[\varepsilon_i + \varepsilon_j + \sum\limits_{t_k \in (T_i \cap T_j)} t_k + d_i^j\right] \geq \min_s \\ x_j - x_i + B \times \left[\varepsilon_i + \varepsilon_j + \sum\limits_{t_k \in (T_i \cap T_j)} t_k + 1 - d_i^j\right] \geq \min_s \end{cases},$$

where d_i^j is a binary variable denoting the relative positions of cut i and cut j. If $d_i^j = 1$, x_i is on the left side of x_j. If $d_i^j = 0$, x_i is on the right side of x_j.

C4: constraints inside templates. For different DSA guiding patterns, the positions of the inside cuts are different. For two-hole and three-hole vertical templates, the cuts inside them should be vertically aligned; for a four-hole template, the horizontal cuts have a certain distance between them, and the vertical aligned cuts have the same horizontal coordinates. We can also assume the same conclusion for the two-hole diagonal templates and horizontal two-hole templates. If cut i and cut j are in the same template t_k:

$$\begin{cases} x_i - x_j + B \times (1 - t_k) \geq \theta_k \\ x_i - x_j - B \times (1 - t_k) \leq \theta_k \end{cases},$$

where θ_k indicates the required distance for cut i and j in template t_k.

3.2 *Speed-Up Method*

3.2.1 *Integer Linear Programming Speed-Up*

For a large scale size problem, the initial ILP formulation may be quite computationally intensive, due to the NP-hardness of the ILP problem. For the current biggest benchmark, the number of variables could be more than 90 k, and constraints could be more than 70 k.

We can divide the original layout into smaller ones and apply the cut mask optimization algorithm on them separately. The runtime for the entire problem is, therefore, reduced through the reduction of variables in each sublayout. However, some overlaps or violations may occur on the boundaries of each group, thus a template legalization will be performed to resolve these conflicts.

The overall flow of the speed-up method is illustrated in Fig. 4, which mainly consists of three stages: (1) layout division, (2) ILP optimization, and (3) templates' legalization. The ILP formulation is the same as the initial formulation. For layout division, it is noticed that some cuts are far away from their adjacent cuts, so there is low possibility that these cuts will conflict with each other. Thus, these adjacent cuts can be grouped together and then be independently solved by the ILP solver. However, if the number of variables in one group is smaller than a certain value, the total initialization time for the ILP solver might exceed the program runtime which has been saved. Therefore, the number of cuts in each group should be larger than a certain threshold value.

For the template legalization, we can remove these conflicts by merging adjacent templates into one or grouping them to form a new DSA guiding template. If the conflict cannot be resolved, we will mark it as a real conflict.

3.3 *Heuristic Speed-Up*

In this section, we will show that the DSA template assignment can be modeled as a set cover problem. Then a greedy

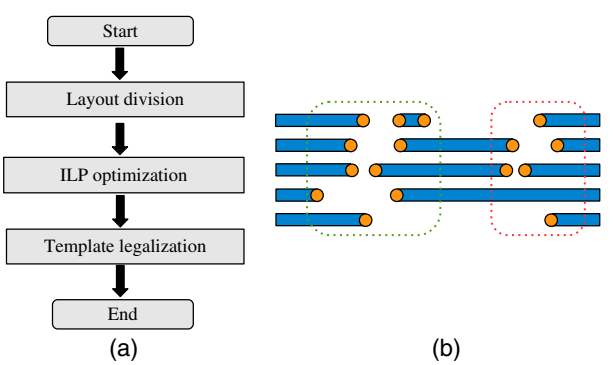

Fig. 4 (a) Overall flow of integer linear programming (ILP) speed-up method and (b) an example of layout division strategy.

Algorithm 1 Greedy set cover algorithm

Require: $U = \{u_1, u_2, \ldots, u_{2n}\}$; $T = \{t_1, t_2, \ldots, t_m\}$;

Ensure: Set cover S with minimum cost;

1: $S \leftarrow \phi$;

2: **while** $f(S) \neq f(U)$ **do**

3: calculate α_j for each unpicked set $\{t_1, t_2, \ldots, t_k\}$;

4: pick t_j with minimum α_j;

5: $S \leftarrow S \cup T$;

6: update T; ▷ remove t_k from T which contains covered cuts;

7: **end while**

8: Write the picked sets S;

algorithm is used to solve the set cover problem. Some notations used in the algorithm are as follows.

