Cut Redistribution and DSA Template Assignment for Unidirectional Design

Ye Huang[1], Xingquan Li[1], Wenxing Zhu[1] and Jianli Chen[1*]
[1]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China
*Email: jlchen@fzu.edu.cn

***Abstract*—With shrinking transistors in advanced circuit designs, directed self-assembly (DSA) is considered as one of the most promising techniques for cut patterning in 1-D grided design, due to its low cost and high-manufacturing throughput. In this paper, we consider the cut redistribution and DSA template assignment problem. For a given layout, we first convert the problem to a weighted gap conflict graph. Then, a minimum weighted vertex-disjoint-path cover algorithm is proposed to divide the graph into a set of paths. Finally, for each path, a dynamic programming algorithm is used to minimize the conflicts number and the total wire cost. In particular, the dynamic programming algorithm can guarantee the optimal conflicts number. Experimental results show that our proposed method can obtain free-conflict results for all benchmarks, and achieve the best wire cost, i.e., 34.7% less than a state-of-the-art work.**

***Index Terms*—Directed self-assembly; cut redistribution; vertex-disjoint-path; dynamic programming.**

I. Introduction

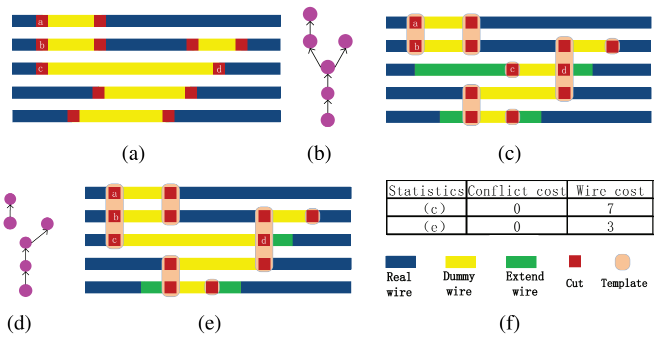
For the advanced integrated circuit (IC) manufacture, 1-D design has attracted more and more attention. Compared with the 2-D layout, 1-D design has a better yield, smaller area, and better uniformity property [1], [2]. What is more, unidirectional (1-D) design enables an effective and simplified design flow, while design rules of conventional 2-D design are dramatically increased along with highly scaled technology nodes. Hence, the 1-D layout design has become the mainstream for lower metal layers [3], [4].

In a regular 1-D layout, cuts are used to find desired layout patterns. Due to the physical limitation of the conventional 193nm immersion lithography system, it is still challenging to print dense cuts for highly unidirectional layout [5], [6].

The emerging block copolymer directed self-assembly (DSA) has considered as one of the most promising techniques for cuts in the sub-10nm technology node, due to its low cost and high-manufacturing throughput [7], [8]. In the DSA process, block copolymers can be formed cylinders, and the material can be used to fabricate contacts/vias by removing these cylinders. DSA holes surrounded by templates can have higher accuracy [9].

Since the DSA template is generally fabricated by the traditional lithography [10], two close templates might conflict with each other and can not be printed. In order to reduce the template conflicts, cuts in layout should be redistributed and be assigned into usable templates. The primary objective of above operations is to minimize the number of conflicts between templates. Consequently, cut redistribution may perturb the locations of some cuts, and thus the lengths of wires for particular circuit functionality vary. Therefore, the total length of extended wires should be also minimized to mitigate the impact on circuit performance [11].

