Layout Compliance for Triple Patterning Lithography: An Iterative Approach

Bei Yu[†], Gilda Garreton[‡], and David Z. Pan[†]

[†]ECE Dept. University of Texas at Austin, Austin, TX, USA
[‡]Oracle Labs, Oracle Corporation, Redwood Shores, CA, USA
Email: {bei, dpan}@cerc.utexas.edu, gilda.garreton@oracle.com

Abstract

As the semiconductor process further scales down, the industry encounters many lithography-related issues. In the 14nm logic node and beyond, triple patterning lithography (TPL) is one of the most promising techniques for Metal1 layer and possibly Via0 layer. As one of the most challenging problems in TPL, recently layout decomposition efforts have received more attention from both industry and academia. Ideally the decomposer should point out locations in the layout that are not triple patterning decomposable and therefore manual intervention by designers is required. A traditional decomposition flow would be an iterative process, where each iteration consists of an automatic layout decomposition step and manual layout modification task. However, due to the NP-hardness of triple patterning layout decomposition, automatic full chip level layout decomposition requires long computational time and therefore design closure issues continue to linger around in the traditional flow. Challenged by this issue, we present a novel incremental layout decomposition framework to facilitate accelerated iterative decomposition. In the first iteration, our decomposer not only points out all conflicts, but also provides the suggestions to fix them. After the layout modification, instead of solving the full chip problem from scratch, our decomposer can provide a quick solution for a selected portion of layout. We believe this framework is efficient, in terms of performance and designer friendly.

1. INTRODUCTION

As the feature size of semiconductor process nodes further scales down, the industry is looking for new lithography techniques, e.g., multiple patterning lithography (MPL), extreme ultra violet (EUV), electron beam lithography (EBL), and directed self-assembly (DSA).[1,2] MPL is a viable solution for emerging logic nodes, as other candidates suffer from either throughput problem or yield issues.[3,4] Multiple patterning lithography consists of double patterning lithography (DPL),[5] triple patterning lithography (TPL),[6] or even quadruple patterning lithography (QPL),[7] based on the number of masks utilized.

In MPL manufacturing process, one critical stage, *layout decomposition*, is usually adopted to divide the initial layout patterns into multiple masks. Then each mask is implemented through one exposure-etch process, through which the layout can be produced. In initial layout, two features with distance less than minimum coloring distance d_{min} should be assigned into different masks. One *conflict* occurs when two features whose spacing is less than d_{min}. Sometimes the conflict can be also resolved by inserting *stitch* to split a pattern into

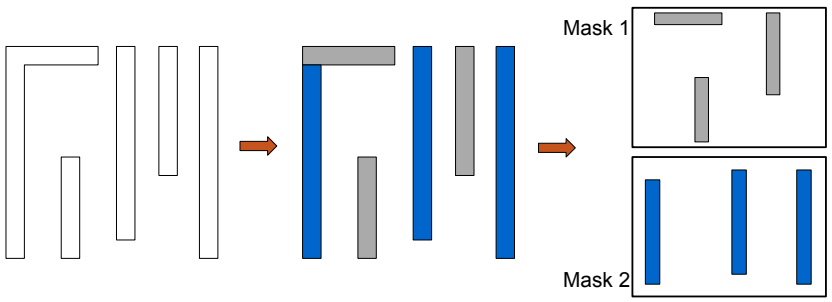

Figure 1. Double patterning lithography (DPL) layout decomposition.

Photomask Technology 2014, edited by Paul W. Ackmann, Naoya Hayashi, Proc. of SPIE
Vol. 9235, 923504 · © 2014 SPIE · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2066034

two touching parts. However, the introduced stitches lead to yield loss due to overlay error,[8] thus two of the main objectives in layout decomposition are conflict minimization and stitch minimization. An example of DPL layout decomposition is shown in Fig. 1, where two different masks are represented by two different colors.

Triple patterning lithography (TPL), which is a natural extension from DPL, is one of the most viable solutions for 14nm node.[9] ITRS roadmap[10] shows that if EUV is not ready, triple patterning lithography is a viable solution for sub-14nm logic node to introduce better printability.[9] Industry has already explored the test-chip patterns with triple patterning or even quadruple patterning.[11] In addition, triple even quadruple patterning is being considered for use on the middle-of-line (MOL) layers in the 10nm technology node and beyond.[12] Like in double patterning, the key challenge of TPL is the layout decomposition. Fig. 2 demonstrates one example of such TPL layout decomposition, where all patterns in input layout are divided into three different masks (colors).

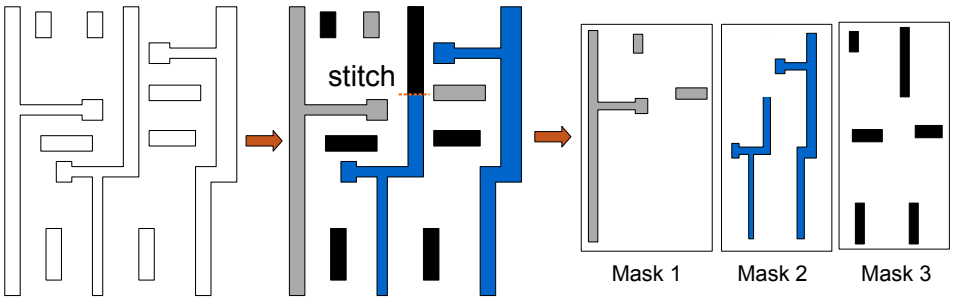

Figure 2. Triple patterning layout decomposition.

Triple patterning or even quadruple patterning layout decomposition problem with conflict and stitch minimization has been well studied in the past few years.[6,7,13–21] For example, Cork et al. proposed a three coloring algorithm adopting SAT formulation.[13] Yu et al. presented integer linear programming (ILP) based method to optimally solve the problem, and semidefinite programming (SDP) based algorithms to provide a faster solution.[6,19] Ghaida et al. reused the double patterning techniques.[14] Fang and Kuang proposed different heuristic strategies.[17,22] However, most existing work suffers from one or more of the following drawbacks.

