Native conflict awared layout decomposition in triple patterning lithography using bin-based library matching method

Xianhua Ke, Hao Jiang, Wen Lv, and Shiyuan Liu*

State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

ABSTRACT

Triple patterning (TP) lithography becomes a feasible technology for manufacturing as the feature size further scale down to sub 14/10 nm. In TP, a layout is decomposed into three masks followed with exposures and etches/freezing processes respectively. Previous works mostly focus on layout decomposition with minimal conflicts and stitches simultaneously. However, since any existence of native conflict will result in layout re-design/modification and re-performing the time-consuming decomposition, the effective method that can be aware of native conflicts (NCs) in layout is desirable. In this paper, a bin-based library matching method is proposed for NCs detection and layout decomposition. First, a layout is divided into bins and the corresponding conflict graph in each bin is constructed. Then, we match the conflict graph in a prebuilt colored library, and as a result the NCs can be located and highlighted quickly.

Keywords: layout decomposition, triple patterning, native conflict, graph library

1. INTRODUCTION

As technology node in integrated circuits further scales down, optical lithography is currently extended to print feature at approximate a twentieth of the optical wavelength using ArF immersion technique. Triple patterning (TP) lithography, as a natural extension from double patterning (DP) lithography, is widely recognized as a promising solution for this technology node, i.e. 14/10 nm. Similar to DP lithography, one of the key challenges in TP lithography is the layout decomposition. In TP layout decomposition, a layout is decomposed into three masks followed with exposures and etches/freezing processes respectively. From the aspect of topological geometry, two polygonal features where the distance between them in a layout is less than a given minimal spacing, are regarded as in conflict, and should be assigned to different masks to achieve pitch relaxation. Generally, in order to resolve some conflicts, stitches are inserted to split one continuous feature into two or more parts assigned to different masks. As shown in Fig. 1(a), the bottom feature is split into two parts to make the layout decomposable. Nevertheless, inserting stitches sometimes may not work for some special conflict pattern. For example, Fig. 1(b) shows a conflict that cannot be resolved by inserting stitches anywhere, which is called a native conflict (NC).

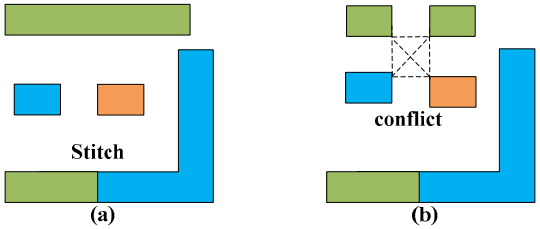

Fig. 1 (a) The bottom polygon is split into two parts by a stitch to make the layout decomposable; (b) there is a native conflict in the layout and this conflict cannot be resolved by stitches inserting.

* Corresponding author: shyliu@hust.edu.cn; phone: +86 27 87559543; webpage: http://www2.hust.edu.cn/nom.

Optical Microlithography XXIX, edited by Andreas Erdmann and Jongwook Kye, Proc. of SPIE
Vol. 9780, 97801F · © 2016 SPIE · CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2219144

In DP lithography, the layout decomposition problem is essentially a two-color labeling problem and is often referred to as DP coloring. A lot of works have been done to minimizing the number of the conflicts and stitches simultaneously [2-8]. It should be noted that any existence of native conflict would result in layout re-design/modification. Fortunately, detecting a NC in DP lithography is very intuitive, which is equivalent to odd cycle checking in a conflict graph. Some previous works pointed out the NC issue in DP and utilized a layout migration technique to modify the layout automatically [9, 10]. Compared to DP coloring, the NC detection in TP coloring is not straightforward and even finding a four-clique is NP-complete. Current TP layout decomposition methods focus on minimizing the number of conflicts and stitches [1, 11-15]. Since the existence of NC leads to layout re-design/modification, a method that is capable to predict and highlight the occurrences of NCs is highly desirable.