There are many methods have been proposed to handle the critical 1-D layout cut redistribution problem. Xiao et al. in [11] first considered the cut redistribution and DSA template assignment problem. They presented an algorithm to redistribute the original cuts such that the cuts are grouped into non-conflict DSA templates. However, this greedy-like method might



Statistics	Conflict cost	Wire cost
(c)	0	7
(e)	0	3

Fig. 1: Comparisons on different minimum vertex-disjoint path algorithms. (a) An original layout; (b) Gap conflict graph; (c) Result generated by solving minimum vertex-disjoint-path cover in [13]; (d) Minimum vertex-disjoint-paths; (e) Result generated by applying our proposed algorithm; (f) Statistics results of (c) and (e)

not reduce the number of conflicts effectively. What is more, matching cuts to multi-cut templates may increase wire lengths. In order to obtain a optimal solution for the cut redistribution and DSA template assignment problem, Ou et al. formulated the problem as an integer linear programming (ILP) to assign cuts to different guiding templates [12]. Its objective is to minimize the number of conflicts and line-end extension. Due to the complexity of this problem, solving the ILP is time-consuming.

The state-of-the-art work is proposed by Lin et al. [13]. They first presented a dynamic programming algorithm for a special case of the cut redistribution and DSA template assignment problem, where there is at most one dummy wire segment on a track. For general cases, a given layout is divided into several independent special subproblem (case). The operation of division is achieved by solving a minimum vertex-disjoint path cover problem. There is a problem that, such divisions may be not desirable, and limit the solution quality.

As shown in Fig. 1, Fig. 1(a) is a given original layout. Applying the method in [13], conflict graph in Fig. 1(b) is divided into paths in Fig. 1(d). Then by executing the dynamic programming for every path, a final result as shown in Fig. 1(c). The number of conflicts is 0, and the wire cost is 7. However, applying our proposed algorithm, a better result for this layout is shown in Fig. 1(e), whose number of conflicts is 0, and wire cost is 3.

In this paper, we propose an effective method to handle the cut redistribution and DSA template assignment problem. The major contributions of our proposed are summarized as follows.

- For a given layout, we convert the cut redistribution and DSA template assignment problem to a weighted gap conflict graph.
- We propose a new minimum weighted vertex-disjoint-path cover algorithm to divide a the weighted gap conflict graph into a series of vertex-disjoint-paths, which can reduce the extend wire cost significantly.

- For each path, a dynamic programming algorithm is used to minimize the conflicts number and the total extension of wire. In particular, the dynamic programming algorithm can guarantee the optimal conflicts number.
- Experimental results show that our proposed method can obtain free-conflict results for all benchmarks, and achieve the best wire cost, i.e., 34.7% less than a state-of-the-art work.

The rest of this paper is organized as follows. In Section II, we introduce the related concepts and the problem. In Section III, we describe our algorithm. Experimental results are presented in Section IV, and conclusions are made in Section V.

II. Problem Statement

In this section, we firstly introduce the cut redistribution and DSA template assignment for 1-D layout problem, respectively. Then, we give the cut redistribution and DSA template assignment problem.

A. 1-D Layout End-Cutting Process

In 1-D design, lines-cut is an effective approach to solving physical connectivity. In order to consider physical connectivity, some mental lines should be extended when printing mental lines. The 1-D layout is manufactured via printing unidirectional lines and removing parts of the lines by cuts. As shown in Fig. 1 (a), there are 5 parallel mental lines in this layout, and a series of cuts are inserted to split these mental lines into real wires and dummy wires. One of crucial problems is the manufacture of these inserted cuts. Due to the conflict generating by the optical resolution spacing limit, too dense cuts cannot be well patterned. In practice, in order to eliminate conflict, some cuts are redistributed. It must be noted that a cut in a metal row can be only moved in the interval of dummy wire.

B. DSA Template

DSA enables patterning of dense contact holes with smaller spacing via the use of templates. The close contacts can be guided be a template for patterning. For example, in Fig. 1(a), two close contacts a and b are guided by a two-hole template as shown in Fig. 1(b). Since irregular template has a higher chance of generating overlay error, to guarantee the overlay accuracy, only some regular templates with few holes can be used. In this work, seven types of templates $t_1 \sim t_7$ in Fig. 2 are used to guided cuts. These templates are regular and reliable.