- First, due to the NP-hardness of TPL layout decomposition,[6] most of the decomposers are based on approximation or heuristic methods, thus some extra conflicts may be reported.
- Second, successfully carrying out these decomposition techniques requires the input layouts to be TPL-friendly. How to achieve the final layout compliance, that the decomposed layout is conflict free, is still an open question.

Figure 3 demonstrates two common native TPL conflicts in via layer (see Fig. 3 (a)) and Metal1 layer (see Fig. 3 (b)), respectively. Breaking any conflict edge in the four-clique structure can introduce triple patterning friendly. However, conventional layout decomposition tools usually only report a random conflict edge, which limits the potential to remove the native conflict. From designers' perspective, a useful layout decomposer should be able to label all native conflicts and provide layout modification suggestions. In this paper, we study the **triple patterning layout compliance** problem, where layout modification and layout color assignment are iteratively carried out to achieve final layout compliance.

The rest of the paper is organized as follows. In Section 2 we provide the problem formulation. In Section 3 we discuss the new challenges stemmed from triple patterning, and the limitations of current triple patterning layout decomposition works. In Section 4 and Section 5 we provide the details of our proposed layout compliance flow. Section 6 presents the experimental results, and we conclude this paper in Section 7.

2. PROBLEM FORMULATION

In this section we will provide the problem formulation of triple patterning layout compliance, and also discuss its difference with the conventional layout decomposition problem.

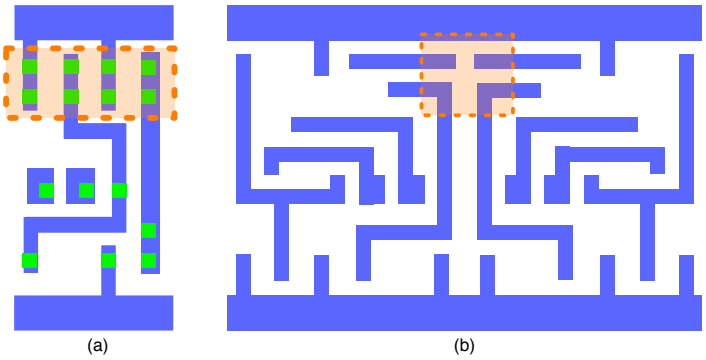

Figure 3. Two native conflicts (in read boxes) from (a) via layer; (b) Metal1 layer. (Source: [23])

Given input layout which is specified by features in polygonal shapes, a **decomposition graphs**[6,24] is constructed by Definition 1.

Definition 1 (Decomposition Graph). *A decomposition graph is an undirected graph* $\{V, CE, SE\}$ *with a single set of vertices* V*, and two edge sets* CE *and* SE *containing the conflict edges (CE) and stitch edges (SE), respectively. Each vertex* $v \in V$ *represents a polygonal shape, an edge* $e \in CE$ *exists iff the two polygonal shapes are within minimum coloring distance* d_{min}*, and an edge* $e \in SE$ *iff there is a stitch candidate between the two vertices which are associated with the same polygonal shape.*

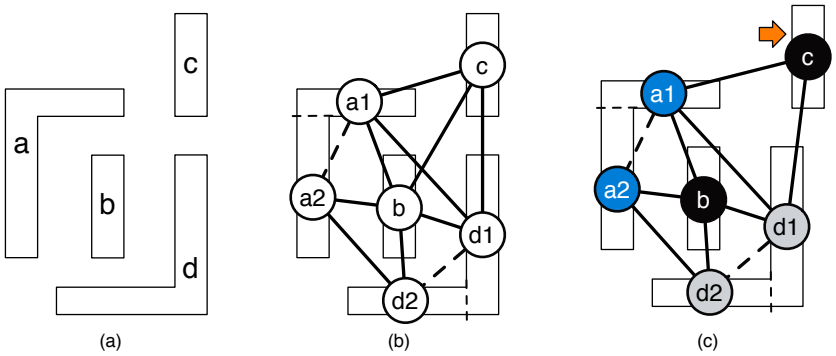

Figure 4. (a) Input layout and all stitch candidates; (b) Decomposition graph; (c) Color assignment of the decomposition graph.

We use Fig. 4 for illustration. Firstly, vertex projection is performed on input layout to search all stitch candidates. Then decomposition graph (see Fig. 4 (b)) is constructed to store all the layout information and the stitch candidates. Note that in vertex projection we choose all the possible stitch candidates as described in by J. Kuang and E. F. Young in Ref. [17].

Based on the decomposition graph, we give the problem formulation of triple patterning layout compliance.

Problem 1 (Triple Patterning Layout Compliance). *Given an input layout which is specified by features in polygonal shapes and minimum coloring distance* d_{min}*, the decomposition graph is constructed. Layout compliance seeks to modify layout and assigns all the vertices into one of three colors (masks) to remove all conflicts.*

At first glance, the layout compliance problem is similar to conventional *layout decomposition* problem, that all vertices in the decomposition graph are assigned into one of three colors. However, as shown in Fig. 5, the new aspect in layout compliance is twofold: (1) The layout decomposition does not modify the layout, while in layout compliance problem we can modify layout to clean up some triple patterning native conflicts. (2) The layout decomposition minimizes the conflict number, and left the conflicts to designers to fix. While in layout compliance problem, the output should be guaranteed to be conflict free.