- $U = \{u_1, u_2, \ldots, u_{2n}\}$, u_i denotes cut i, n is the number of wires;
- $T = \{t_1, t_2, \ldots, t_m\}$, t_j denotes DSA guiding template candidate j, m is the number of all potential templates;
- w_j denotes the cost of each template, the cost of each template w_j is defined as the sum of total line extensions required to get the cuts fitted into template j.

3.3.1 *Set cover algorithm*

The details of our set cover based algorithm are shown in Algorithm 1. Given a set of cuts U and a set of guiding template candidates T, we initialize an empty set S (line 1). Our algorithm consists of several iterations. During each iteration, cost effectiveness α of all the potential templates T are calculated first (line 3), then a guiding template t_j with minimum cost effectiveness α_j is selected (line 4). Since a cut can only be covered by one template, all the other unpicked templates which contain this cut should be deleted from the unpicked templates set (line 6). The process will iterate until all the line ends are covered by the chosen templates' set $f(S) = f(U)$. Cost effectiveness α_j is the evaluation metric for each unselected guiding template t_j during every iteration. α_j can be derived as follows:

$$\alpha_j = \frac{w_j}{|f(S \cup t_j) - f(S)|},$$

where S is the set of selected DSA guiding templates after each iteration. $f(S)$ is taken as the number of total line ends for the selected DSA guiding template set S. Template t_j can cover more cuts with the smallest cost if α_j is the minimal value.

Figure 5 gives a specific example for the algorithm flow. t_1 to t_7 indicate all possible DSA templates, where each template is denoted as a node in the graph. An edge will be added between two nodes if they share the same line end. We can assume that the manufacturing cost for the single-hole template is 1, two-hole template is 2, peanut-shaped template is 4, three-hole template is 3, and four-hole template is 5. Then, in addition to the necessary line extension for each candidate, the weights of different candidates are $w_1 = 2$, $w_2 = 1$, $w_3 = 1$, $w_4 = 6$, $w_5 = 1$, $w_6 = 4$, $w_7 = 6$. Then the initial cost effectiveness results for different candidates are $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 1$, $\alpha_4 = 3$, $\alpha_5 = 1$, $\alpha_6 = 2$, $\alpha_7 = 3$. In the first iteration, DSA guiding template t_2 has the smallest cost-effectiveness, therefore, it is selected and all the nodes, t_4, t_6, t_7 connecting to it are deleted; then among the remaining templates, the cost-effectiveness results are $\alpha_1 = 1$, $\alpha_5 = 1$, $\alpha_3 = 1$. Then t_1 is selected, t_5 is deleted, and finally, t_3 is selected. The process terminates when all the line ends have been covered by the selected DSA guiding templates.

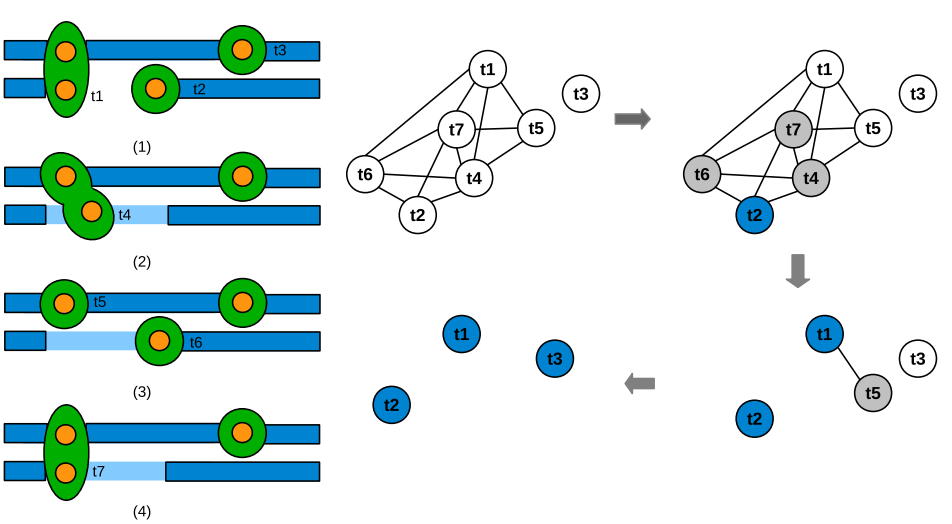

Fig. 5 Set cover algorithm flow.

3.3.2 *Template legalization*

After the DSA template assignment has been decided, the template legalization is then performed to remove overlaps and conflicts. Starting from the left most template: if the current template does not conflict with adjacent templates, then it is placed at its original location; if it does not meet the minimum space requirement but the conflict can be removed by shifting the template to the right, then it is placed at the moved location; if conflict cannot be removed by shifting, it is then marked as a hard conflict. If all templates have been processed, we will try to merge the marked templates with adjacent templates to remove more hard conflicts. The merge process will continue until the number of conflicts converges.