In this paper, we propose a strategy which combines a bin-based layout partition method with a graph matching algorithm to detect the NCs in TP. In this method, an isomorphism-free graph library is built firstly, and each graph in the library is pre-assigned into three different colors. For a given layout, we divide it into bins and construct the corresponding conflict graphs in each bin. Then, we match the constructed conflict graphs with the TP colored graphs in the library and therefore NCs can be detected. Besides, the proposed method is extendable by applying the speedup techniques presented in refs. [1] and [12]. The major contributions of this paper are summarized as follows:

(1). We propose a bin-based partition method. Firstly we divide a mask layout into bins to reduce a complicate problem to several simple sub-problems. After the TP coloring in one bin is completed, this bin is recognized as a node (referred as bin-node), and the combination of a few bin-nodes can be regarded as a new graph for further recursive graph matching;

(2). We propose a NC detection method based on graph matching. A graph library is built with the isomorphism-free generation methodology, and then we match the conflict graph to the graphs in the library to detect NCs;

(3). We propose an ant colony algorithm to perform nodes coloring when the number of nodes in a conflict graph is larger than the upper limit nodes number we set in the library.

2. METHODOLOGY

As triple patterning (TP) layout decomposition is essentially a three-color graph coloring problem, finding native conflicts (NCs) in the layout is converted to the NC detection problem in the corresponding conflict graph. This problem can be defined as follows: Given a layout and a minimal coloring spacing cs_{min}, the location of the NCs should be highlighted exactly and quickly for further layout local modifying.

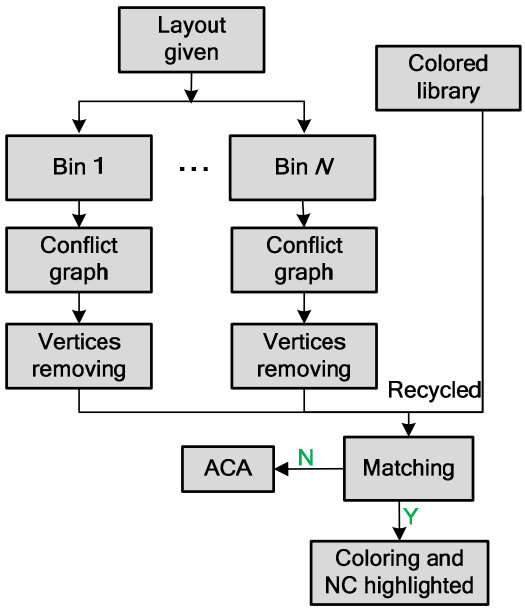

Fig. 2 The overall flow of bin-based library matching method.

Inspired by Refs. [7] and [14], we propose a bin-based library matching method to detect the NCs in TP coloring. Figure 2 shows the overall flow of our proposed method. First, an isomorphism-free graph generation method is adopted to build a graph library. Assume that an upper limit of nodes number in a graph is set as 7, and then we can compute all corresponding graphs with nodes number no larger than 7. After a graph library being built, each graph in this library would be colored and NC highlighted. Therefore, a colored graph library is generated.

When a layout is given, we divide it into bins according to density of the layout, and then a conflict graph is constructed in each bin. In a conflict graph, the polygonal features are represented by nodes and there is a line between them when their distance is less than cs_{min}. Generally, the edge number of one node is called the degree of this node. The nodes removal method is applied for conflict graph repetitively to remove the node with a degree less than three. After nodes removal, we get a sub-graph in each bin. Then the sub-graph matches the graphs in the colored library using a graph isomorphism algorithm in polynomial-time. The polygonal features coloring can be done simply according to the matched graph in the colored library.

In most cases, the number of nodes in the sub-graph we obtained is less than eight, and so that almost all the sub-graphs can be covered in the library. However, for completeness of our method, ant colony algorithm (ACA) is applied to color the graphs with larger node number.

2.1 Native Conflict

In TP layout decomposition, a layout is decomposed into three masks followed with exposures and etches/freezing processes respectively. Two polygonal features in the layout are regarded as in conflict when their distance is less than the minimal coloring spacing cs_{min}, so that they should be assigned into different masks. Since the existing of conflicts make the layouts un-decomposable, stitches are inserted to split a polynomial feature into two or more parts to solve some conflicts. However, stitches insertion might not be powerful enough for resolving all the conflicts. The conflicts that cannot be resolved by inserting stitches anywhere is called a NC. As shown in Fig. 3(a), polygonal features P4, P5, P6, and P7 are in conflict with each other and these conflicts are NCs.

As TP layout decomposition is essentially a three-color graph coloring problem, finding NCs in the layout is converted to NCs detection in the corresponding conflict graph. Normally if a minimal coloring spacing cs_{min} is given, the conflict graph of the layout can be constructed accordingly, where the polygonal features in the layout are represented by nodes in the graph, and the conflicts in layout are represented by lines between these nodes in the graph.