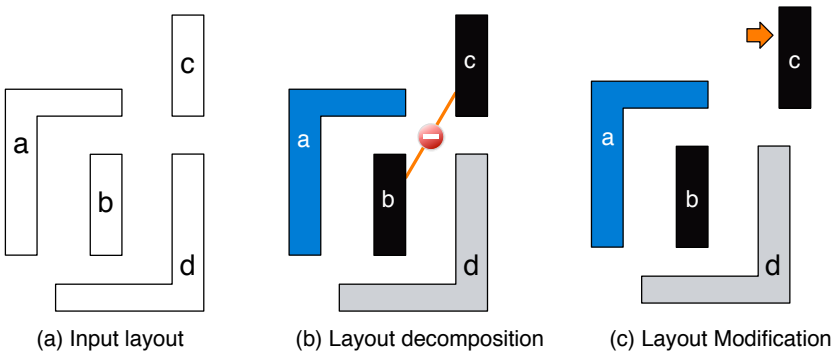

Figure 5. An example of layout compliance process, which insists of layout decomposition and layout modification.

3. NEW CHALLENGES IN TRIPLE PATTERNING

Triple patterning is a natural extension along the paradigm of double patterning, thus the manufacturing process in triple patterning can be easily extended from double patterning. However, from design or layout synthesis perspective, triple patterning does introduce many new challenges. In this section, we discuss some challenges stemmed from triple patterning, and compare them with the double patterning counterparts. We will also discuss the limitations of state-of-the-art layout decomposers.

3.1 No Shortcut Any More

As discussed above, two major objectives of layout decomposition are conflict minimization and stitch minimization. Generally speaking, optimizing conflict and stitch simultaneously is NP-hard problem for both double patterning and triple patterning.[6,25] Here NP-hard problem is a set of computational search problems that are difficult to solve, and no NP-hard problem can be solved in polynomial time in the worst case under the assumption that $P \neq NP$.[26] In other words, we cannot find an effective methods to optimally solve layout decomposition for either double patterning or triple patterning.

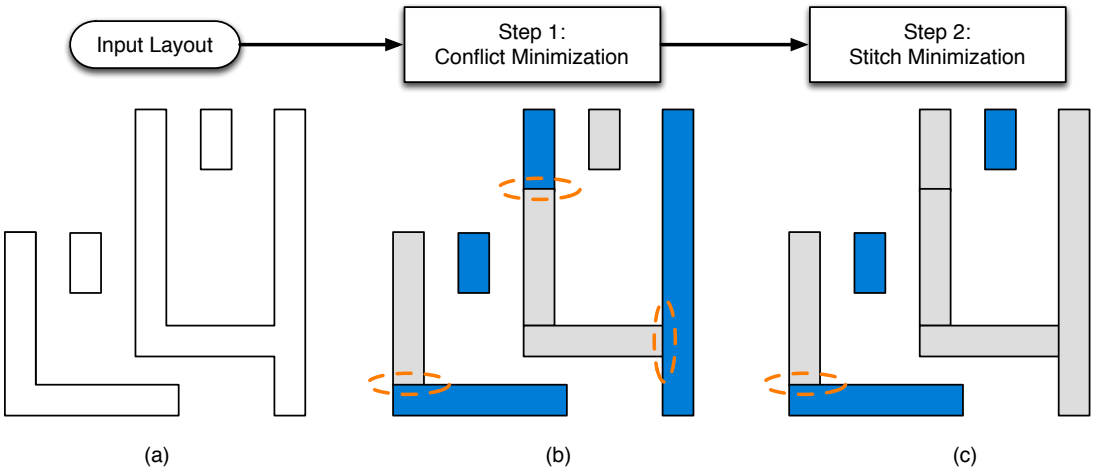

Figure 6. Effective two-step double patterning layout decomposition flow.

Fortunately, for double patterning we have a shortcut. That is, if we optimize conflict number and stitch number step by step, then each step can be solved optimally in polynomial time.[5,27] For example, Fig. 6 illustrates one example of double patterning layout compliance flow. Given input layout in Fig. 6 (a), the first step of layout compliance is to clean up all conflicts [see Fig. 6 (b)]. During conflict clean up, one can either insert stitches or shift layout patterns. Since all conflicts in double patterning can be detected through graph breath first search,[5] such clean up can guarantee to remove all conflicts effectively. The second step is to remove all un-necessary stitches, or so called stitch number minimization. The stitch minimization can be solved through

a planer graph bi-partitioning.[27,28] For example, given the three stitches in Fig. 6 (b), after stitch minimization there is only one stitch left [see Fig. 6 (c)]. Tang et al. has demonstrated the effectiveness of such layout compliance flow in industry design.[27]

However, for triple patterning layout compliance, there is no shortcut any more. The reason is that even detecting whether a layout can be solved using three colors is NP-hard.[6] Therefore, the layout compliance flow for double patterning and triple patterning would be quite different. Triple patterning layout decomposers have to solve conflict minimization and stitch minimization together, which is very difficult and time consuming.

3.2 Where do the Conflicts Come From?

In double patterning, the layout decomposition problem is very similar to graph two coloring problem. Checking whether a graph is two-colorable equals to checking whether there is odd-cycle in the graph. Here a cycle with an odd number of vertices is called an odd cycle. Since odd-cycle is common in a conflict graph, we can easily detect the double patterning conflict sources. Besides, as shown in Fig. 7 such cycles can be stemmed from a very long pattern chain. Therefore, there are many choices to break down the odd-cycle in a graph.

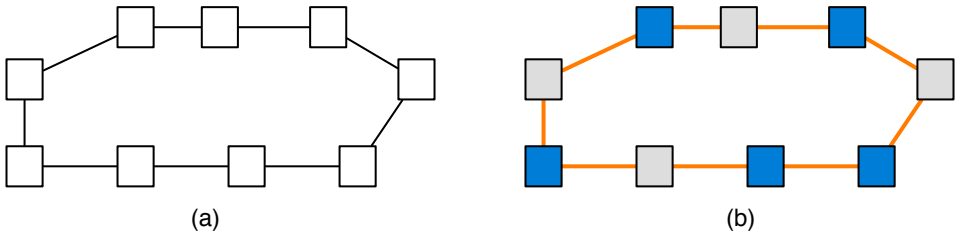

Figure 7. Odd cycle is the only source of double patterning conflicts. Some odd cycles may be from very long pattern chain.