4 Experimental Results

Our algorithms were implemented in C++ and all the experiments were performed on a Linux workstation with Intel Core i7 3.71GHz CPU and 8-GB memory. The state of the art ILP solver, Gurobi 5.6.3 linux64,[25] was used to solve the ILP formulations. Six test benchmarks from Xiao et al.,[10] are used in this work. The benchmarks represent unidirectional metal layers and they are in different sizes. The design rules used in their paper are adopted in our implementation as well. Since the flow in SPIE'13[10] was implemented in MATLAB® and was not available to us, we implemented their method by ourselves for comparison. In this section, we first compare our methodologies with those of the SPIE'13 work,[10] then we further tested the effectiveness of our methodologies on an extended DSA template set.

4.1 *Comparison with SPIE'13*

In the first experiment, we compare our proposed methodologies with the SPIE'13 work.[10] For fair comparison, the same types of templates from their setting are used: single-hole template, two-hole vertical template, three-hole template, and four-hole template. In addition, the weight of DSA guiding templates is not considered. Table 1 shows the results of comparison for each test benchmark. The first column shows the number of cuts for each test benchmark. For each method, columns "#cflt", "ext." and "CPU(s)" represent the number of conflicts, the total wire extensions, and the runtime in seconds, respectively. From the table, we can see that our ILP formulation and ILP speed-up method have no template conflicts. Although our heuristic method has 28 conflicts on average, it can still achieve a lower conflict number than SPIE'13 work. For the total wire extensions, our ILP formulation and ILP speed-up method can achieve about 77% and 68% fewer wire extensions on average than SPIE'13,[10] while our heuristic method can achieve about 21% fewer wire extensions on average. The runtime of our ILP speed-up method increases almost linearly with the problem size as shown in Table 1, which is much faster than the original ILP formulation, but it about 2× slower than SPIE'13. And Our heuristic method is 6× faster than SPIE'13.

4.2 *Optimization Result with Extended Directed Self-Assembly Guiding Templates*

In the second experiment, we compare our algorithms with the extended DSA guiding templates types: one-hole template, two-hole vertical/horizontal template, diagonal template, three-hole template, and four-hole template. The experimental results are shown in Tables 2 and 3. Table 2 shows the results without considerations of the DSA guiding templates weight, while Table 3 incorporates the weight.

As shown in Table 2, the increased guiding template types will increase the problem size of the ILP as expected, thus the runtime greatly increases for the initial ILP. The runtime increases about 7× for the ILP speed-up method. For the heuristic method, there is no significant change. But the wire extensions have been reduced with the increase of template types. If we consider the weight of the DSA guiding templates, the wire extensions will be longer expected, which is shown in Table 3.

Figures 6 and 7 show the DSA guiding templates' assignments of our different methods. The green rectangles depict the guiding templates in different types. For the final template assignment, the ILP and ILP speedup have little difference. The heuristic method will aggressively merge critical cuts into one cut in order to remove conflicts, thus the wire extensions will greatly increase, which can be observed from the figures.

Table 1 Comparison with SPIE'13.[10]

	SPIE'13 (Ref. 10)			ILP			ILP speed-up			Heuristic		
#Cut	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)
50	1	201	0.01	0	28	0.30	0	40	0.24	1	201	0.00
100	6	219	0.02	0	58	2.48	0	60	0.50	5	162	0.01
200	12	534	0.08	0	134	7.88	0	159	1.32	13	215	0.03
500	21	1366	0.42	0	274	354.33	0	338	2.16	18	1122	0.09
1000	46	2709	1.74	0	617	86.00	0	722	4.77	41	2104	0.29
2000	99	5553	7.17	0	1321	55158.58	0	1448	8.91	88	4598	1.15
Avg	30	1764	1.5741	0	405	9268.26	0	461	2.98	28	1400	0.26

Table 2 Comparison with extended directed self-assembly (DSA) guiding templates.

	ILP			ILP speed-up			Heuristic		
#Cut	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)
50	0	26	0.36	0	33	1.48	1	151	0.01
100	0	50	3.00	0	53	4.12	5	159	0.02
200	0	125	198.63	0	138	12.67	12	233	0.02
500	0	256	4882.63	0	287	14.51	25	1008	0.11
1000	N/A	N/A	N/A	0	616	27.73	43	2061	0.41
2000	N/A	N/A	N/A	0	1269	52.55	86	4518	1.73
Avg	N/A	N/A	N/A	0	399	18.84	29	1355	0.38

Table 3 Comparison with weighted and extended DSA guiding templates.