It's well-known that in a conflict graph for double patterning (DP), NC detecting is equivalent to odd cycle (OC) detecting and it can be resolved in linear time using a breath-first-search algorithm. As shown in Fig. 3(b), there some OCs exist. For example, since features P1, P2, and P7 are in conflict mutually, they form an OC. But in TP, NC detecting is not so straightforward and determining the decomposability of a layout is very complex. As shown in Fig. 3(b), features P4, P5, P6, and P7 are in conflict mutually, and they form a 4-clique which is one of the simplest NC patterns in TP. However, even finding a 4-clique in the conflict graph is NP-complete.

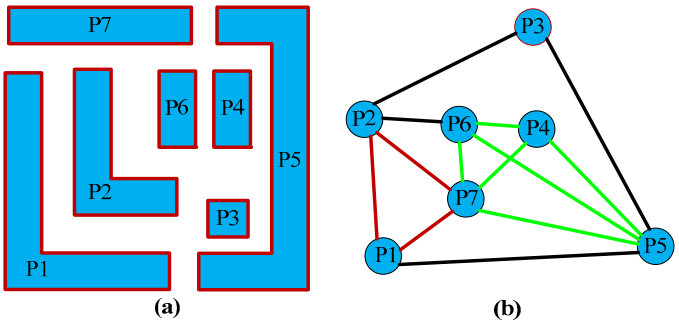

Fig. 3 (a) An example of a layout; (b) the corresponding conflict graph.

2.2 Colored Library

As we know, a graph G is a 2-connected graph only when there is no single node whose removal disconnected this graph. Two 2-connected graphs, G_A and G_B are said to be isomorphic if and only if their edge structures can be exactly

same after nodes rearrangement [16]. As shown in Fig. 4, graph G_A and G_B is isomorphic even though the explicit labeling are not the same. When node 2 and node 5, node 3 and node 4 exchange their position, the edge structures of G_A and G_B are identical. Obviously, only explicit labeling graphs can be stored, and then an unlabeled graph may be repeated $n!$ times without controlling in the process of building a graph library, for the reason that different labeling graphs may be isomorphic and are the same unlabeled graph. This repetition makes the process of matching more redundant. Taking account of the above consideration, an isomorphism-free graph library is needed.

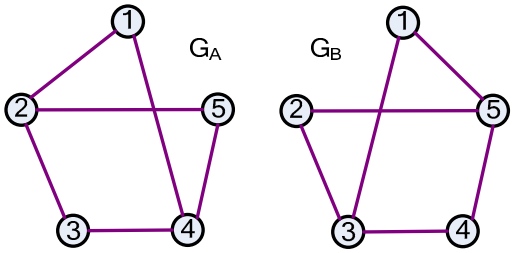

Fig. 4 Graph G_A and G_B are isomorphic without the same labeling.

In this paper, the McKay's generation technique proposed in ref. [18] is applied to generate 2-connected isomorphic graph classes. Based on these classes, we employ the ear-augment method [16] to the isomorphic classes to get a 2-connected isomorphism-free graph library. In this library, only 2-connected, isomorphic-free graphs are contained.

While an upper limit of nodes number in a graph is set as 7, an isomorphism-free graph library can be constructed. For a graph in this library, if there is node whose degree less than three, this graph is called a simple graph. In our library, we filter out all the simple graphs, and obtain a new library. In this new library, each graph is colored with three colors under the rule that the color of nodes connected by a line should be different, and the nodes can not be colored will be NC highlighted in a dash rectangle in each graph. Finally, a colored library is finished.

In our colored library, a few parts of isomorphism-free graphs are presented in Fig. 5. In these graphs, the degree of every node is not less than three, and at the same time the NCs are high-lighted in a light–salmon dash rectangle. These graphs can be used for matching graphs whose node number no more than 7.