However, most of triple patterning conflicts, especially those native conflicts, come from very local four-clique structures. For example, Fig. 8 illustrates a four-clique conflict among these four patterns. No matter how to assign the colors, there is at least one conflict. Since the conflict sources are very local, usually there may be only few options to break the four-clique structures.

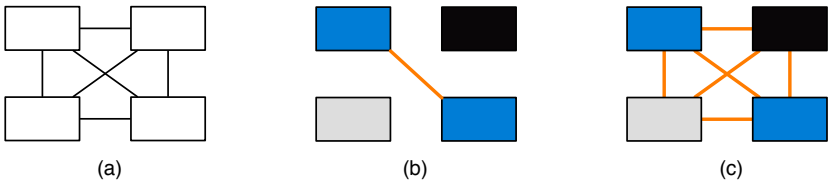

Figure 8. Four clique structure is the sources of most triple patterning conflicts.

3.3 Decomposer: Clutching at Straws

Although triple patterning layout decomposition has been well studied in recent years, there is still a huge gap between the decomposer performance and the expected performance required by designers. For example, Fig. 9 depicts the performance of the different triple patterning decomposition algorithms. Integer linear programming (ILP) can search layout decomposition solutions with minimum conflict number. However, since ILP is a classical NP-hard problem, i.e., there is no polynomial time optimal algorithm to solve it,[26] for large layout cases this method may suffer from long runtime penalty.[6] The greedy or heuristic methods are fast, but their solutions may not provide good quality in term of number of conflicts.[7,15] There are some other approaches trying to achieve some trade-off between runtime and solution quality. However, their solutions may not provide the required quality.[6,20]

Figure 9 points out the performance target of a layout decomposer in dark zoom. We can see that no conventional layout decomposition work can satisfy the performance requirement, due to the following possible reasons. (1) The layout decomposition needs to achieve minimum conflict number in short computational time.

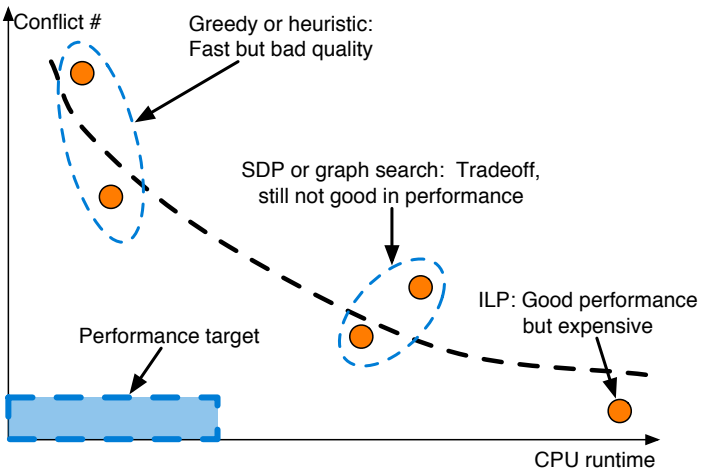

Figure 9. TPL layout decomposer status.

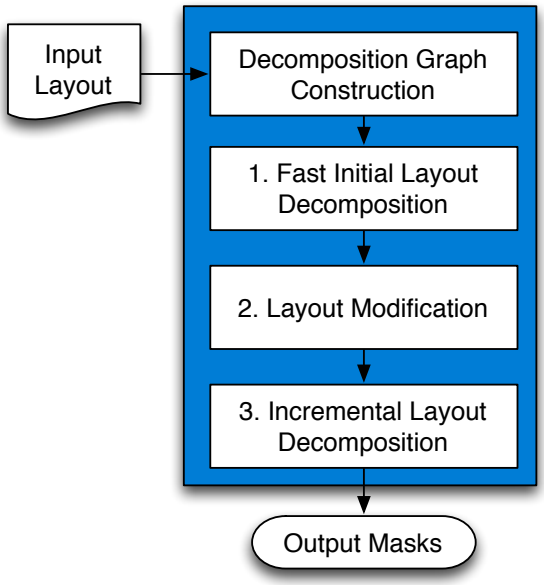

Figure 10. The proposed fast and effective triple patterning layout compliance flow.

However, due to the NP-hardness, currently no methodology can achieve good trade-off between runtime and solution quality; (2) So far all works are limited to layout decomposition, and no layout modification is considered. Therefore, conventional layout decomposition may suffer from large amount of conflict number, due to some native conflict structures.

4. LAYOUT COMPLIANCE OVERALL FLOW

In this paper we propose a novel and effective layout compliance framework, where both layout decomposition and layout modification is considered. The overall flow of our layout compliance is illustrated in Fig. 10. We first construct a decomposition graph to transform the original geometric patterns into a graph model. By this way, the layout decomposition can be formulated as three coloring on the decomposition graph. Then a fast initial layout decomposition is carried out the assign all vertices in the decomposition graph some colors. Besides, our initial layout decomposition can point out all native conflicts, which would be removed in the layout modification stage. After the native conflict removal, the updated layout is triple patterning friendly, but there might be still

some conflicts reported by initial layout decomposition. Finally, a set of incremental layout decompositions are proposed to clean all conflicts through locally color re-assignment.

Figure 11 illustrates one example of our triple patterning layout compliance flow.