	ILP			ILP speed-up			Heuristic		
#Cut	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)	#cflt	ext.	CPU(s)
50	0	28	0.27	0	39	0.28	1	201	0.00
100	0	58	0.74	0	66	0.87	5	162	0.01
200	0	138	3.65	0	153	3.23	14	201	0.04
500	0	276	39.03	0	350	4.14	20	1122	0.13
1000	0	623	139.45	0	731	8.33	35	2148	0.50
2000	N/A	N/A	N/A	0	1594	16.29	90	4505	3.39
Avg	N/A	N/A	N/A	0	488	5.52	27	1389	0.68

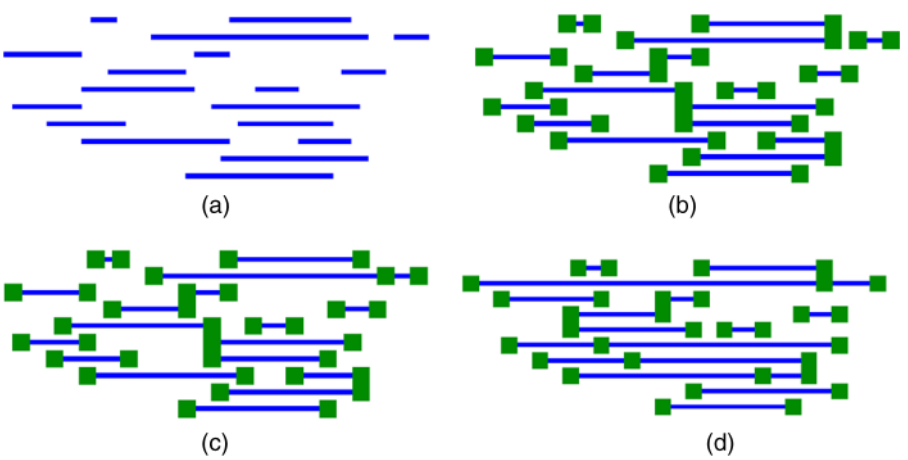

Fig. 6 DSA guiding templates assignment for clip A: (a) target layer, (b) ILP assignment, (c) ILP speed-up assignment, and (d) heuristic assignment.

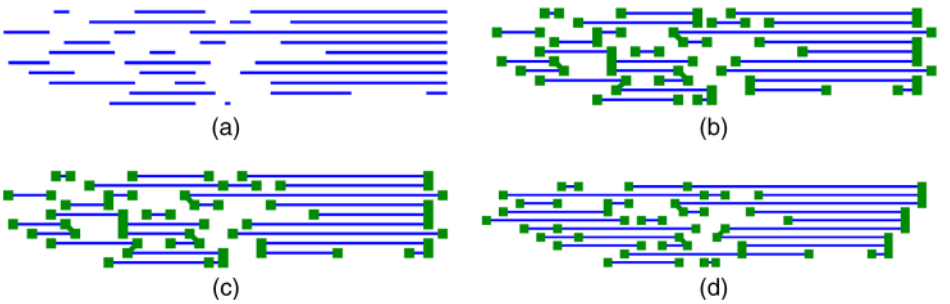

Fig. 7 DSA guiding templates assignment for clip B: (a) target layer, (b) ILP assignment, (c) ILP speed-up assignment, and (d) heuristic assignment.

5 Conclusions

DSA is emerging as the next generation lithography technique due to its ability to scale, its low cost, and its high throughput. However, to fully take advantage of the potential benefits, DSA aware optimization is required at the design stage. In this paper, we perform a thorough investigation on the DSA-based end-cutting problem for unidirectional layout. Three different approaches have been proposed to assign cuts into different DSA guiding templates. The first ILP approach solves the problem with 77% fewer wire extensions and zero conflicts compared to the previous work. The ILP speed-up method solves the problem much more efficiently, with 68% fewer wire extensions and zero conflicts. The heuristic method has 21% fewer wire extensions and 2 less conflicts compared to previous work. We also perform our method with the extended DSA guiding template sets. Our methods are generic and can be adapted to different DSA guiding template shapes, which will be beneficial for the development of the DSA manufacturing technique.