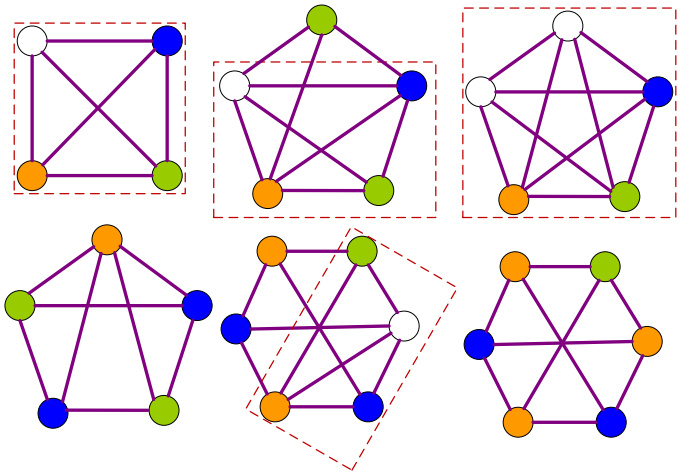

Fig. 5 A few of graphs with four or five nodes in the library and pre-assigned with three colors.

2.3 Conflict Graph Matching

For a given layout, we divide it into bins according to the density of the layout. In each bin, the corresponding conflict graph is constructed. Then the nodes removal method [12] is applied for this conflict graph repetitively to remove those nodes whose degree less than three. After nodes removal, we get a sub-graph, represented by G_m here, and the G_m will match the graphs in our colored graph library in polynomial time using an isomorphism algorithm [17]. Once G_m matches one graph G_n in the colored library, we can do coloring or NCs highlighting by simply mapping the colors in G_n

to G_m. When the sub-graph G_m is colored, we can recall and easily color the nodes removed in the process of nodes removal. Therefore, the conflict graph in this bin is colored, and so does the corresponding polygonal features.

After the conflict graph in the bin is colored by three colors, the colors of nodes in the bin can be interchanged. So after TP in each bin is resolved, it can be treated as a new node-bin, and the conflicts between the bins becomes the conflicts of the corresponding bin-nodes. Based on this formulation, the library matching process can be repeated to determinate the final colors of the nodes in the graph and the polygonal features in the layout.

2.4 Ant Colony Algorithm for Graphs Un-matching

However, an upper limit of nodes number in a graph is set in the graphs of our library, sub-graph with nodes more than the upper limit cannot be coloring by graph matching. Here, we propose an ant colony algorithm (ACA) for the graphs un-matching. In ACA, each ant acts as a solution explorer to randomly assign a color to one polygonal feature, and leaves a pheromone trail between the feature and the color [15]. As initial setup, a greedy factor $\eta_{i,m}$ is set for the process of assigning coloring feature m to color i, and the corresponding probability of is $P_{i,m}$:

$$P_{i,m} = \frac{\eta_{i,m}^{\beta}}{\sum_{i \in W} \eta_{i,m}^{\beta}}, \tag{1}$$

Here, the W presents the space where features can be assigned into m color. After each assignment, the trail $\tau_{i,m}$ between feature i and color m is update as:

$$\tau_{i,m} = (1-\rho)\tau_{i,m} + \sum_{n=1} \Delta\tau_{i,m}(n), \tag{2}$$

Here ρ is the evaporation rate of pheromone trails, and $\Delta\tau_{i,m}$ is the best quantity of pheromone tails deposited on the node i and color m by the ant n. Based on the pheromone trails, the assignment will under the new probability $P_{i,m}$ for better solution exploring :

$$P_{i,m} = \frac{\tau_{i,m}^{\alpha}\eta_{i,m}^{\beta}}{\sum_{i \in W} \tau_{i,m}^{\alpha}\eta_{i,m}^{\beta}}. \tag{3}$$

3. EXPRIMENTAL RESULTS

The proposed method is implemented and tested in MATLAB on a 2 GHz Linux machine with 4GB memory. The upper limit of node number in a graph is set to seven, and an isomorphism-free colored graph library is built based on this upper limit. As a result, there are 147 graphs in our colored library totally. Some graphs in the library are shown in the Fig. 5.

Some benchmarks in Table 1 are test with our method, the file CSIM1 contains 1109 polygonal features and the layout is divided into 80 bins with 8 row and 10 column, CSIM2 contains 2216 polygonal features and the layout is divided into 132 bins with 11 row and 12 column, CSIM3 contains 2411 polygonal features and the layout is divided into 168 bins with 12 row and 14 column. The pitches of all the benchmarks are 22 nm. The minimal coloring space we set is 20 nm. In each bin, the corresponding conflict graph is constructed, and then nodes removal method is applied for this conflict graph repetitively to remove those nodes whose degree less than three. The sub-graph goes to match graphs in the library. Once it is matched, we can do polygonal features coloring simply according to the graph in the library and NCs can be detected efficiently in the benchmarks.