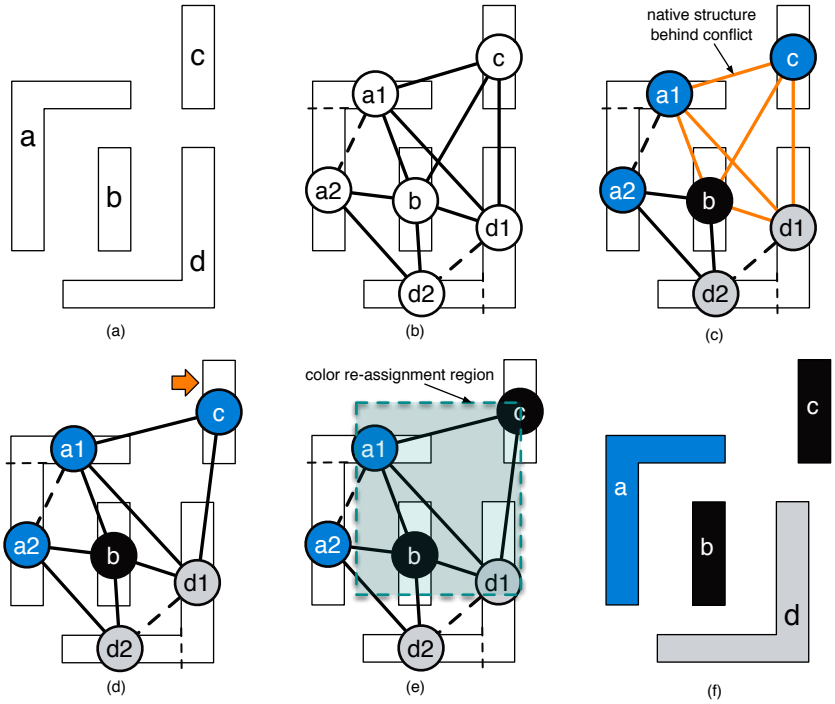

Figure 11. An example of triple patterning layout compliance flow.

5. LAYOUT COMPLIANCE ALGORITHMS

In this section we provide the algorithmic details of our triple patterning layout compliance framework.

5.1 Fast Initial Layout Decomposition

Our first step is a fast initial layout decomposition to provide a color assignment to the decomposition graph. Our layout decomposition is inspired by the linear layout decomposer in Ref. [7]. The runtime comparison between our decomposer and two conventional methodologies, integer linear programming (ILP)[6] and semidefinite programming (SDP)[19] is shown in Fig. 12. ILP based method is very expensive to solve, thus for some large or complex design cases, its runtime would be unacceptable. SDP is faster than ILP, but for the cases with more than 5000 decomposition graph nodes, it can also take around eight hours to search solution. We can see that due to linear runtime complexity, our initial layout decomposition is extremely fast.

To keep the linear runtime complexity, the color assignment of all vertices would be carried out one by one. However, color assignment through one specific order may get stuck at *local optimum* which stems from the greedy nature. To alleviate the impact of vertex ordering, we propose two strategies.

The first strategy is called *color-friendly rules*. For example, given a decomposition graph in Fig. 13 (a), if the coloring order is *a*-*b*-*c*-*d*, when vertex c is greedily selected grey color, the following vertex d cannot find any color without conflict (see Fig. 13 (b)). If color-friendly rules are applied, in Fig. 13 (c), all conflict neighbors of pattern d are labeled inside a grey box. Since the distance between a and c is within the range of $(d_{min}, d_{min} + hp)$, a is color-friendly to c. Here hp is the half-pitch value. Interestingly, we discover a rule that for a complex/dense layout, color-friendly patterns tend to be with the same color.

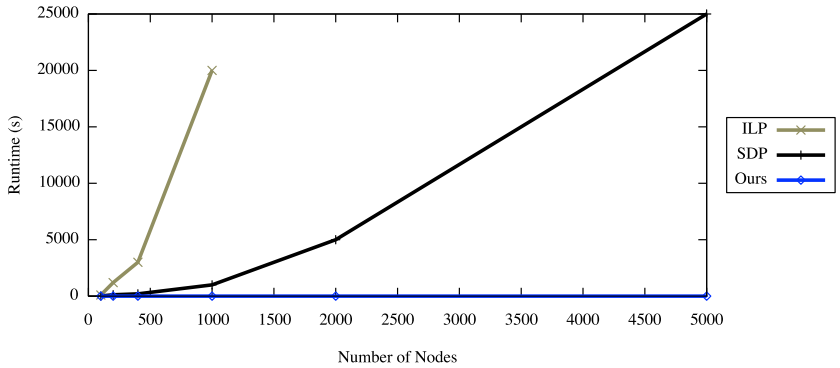

Figure 12. Compare our decomposition runtime with ILP and SDP based methods.

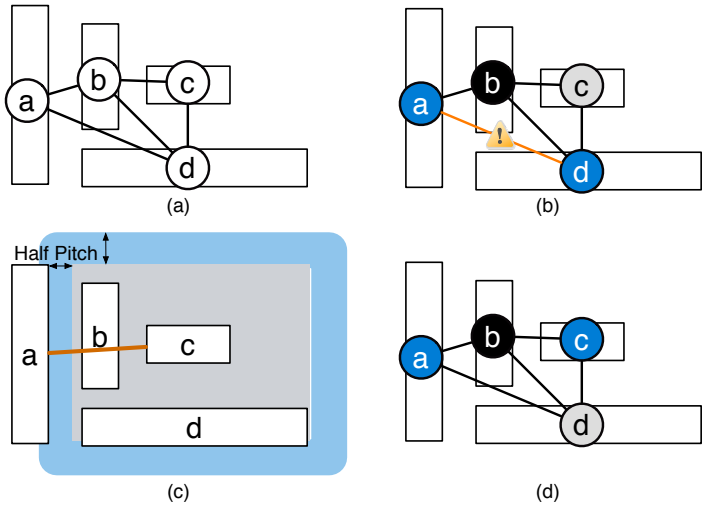

Figure 13. Color-friendly rule. (a) Input layout patterns; (b) Greedy color assignment may stuck at local optimum; (c) Label patterns a and c as color-friendly; (d) Coloring result considering color-friendly rules.