Since the cuts in a template should be aligned, we consider the cut redistribution and DSA template assignment problem on grid model. Let the coordinate of the bottom left is (0, 0), and the seven types of templates are denoted as follows. t_1, a cut at (0, 0). t_2, two cuts at (0, 0) and (0,1). t_3, two cuts at (0, 0) and (1,0). t_4, two cuts at (0, 0) and (1,1). t_5, two cuts at (0, 0) and (1,-1). t_6, three cuts at (0, 0), (0,1) and (0,2). t_7, four cuts at (0, 0), (0,1), (1,0) and (1,1). In the DSA template guided cut redistribution problem, every cut must be assigned to a template in a layout.

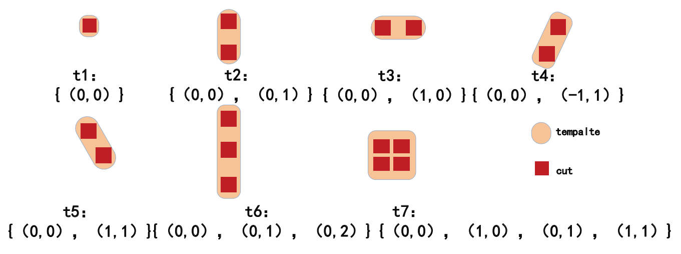


Fig. 2: Seven types of templates

C. Problem Formulation

Problem 1 (Cut redistribution and DSA template assignment)**.** *Given a 1-D layout with n metal rows, a series of cuts and seven types usable templates, the objective of cut redistribution and DSA template assignment problem is to move cuts in the intervals of dummy wires and determine the template assignment for every cut. The primary objective is minimizing the number of conflicts between cuts in different templates. The secondary objective is minimizing the total displacement of cuts.*

III. Our Proposed Method

Since the cut redistribution and DSA template assignment in 1-D design is a NP-hard problem, it is hard for us to deal with it directly. In our proposed method, the overflow consists three parts: (1) weighted gap conflict graph construction, (2) minimum weigh vertex-disjoint-path cover, and (3) redistributing cuts and assigning template on each vertex-disjoint path by a dynamic programming.

A. Weighted Gap Conflict Graph Construction

A metal wire row is divided into many real wires and dummy wires by the inserted cuts. If the distance between two cuts is less than the optical resolution spacing limit d_s, then a conflict is generated. To eliminate conflicts, we can move some cuts in the dummy wires or assign some close cuts into a template. Based on the Definitions 1 and 2 in [13], we construct a weighted gap conflict graph in Definition 3 as follow.

Definition 1 (Gap [13])**.** *A gap g_i of a dummy wire is denoted by (y_i, a_i, b_i), where y_i is the row number of gap i, and a_i, b_i are the horizontal coordinates of the original left, right cut in a dummy wire interval, i.e., the left (right) line end of a dummy wire.*

A gap (y_i, a_i, b_i) is shown in Fig.3. Cuts are allowed to move only within interval $[a_i, b_i]$. In Fig. 3, the left cut at a_i only can move to the right, and the right cut at b_i only can move to the left. Let l_i and r_i are the final locations of the cuts in this gap. Then, the extend wire cost of this gap is $(l_i - a_i) + (b_i - r_i)$. Given a 1-D layout, we can easily find all gaps. Further, a gap is regarded as a vertex in the weighted gap conflict graph. The gap conflict graph is defined as follow.

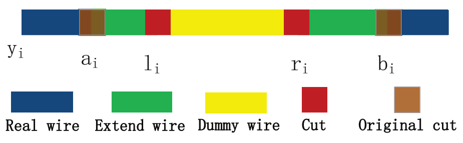


Fig. 3: Structure of a gap

Definition 2 (Gap conflict graph [13])**.** *The Gap conflict graph is a directed graph $G(V, E)$, where each vertex in V represents a gap, and e_{ij} in E is the directed edge between gaps g_i and g_j.*

According to the partition in [13], cuts b and c in Fig.1(a) are divided into different paths. Comparing to Fig. 1(e), the above operation lead to more extend wire cost for eliminating conflict between b and c. Obviously, if cuts b and c are considered in a path, we can obtain a better solution result.