Table 1. Information of the benchmarks used for NC detection.

Benchmark	Polygon number	Bin number
CSIM1	1109	80 = 8×10
CSIM2	2216	132 = 11×12
CSIM3	2411	168 = 12×14

The layout decomposition and NC highlighted results of CSIM1 are shown in Fig. 6. Three NCs in the layout are highlighted with mustard efficiently and the partial views of NC bins are enlarged for a better visualization. With the bin-based library matching method, it easy to locate the bin, predict the occurrences of NCs and highlight the association area for flexible perturbation.

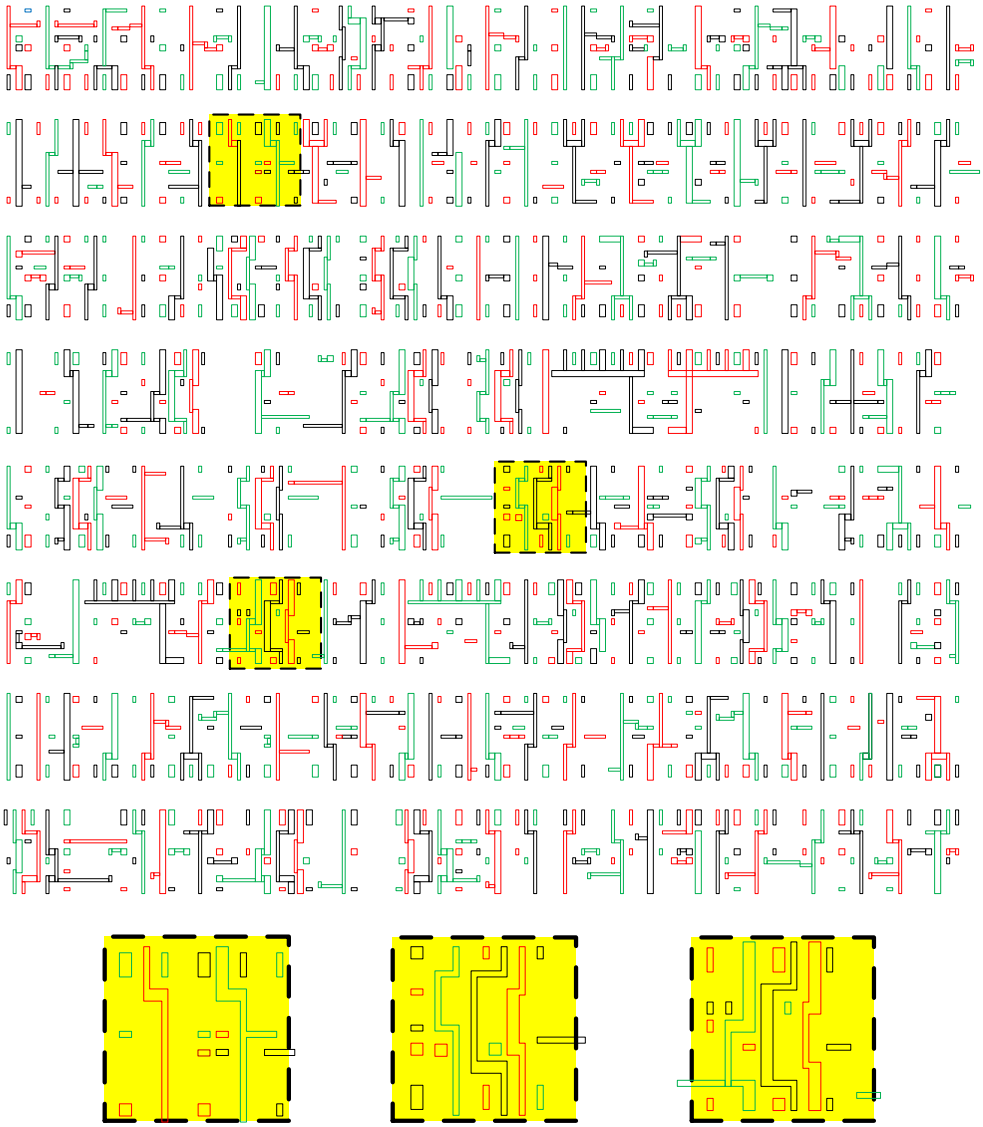

Fig. 6 The layout decomposition result of CSIM1. The NCs are highlighted in a dash box, and enlarged views of NC area are shown in the button of the layout figure.