Our second strategy is called *peer selection*, where three different vertex orders would be processed simultaneously, and the best one would be selected as the final coloring solution. Although color assignment is solved thrice, since for each order the coloring is in linear time, the total computational time is still linear.

Although the initial layout decomposition is fast, its quality is not good enough that many conflicts may be reported. Some conflicts would be removed through layout modification (see Sec. 5.2), while some other conflicts would be handled through incremental layout decomposition (see Sec. 5.3).

5.2 Layout Modification: Native Conflict Removal

Our second step is layout modification to remove all the native conflicts. As shown in Fig. 14, our initial layout decomposition can report all four-clique structures. Therefore, the designers can identify the layout patterns that need to be modified. Fig. 15 gives one example how we shift one pattern to remove the four-clique structure.

5.3 Incremental Layout Decomposition

The last step is incremental layout decomposition, where we resolve each remained conflict through re-assigning the colors of its neighborhood patterns. During the incremental layout decomposition, the vertex colors in decomposition graph may be changed, thus incremental layout decomposition is also called color re-assignment. Figure 16 gives one example of incremental layout decomposition. In Fig. 16 (a) a decomposed result is shown, and the dashed bounding box is the target to run color re-assignment. In Fig. 16 (b) all layout patterns to be re-assigned colors are labeled as different colors. Figure 16 (c) illustrates the constructed local decomposition graph, where different colors represent different vertices, and one edge means two vertices cannot be assigned

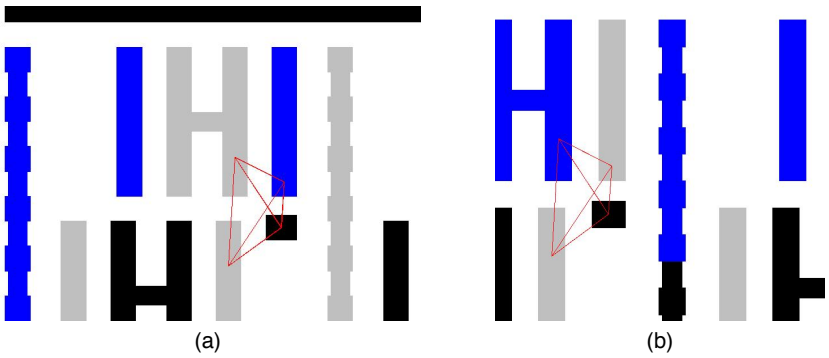

Figure 14. Labeled native conflicts through initial layout decomposition.

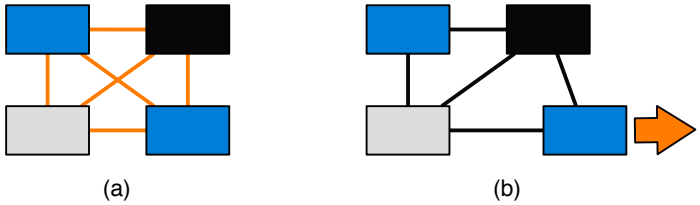

Figure 15. An example of local conflict removal.

same color. The result of incremental layout decomposition is shown in Fig. 16 (d). Our color re-assignment is based on backtracking algorithm.[29] Note that although the decomposition graph in Fig. 16 (c) looks complicated, the backtrack based color re-assignment can be finished in one second.

6. EXPERIMENTAL RESULTS

The new algorithms are implemented in C++ and interfaced with the open source VLSI CAD tool called *Electric*[30] used to test the decomposition flow. Tests have been executed on an Intel® Xeon® Processor E3-1245 (8M Cache, 3.40 GHz, 16GB RAM). Figure 17 shows an example of four-clique structure removal interfaced through the CAD tool Electric. Figure 18 shows the flow working for a cache array used in a 32-bit MIPS microprocessor designed with Electric by students @ Harvey Mudd College.[31]

7. CONCLUSIONS

In this paper we have proposed a triple patterning layout compliance flow for full chip level design. Our flow consists of several iterations to provide a fast conflict free decomposed layout. We believe this paper will stimulate more future research into the layout compliance problem, thereby facilitating the advancement of multiple patterning lithography technique.

REFERENCES

[1] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 10, pp. 1453–1472, 2013.

[2] B. Yu, J.-R. Gao, D. Ding, Y. Ban, J.-S. Yang, K. Yuan, M. Cho, and D. Z. Pan, "Dealing with IC manufacturability in extreme scaling," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 240–242.

[3] C. Wagner and N. Harned, "EUV lithography: Lithography gets extreme," *Nature Photonics*, vol. 4, no. 1, pp. 24–26, 2010.

[4] K. Yuan, B. Yu, and D. Z. Pan, "E-Beam lithography stencil planning and optimization with overlapped characters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 2, pp. 167–179, Feb. 2012.