In our proposed method, a new structure 'virtual-gap node' is used to replaced the gap with out-degree or in-degree greater than 1 in gap conflict graph. This replacement has the follow characters: (1) A gap node will be divided into two virtual gap nodes; (2) Single virtual-gap node has only single cut, another hole is empty in this gap; (3) Both two virtual gaps inherited left or right cut of original gap.

The objective of this process is to divide a gap conflict graph into some vertex-disjoint paths and reduce the cost of partition. We assign weighted for each edge between gaps to obtain a weighted gap conflict graph. The weighted gap conflict graph is defined in Definition 3.

Definition 3 (Weighted gap conflict graph)**.** *The Weighted gap conflict graph is a weighted undirected graph $G(V, E, W)$, where each vertex in V represents a gap or a virtual gap, and e_{ij} in E is weighted undirected edge between gaps g_i and g_j, W is the set of weights of edges.*

B. Minimum Weighed Vertex-Disjoint Path Cover

Based on the Definition 3, a minimum weigh vertex-disjoint path cover instead of minimum vertex-disjoint path cover [13] is used to focus on reducing the cost of discarding conflict edge.

As shown in Fig. 4, minimum weighted edges is equivalent to minimizing the sum of edge which are cut by red secant. Fig. 4(a) and Fig. 4(b) is the original layout, and the gap conflict graph generated by [13], respectively. In our weighted gap conflict graph, as shown in Fig. 4(c), we divide key-gap 1 into two virtual gaps 4 and 5, where a gap is defined as a key-gap if its in-degree or out-degree is greater than 1. As a result, there would be almost no effect between nodes 4 and 5. In Fig. 4(c), the red number is the weighted of the corresponding edge; and we select a partition solution which is provided by red secant in Fig. 4(d). In Fig. 4(e), we obtain two vertex-disjoint paths finally. As shown in Fig. $1(e)$, intuitively cut a, b, c can be put into a template type t_6.

Our weighted gap conflict graph is builded based on the gap conflict graph. The overflow of the constructing weighted gap conflict graph and our partition includes following three parts: (1) divide key-gap into two virtual gaps; (2) assign different values to each edge of the weighted gap conflict graph; (3) select the best feasible solution.

In the weighted gap conflict graph, there are three types for key-gap: (1) convergent type; (2) bifurcates type; and (3) mixed type. The convergent type is the structure with the in-degree of key-gap node is only equals to 2. The bifurcates type is the structure with the out-degree of key-gap node is only equals to 2. And the mixed type is the structure with both the in-degree and out-degree of key-gap node are equals to 2. It is easy to show that other types can be reduced to above three types. Fig. 5 (1-a), Fig. 5 (2-a) and Fig. 5 (3-a) represent the three key-gap types, respectively.

As shown in Fig. 5(1-b), (2-b), (3-b), the key-gap 1 is split into two virtual-gap nodes for the three types. The weighted of edge is relevant to the distance between nodes. The further distance between nodes, the small value of weighted edge. On the contrary, the closer between two nodes, the bigger value of weighted edge. In our proposed method, each value is not less than 0, and it is not greater than 100. Then, we classify edges into four types: conflict edge, connected edge, non-connected edge, and non-segmentation edge. Conflict edge denotes two nodes lie to the adjacent row and violent design rules. Connected edge denotes two gap nodes have connection. Non-connected edge denotes two gaps have not connection, which means the distance of the two nodes is too large. Non-segmentation edge denotes two nodes can not be separated, which means both of the two nodes are on the same row,and the distance is equals to 1. The four types edge have different weights assigned, and they are set to 5, 1, 0, and 100, respectively.