References

1. L. Liebmann, A. Chu, and P. Gutwin, "The daunting complexity of scaling to 7 nm without EUV: Pushing DTCO to the extreme," *Proc. SPIE* **9427**, 942702 (2015).
2. T. Jhaveri et al., "Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **29**(4), 509–527 (2010).
3. W. Ye et al., "Standard cell layout regularity and pin access optimization considering middle-of-line," in *ACM Great Lakes Symp. on VLSI (GLSVLSI)*, pp. 289–294 (2015).
4. D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **32**(10), 1453–1472 (2013).
5. B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *IEEE/ACM Design Automation Conf. (DAC)*, pp. 1–6 (2014).
6. B. Yu et al., "E-BLOW: E-beam lithography overlapping aware stencil planning for MCC system," in *IEEE/ACM Design Automation Conf. (DAC)*, pp. 70 (2013).
7. H. Zhang et al., "Efficient pattern relocation for EUV blank defect mitigation," in *IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*, pp. 719–724 (2012).
8. S.-J. Jeong et al., "Directed self-assembly of block copolymer for next generation nanolithography," *Mater. Today* **16**(12), 468–476 (2013).
9. Y. Seino et al., "Contact hole shrink process using graphoepitaxial directed self-assembly lithography," *Micro/Nanolithogr. MEMS MOEMS* **12**(3), 033011 (2013).
10. Z. Xiao et al., "DSA template mask determination and cut redistribution for advanced 1D gridded design," *Proc. SPIE* **8880**, 888017 (2013).
11. H. Yi et al., "Contact-hole patterning for random logic circuit using block copolymer directed self-assembly," *Proc. SPIE* **8323**, 83230W (2012).
12. Y. Du et al., "Block copolymer directed self-assembly (DSA) aware contact layer optimization for 10 nm 1D standard cell library," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 186–193 (2013).
13. Y. Du et al., "DSA-aware detailed routing for via layer optimization," *Proc. SPIE* **9049**, 90492J (2014).
14. V. Axelrad et al., "Characterization of 1D layout technology at advanced nodes and low k1," *Proc. SPIE* **9052**, 905213 (2014).
15. V. Axelrad et al., "OPC-lite for gridded designs at low k1," *Proc. SPIE* **9235**, 92350C (2014).
16. C. Bencher, H. Dai, and Y. Chen, "Gridded design rule scaling: taking the CPU toward the 16 nm node," *Proc. SPIE* **7274**, 72740G (2009).
17. Y. Ding, C. Chu, and W.-K. Mak, "Throughput optimization for SADP and e-beam based manufacturing of 1D layout," in *IEEE/ACM Design Automation Conf. (DAC)*, pp. 1–6 (2014).
18. Y. Du et al., "Hybrid lithography optimization with e-beam and immersion processes for 16nm 1D gridded design," in *IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*, pp. 707–712 (2012).
19. J.-R. Gao, B. Yu, and D. Z. Pan, "Self-aligned double patterning layout decomposition with complementary e-beam lithography," in *IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*, pp. 143–148 (2014).
20. S. Steen et al., "Hybrid lithography: The marriage between optical and e-beam lithography. a method to study process integration and device performance for advanced device nodes," *Microelectron. Eng.* **83**(4–9), 754–761 (2006).
21. B. Yu et al., "Triple patterning lithography layout decomposition using end-cutting," *J. Micro/Nanolithogr. MEMS MOEMS* **14**(1), 011002 (2015).
22. Y. He et al., "Flexible control of block copolymer directed self-assembly using small, topographical templates: potential lithography solution for integrated circuit contact hole patterning," *Adv. Mater.* **24**(23) (2012).
23. Y. Ma et al., "Challenges and opportunities in applying grapho-epitaxy DSA lithography to metal cut and contact/via applications," *Proc. SPIE* **9231**, 92310T (2014).
24. H.-S. P. Wong et al., "Block copolymer directed self-assembly enables sublithographic patterning for device fabrication," *Proc. SPIE* **8323**, 832303 (2012).
25. Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com (20 May 2014).

Jiaojiao Ou received her BE degree from Tsinghua University and her MS degree from Peking University, Beijing, China. Currently she is pursuing her PhD in electrical and computer engineering from the University of Texas at Austin under the supervision of Prof. D. Z. Pan. Her current research interest is design for manufacturability.

Bei Yu is a postdoctoral scholar in the Department of Electrical and Computer Engineering, University of Texas at Austin, where he received his PhD 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.

Jhih-Rong Gao is a senior technical staffer in Cadence Design Systems, Inc., Austin. She received her PhD 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 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 US patents. He has received many awards, including SRC Technical Excellence Award, 11 Best Paper Awards, among others. He is an IEEE fellow.