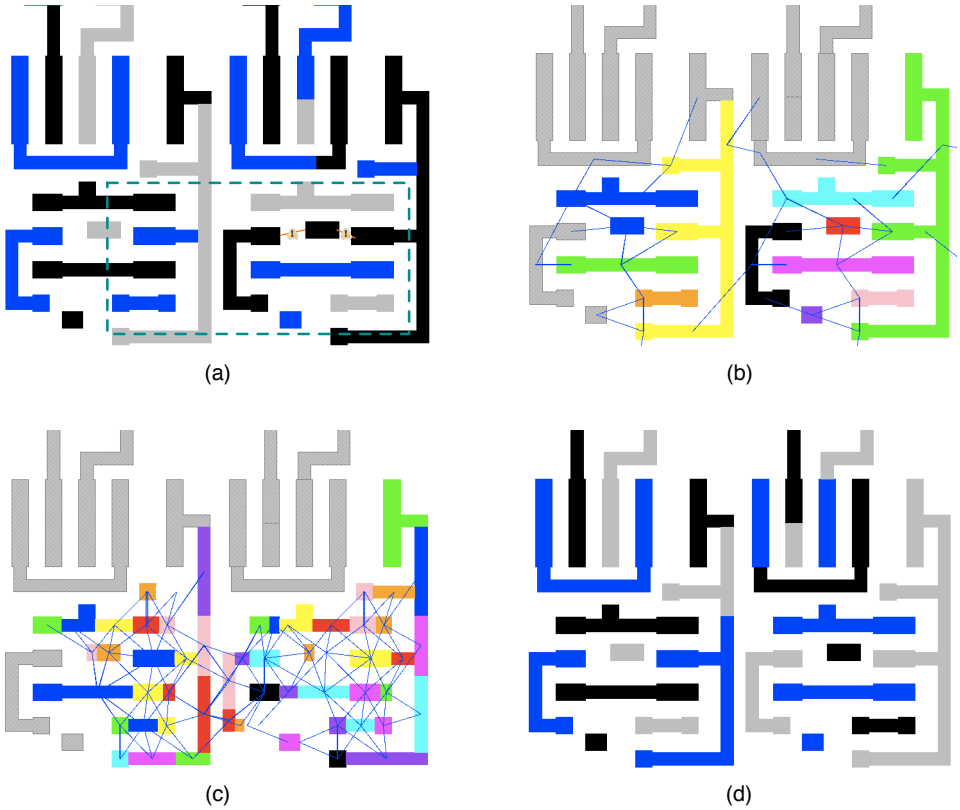

Figure 16. An example of incremental layout decomposition. (a) Decomposed result after initial layout decomposition. The dashed bounding box is the target to run color re-assignment. (b) All layout patterns to be re-assigned colors are labeled as different colors. (c) The constructed local decomposition graph, where different colors represent different vertices, and one edge means two vertices cannot be assigned same color. (d) The result of incremental layout decomposition.

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

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

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

[8] V. Wiaux, S. Verhaegen, S. Cheng, F. Iwamoto, P. Jaenen, M. Maenhoudt, T. Matsuda, S. Postnikov, and G. Vandenberghe, "Split and design guidelines for double patterning," in *Proc. of SPIE*, vol. 6924, Feb 2008.

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

[10] "ITRS," http://www.itrs.net.

[11] Y. Borodovsky, "Lithography 2009 overview of opportunities," in *Semicon West*, 2009.

[12] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan, "A systematic framework for evaluating cell level middle-of-line (MOL) robustness for multiple patterning," in *Submitted to Proc. of SPIE*, 2015.

[13] C. Cork, J.-C. Madre, and L. Barnes, "Comparison of triple-patterning decomposition algorithms using aperiodic tiling patterns," in *Proc. of SPIE*, vol. 7028, 2008.

[14] R. S. Ghaida, K. B. Agarwal, L. W. Liebmann, S. R. Nassif, and P. Gupta, "A novel methodology for triple/multiple-patterning layout decomposition," in *Proc. of SPIE*, vol. 8327, 2012.

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

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

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

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

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

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

[21] W. Fang, S. Arikati, E. Cilingir, M. A. Hug, P. De Bisschop, J. Mailfert, K. Lucas, and W. Gao, "A fast triple-patterning solution with fix guidance," in *Proc. of SPIE*, vol. 9053, 2014, pp. 90 530A–90 530A.

[22] S.-Y. Fang, Y.-W. Chang, and W.-Y. Chen, "A novel layout decomposition algorithm for triple patterning lithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 3, pp. 397–408, March 2014.

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

[24] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *ACM International Symposium on Physical Design (ISPD)*, 2009, pp. 107–114.

[25] J. Sun, Y. Lu, H. Zhou, and X. Zeng, "Post-routing layer assignment for double patterning." in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2011, pp. 793–798.

[26] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. Cambridge, MA, USA: MIT Press, 1990.

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

[28] J.-S. Yang, K. Lu, M. Cho, K. Yuan, and D. Z. Pan, "A new graph-theoretic, multi-objective layout decomposition framework for double patterning lithography," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2010, pp. 637–644.

[29] R. Neapolitan and K. Naimipour, *Foundations of algorithms*. Jones & Bartlett Publishers, 2010.

[30] "Electric," http://www.staticfreesoft.com/.

[31] http://www.staticfreesoft.com/electricGallery.html.