We should assign different edge to different weighted on the basic of Fig. 5(b). In really, the value of weighted edge depends on the distance between nodes. Hence, the value can not be

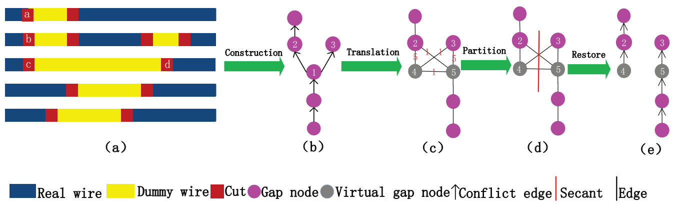


Fig. 4: Vertex-disjoint-path partition. (a) An original layout; (b) Gap conflict graph; (c) Weighted gap conflict graph; (d) Minimum cut; (f) Partition result

exhibited in 5(b). In Fig. 5(c), there are some feasible solutions. Finding vertex-disjoint paths cover for the three types is based upon applying the minimum weighted [14]. For each feasible solution, we calculate the sum of cut weighted edges which are cut by red secant, and then select minimum one from them. Since the weighted gap conflict graph is a undirected graph, a process of restoration is necessary. Fig. 5(d) shows the process of restoration.

C. Dynamic Programming

In our proposed method, the dynamic programming is based on that in [13]. Let $\Phi = \{g_1, g_2, g_3, \cdots, g_n\}$ be a set of gaps ordered from the bottom to the top, where n is the number of gaps. It is above gaps-paths. That is, for any two gaps g_i and g_j, the vertical coordinate of g_i is smaller than that of g_j, if $i < j$.

The structure of $s[i, j] = (l_{i,j}, r_{i,j}, tl_{i,j}, tr_{i,j})$ is used to describe the message of two cuts which are in a gap. $l_{i,j}$ (or $r_{i,j}$) denotes the redistributed horizontal location of the left (right) cut of g_i. $tl_{i,j}$ (or $tr_{i,j}$) denotes the redistributed a template type of the left (right) cut of g_i. $v_{i,j}$ only link to $v_{i+1,k}$. In a subproblem graph, each vertex $v_{i,j}$ represents a feasible subproblem $s[i, j]$. each directed edge $(v_{i,j}, v_{i+1,k})$ have weighted or cost: $d_1(v_{i,j}, v_{i+1,k})$, $d_2(v_{i,j}, v_{i+1,k})$, $d_3(v_{i,j}, v_{i+1,k})$. $d_1(v_{i,j}, v_{i+1,k})$ denotes the number of the conflict edges between g_i and g_{i+1} for $s[i, j]$ and $s[i+1, k]$. $d_2(v_{i,j}, v_{i+1,k})$ denotes the extend wire cost between g_i and g_{i+1} for $s[i, j]$ and $s[i+1, k]$. $d_3(v_{i,j}, v_{i+1,k})$ denotes the used template cost between g_i and g_{i+1} for $s[i, j]$ and $s[i+1, k]$.

The major difference between our dynamic programming and the dynamic programming in [13] is that, we add the new definition of $d_3(v_{i,j}, v_{i+1,k})$. In best predecessor determination, determining the best predecessor is actually deriving the optimal substructure of the dynamic programming problem. Let $C_c[i, j]$ be the best conflict cost of $s[i, j]$, $C_w[i, j]$ be the best extend wire cost of $s[i, j]$, and $C_t[i, j]$ be the best using DSA template cost of $s[i, j]$. In the dynamic programming algorithm, we choose the following best one:

$$C_c[i+1, k] = min\{C_c[i, j] + d_1(v_{i,j}, v_{i+1,k}) + c[i+1, k]\};$$
$$C_w[i+1, k] = min\{C_w[i, j] + d_2(v_{i,j}, v_{i+1,k}) + w[i+1, k]\};$$
$$C_t[i+1, k] = min\{C_t[i, j] + d_3(v_{i,j}, v_{i+1,k}) + t[i+1, k]\}.$$

IV. Empirical Study

In order to evaluate the performance of our proposed method, we implemented it in C++ programming language and performed on an Intel Core(TM) i5-6500 CPU @3.2 GHz CPU and 2GB RAM. The tested benchmarks provided by the author of [11] are used. Since the previous work [11]–[13] only used t_1, t_2, t_3, t_6, and t_7 types of templates. For fair comparison, we also only used t_1, t_2, t_3, t_6, and t_7 templates in our experiment. The experimental results are listed in Table I, in which SPIE'13, GLVLSI'15, ASP-DAC'16 and ours denote the methods in [11], [12], [13] and our method, respectively.