CONCLUSION

In triple patterning (TP), the native conflicts (NCs) detection is not straightforward and even finding a 4-clique is NP-complete. In this paper, we applied a bins based library matching method to locate NCs. The results show that the method we proposed can detect the NCs and decompose the layout efficiently, and the polygonal features surrounding the NCs can be highlighted for better layout perturbation or modification. Using the proposed method, the periodic time of layout redesign for TP could be potentially decreased.

ACKNOWLEDGEMENT

This work was funded by the National Natural Science Foundation of China (51475191, 51405172, 51575214, and 51525502); the Natural Science Foundation of Hubei Province of China (2015CFB278 and 2015CFA005); China Postdoctoral Science Foundation (2014M56067 and 2015T80791); and the Program for Changjiang Scholars and Innovative Research Team in University of China (IRT13017).

REFERENCES

[1] B. Yu, Y. Kun, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **34**, 433-446 (2015).
[2] G. E. Bailey, A. Tritchkov, J. W. Park, L. Hong, V. Wiaux, E. Hendrickx, S. Verhaegen, P. Xie, and J. Versluijs, "Double pattern EDA solutions for 32nm HP and beyond," *Proc. SPIE* **6521**, 65211K (2007).
[3] T. B. Chiou, R. Socha, H. Chen, L. Q. Chen, S. Hsu, P. Nikolsky, A. V. Oosten, and A. C. Chen, "Development of layout split algorithms and printability evaluation for double patterning technology," *Proc. SPIE* **6924**, 69243M (2008).
[4] A. B. Kahng, C. H. Park, X. Xu, and H. L. Yao, "Layout decomposition approaches for double patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **29**, 939-952 (2010).
[5] K. Yuan, J. S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneously conflict and stitch minimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **29**, 185-196 (2010).
[6] Y. Xu and C. Chu, "A matching based decomposer for double patterning lithography," *Proc. ISPD*, 121-126 (2010).
[7] H. L. Yao, Y. C. Cai, and W. Zhao, "WINPAL: Windows-based parallel layout decomposition in double patterning lithography," *Proc. IEEE on ICSICT*, 1-4 (2012).
[8] A. Sinharay and T. Bakshi, "A graph theoretic framework for double patterning lithographic layout decomposition," *Proc. IEEE on IACC*, 1606-1612 (2013).
[9] C. H. Hsu, Y. W. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," *Proc. IEEE on TCAD*, 284-294 (2011).
[10] S. Y. Fang, S. Y. Chen, and Y. W. Chang, "Native- conflict and stitch-aware wire perturbation for double patterning technology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **31**, 703-716 (2012).
[11] C. Cork, J. C. Madre, and L. Barnes, "Comparison of triple patterning decomposition algorithm using aperiodic tiling patterns," *Proc. SPIE* **7028**, 702839 (2008).
[12] S. Y. Fang, Y. W. Chang, and W. Y. Chen, "A novel layout decomposition algorithm for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **33**, 397-408 (2014).
[13] R. S. Ghaida, K. B. Agarwal, L. W. Liebmann, S. R. Nassif, and P. Gupta, "A novel methodology for triple/multiple patterning layout decomposition," *Proc. SPIE* **8327**, 83270M (2012).
[14] J. Kuang and E. F. Y. Young, "An efficient layout decomposition approach for triple pattering lithography," *Proc. IEEE on DAC*, 1-6 (2013).
[15] X. Ke, W. Lv, and S. Liu, "Ant colony algorithm for layout decomposition in double/multiple patterning lithography," *Proc. IEEE* on *CSTIC*, 1-3 (2015).
[16] D. Stolee, "Isomorph-free generation of 2-connected graphs with application," *CSE Technical Reports, University of Nebraska-Lincoln* (2011).
[17] A. Dharwadker and J. T. Tevet, "The graph isomorphism algorithm," *In Proc. Structure Semiotics Research Group S.E.R.R.* (2009).
[18] B. D. McKay, "Isomorph-free exhaustive generation," *J. Algorithms* **26**, 306-324 (1998).