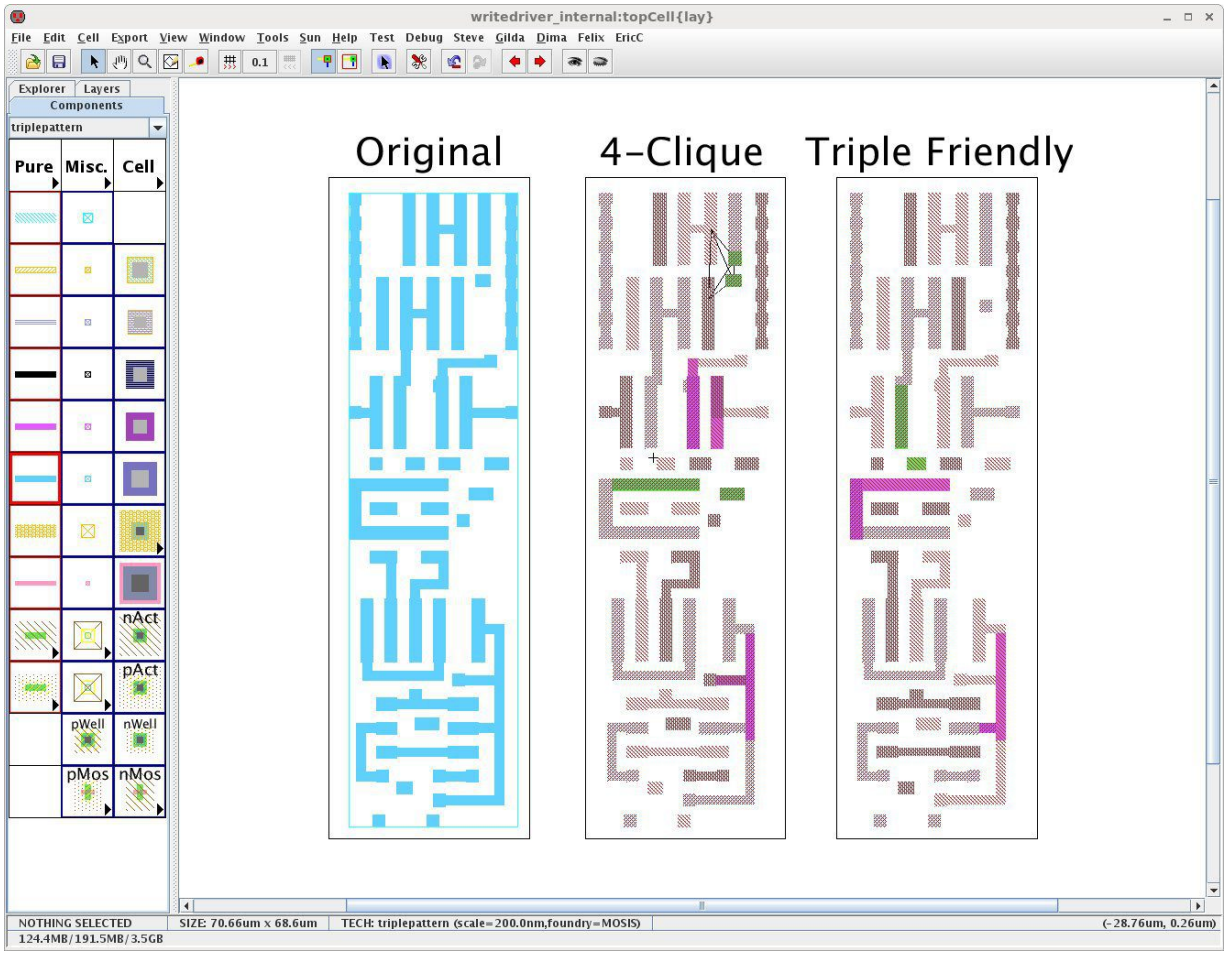

Figure 17. An example of four-clique structure removal using Electric.

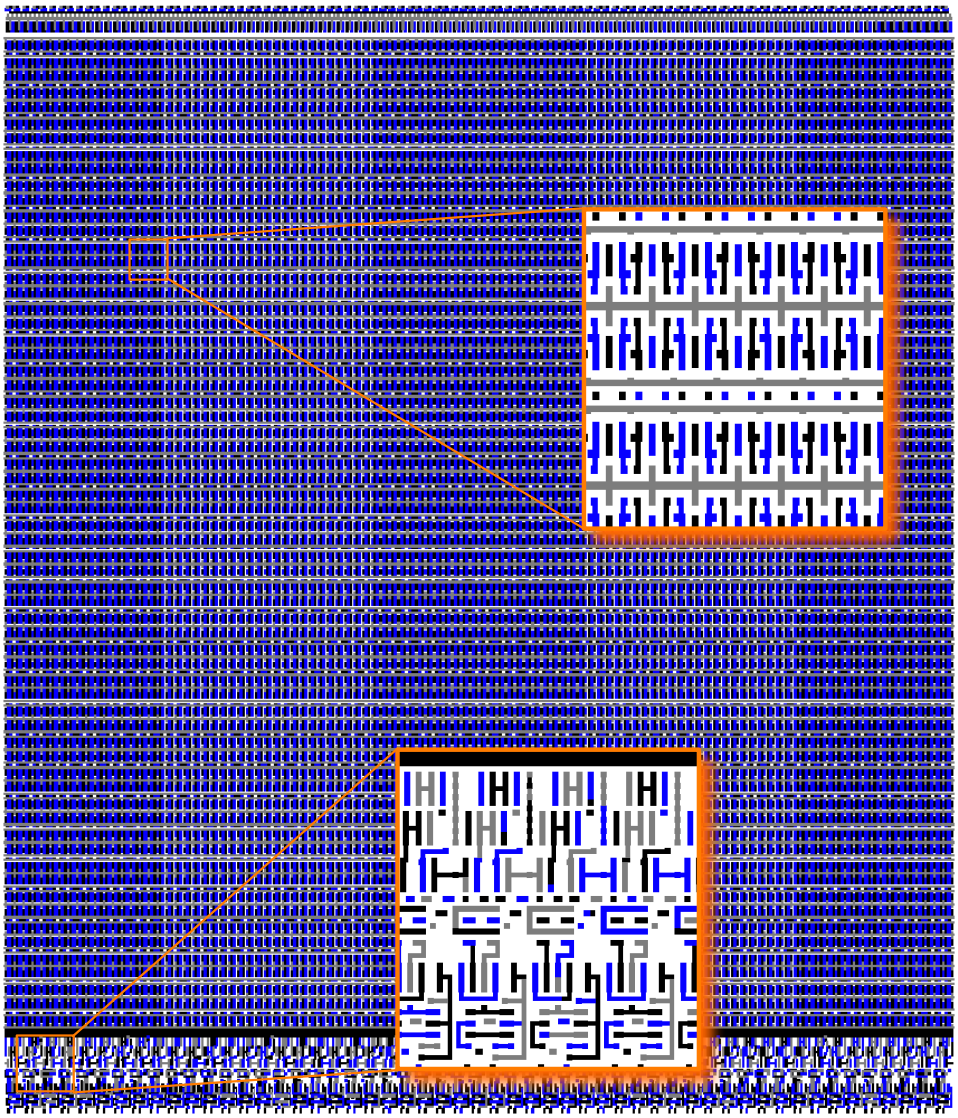

Figure 18. Output of our triple patterning layout compliance framework.