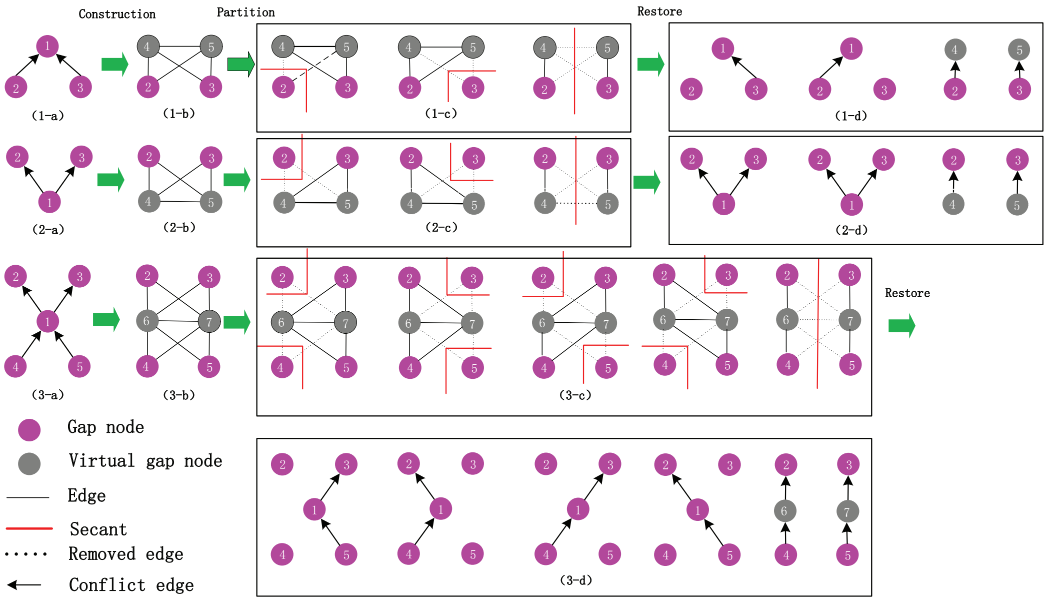


Fig. 5: Our general processes. (a)Original gap conflict graph; (b) After gap replacing by virtual gap; (c) Possible partitions; (d) Vertex-disjoint-paths

TABLE I: Comparison of computational results of three methods for the RDWD problem

		SPIE'13 [11]			GLVLSI'15 [12]			ASP-DAC'16 [13]			Ours		
Benchmarks	#IC	#RC	EW	CPU(s)	#RC	EW	CPU(s)	#RC	EW	CPU(s)	#RC	EW	CPU(s)
50	8	1	32	0.007	0	2	0.117	0	2	0.005	0	2	0.371
100	28	3	71	0.016	0	19	0.278	0	15	0.010	0	13	1.246
200	62	8	206	0.033	2	46	0.530	0	48	0.018	0	39	1.722
500	159	21	541	0.103	6	107	1.827	0	130	0.052	0	94	5.679
1000	295	36	1049	0.277	10	187	2.861	0	248	0.102	0	189	10.811
2000	624	96	1937	0.835	18	384	6.367	0	578	0.222	0	421	19.099
Avg.	196	27	639	0.212	6	124	1.997	0	170	0.068	0	126	6.488
Ratio			5.061	0.033		0.983	0.308		1.347	0.011		1.000	1.000

In Table I, columns #IC, #RC, EW are the number of initial conflicts, the number of remaining conflicts and the total length of extension wires, respectively. It can be seen from Table I that, the ASP-DAC'16 [13] and Ours can achieve free-conflict results for all tested benchmarks, while SPIE'13 [11] and GLVLSI'15 [12] still exist many remaining conflicts. Furthermore, the average EW in Ours is 34.7% less than that in ASP-DAC'16. Although our runtime is the longest, the maximum runtime is only 19.099s, and it is admissible. The experimental results show that our proposed method is effective.

V. Conclusion

In this paper, we considered the cut redistribution and DSA template assignment for 1-D layout design. We converted this problem to a weighted gap conflict graph, presented a minimum weighted vertex-disjoint-path cover algorithm to divide the graph into a set of paths, and solved each path by the dynamic programming. Experimental results show the proposed method can obtain free-conflict results for all benchmarks and reduce the wire cost obviously.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 11501115 and 61672005, and in part by the Distinguished Young Scholars Foundation of FuJian Provincial Educational Committee under Grant 601768.

References

[1] V. Axelrad, K. Mikami, M. C. Smayling, K. Tsujita, and H. Yaegashi,"Characterization of 1D layout technology at advanced nodes and low k1," in *Proc. of SPIE*, Vol. 9052, 905213, 2014.

[2] M. C. Smayling, "1D design style implications for mask making and CEBL," in *Proc. of SPIE*, Vol. 8880, pp. 888012, 2013.

[3] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nano lithography," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst*, Vol. 32, no. 10, pp. 1453-1472, 2013.

[4] T. Jhaveri, V. Rovner, L. W. Liebmann, L. T. Pileggi, A. J. Strojwas, and J. D. Hibbeler, "Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst*, Vol. 29, no. 4, pp. 509-527, 2010.

[5] L. Liebmann, A. Chu, and P. Gutwin, "The daunting complexity of scaling to 7 nm without EUV: Pushing DTCO to the extreme," in *Proc. of SPIE*, Vol. 9427, 942702, 2015.

[6] Y. Ding, C. Chu, and W.-K. Mak, "Throughput optimization for SADP and e-beam based manufacturing of 1D layout," in *Proc. of DAC*, 2014.

[7] Y. Du, D. Guo, M. D. F. Wong, H. Yi, H.-S. P. Wong, H. Zhang, and Q. Ma, "Block copolymer directed self-assembly (DSA) aware contact layer optimization for 10 nm 1D standard cell library," in *Proc. of ICCAD*, 2013.

[8] S. Jeong, J. Y. Kim, B. H. Kim, H. Moon, and S. O. Kim, "Directed self-assembly of block copolymer for next generation nanolithography," *Mater. Today*, Vol. 16, no. 12, pp. 468-476, 2013.

[9] H.-S. P. Wong, C. Bencher, H. Yi, X.-Y. Bao, and L.-W. Chang, "Block copolymer directed self-assembly enables sublithographic patterning for device fabrication," in *Proc. of SPIE*, Vol. 8323, pp. 832303, 2012.

[10] Y. Du, Z. Xiao, M. D. Wong, H. Yi, and H. S. Wong, "DSA-aware detailed routing for via layer optimization," in *Proc. of SPIE*, Vol. 9049, 90492J, 2014.

[11] Z. Xiao, Y. Du, M. D. F. Wong, and H. Zhang, "DSA template mask determination and cut redistribution for advanced 1D gridded design," in *Proc. of SPIE*, Vol. 8880, pp. 888017, 2013.

[12] J. Ou, B. Yu, J.-R. Gao, M. Preil, A. Latypov, and D. Z. Pan, "Directed self-assembly based cut mask optimization for unidirectional design," in *Proc. of GLSVLSI*, pp. 83-86, 2015.

[13] Z.-W. Lin, and Y.-W. Chang, "Cut Redistribution with Directed Self-Assembly Templates for Advanced 1-D Gridded Layouts", in *Proc. of ASPDAC*, pp. 89-94, 2016.

[14] F. T. Boesch and J. F. Gimpel,"Covering the points of a digraph with point-disjoint paths and its application to code optimization," *J. ACM*, Vol. 23, No. 2, pp. 192, 1977.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms," 3rd Ed., *The MIT Press*, 2009.