Triple Patterning Lithography (TPL) Layout Decomposition using End-Cutting

Bei Yu, Jhih-Rong Gao, and David Z. Pan

ECE Dept. University of Texas at Austin, Austin, TX USA 78712
Email: {bei, jrgao, dpan}@cerc.utexas.edu

ABSTRACT

Triple patterning lithography (TPL) is one of the most promising techniques in the 14nm logic node and beyond. However, traditional LELELE type TPL technology suffers from native conflict and overlapping problems. Recently LELEEC process was proposed to overcome the limitations, where the third mask is used to generate the end-cuts. In this paper we propose the first study for LELEEC layout decomposition. Conflict graphs and end-cut graphs are constructed to extract all the geometrical relationships of input layout and end-cut candidates. Based on these graphs, integer linear programming (ILP) is formulated to minimize the conflict number and the stitch number.

1. INTRODUCTION

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)[1] is one of the most promising techniques,[2] because of the delay of other candidates, such as extreme ultraviolet lithography (EUVL) and electron beam lithography (EBL). EUVL is challenged by tremendous technical barriers,[3] while EBL has a serious limitation due to low throughput.[4,5]

One traditional process of TPL is with the same principle of litho-etch-litho-etch (LELE) type double patterning lithography (DPL), where the original layout is decomposed into three masks and manufactured through three exposure/etching steps. This technology is called LELE-litho-etch (LELELE). Many research has been carried out to solve the corresponding design problems, including layout decomposition[6–10] and routing.[11,13] However, even with stitch insertion, there are some native conflicts in LELELE. Fig. 1 shows a 4-clique conflict among features a, b, c, and d. Since this 4-clique structure is common in advanced standard cell design, LELELE type TPL still suffers from the native conflict problem.[7] Besides, compared with LELE, there are more serious overlapping problem in LELELE.[14]

To overcome all these limitations derived from LELELE, recently Lin[15] proposes a new TPL manufacturing process, called LELE-end-cutting (LELEEC). As a TPL, this new manufacturing process contains three mask steps, namely first mask, second mask, and *trim* mask. Fig. 2 illustrates an example of the LELEEC process. To generate target features in Fig. 2(b), the first and second masks are used for pitch splitting, which is similar

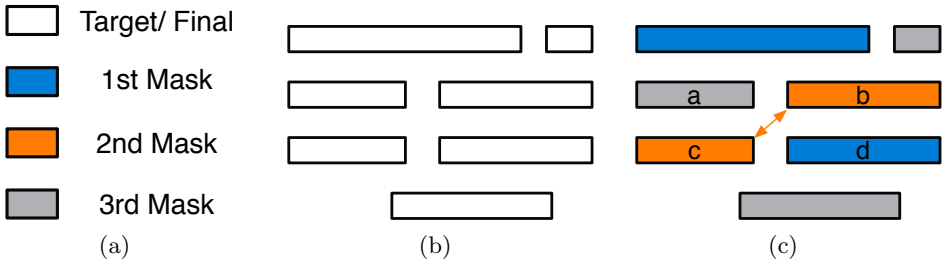


Figure 1. Process of LELELE (a) Some labels. (b) Target features. (c) layout decomposition with one conflict introduced.

Design for Manufacturability through Design-Process Integration VII, edited by Mark E. Mason, John L. Sturtevant,
Proc. of SPIE Vol. 8684, 86840G · © 2013 SPIE · CCC code: 0277-786X/13/$18 · doi: 10.1117/12.2011355

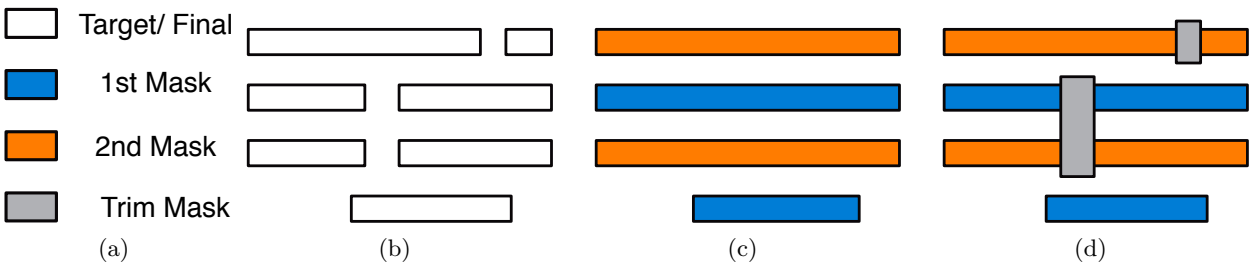


Figure 2. Process of LELEEC (a) Some labels. (b) Target features. (c) First and second mask patterns. (d) Trim mask, and final decomposition with no conflict.

to LELE process in DPL. These two masks are shown in Fig. 2(c). Finally, a trim mask is used to trim out the desired region as shown in Fig. 2(d). In other words, the trim mask is used to generate some end-cuts to further split feature patterns. As a result, the features that are not LELELE-friendly can be printed without introducing any conflict through the LELEEC process. Besides, the end-cuts introduce better printability.[?]

Layout decomposition, which is the most crucial step for TPL, has been extremely studied under LELELE process.[6–9] However, some new design challenges are introduced in LELEEC process that previous layout decomposition methodology cannot be directly borrowed. Since the end-cuts are printed through one mask, some end-cuts may not be compatible to each other. How to design the trim mask is pretty crucial, and therefore the layout decomposition for LELEEC is still an open problem.

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

The rest of the paper is organized as follows. In Section 2, we provide some preliminary, and discuss the problem formulation. In Section 3 we explain the details to generate the end-cut candidates. In Section 4 we provide the integer linear programming formulation for this problem, and followed by several speedup techniques in Section 5. Section 6 presents the experimental results, and we conclude this paper in Section 7.

2. PRELIMINARY AND PROBLEM FORMULATION

2.1 Layout Graph

Given a layout which is specified by features in polygonal shapes, layout graphs[7] are constructed. As shown in Fig. 3, the layout graph is an undirected graph with a set of vertices V and a set of conflict edges CE. Each vertex in V represents one input feature. There is an edge in CE if and only if the two features are within minimum coloring distance dis_m of each other. In other words, each edge in CE is a conflict candidate. Fig. 3(a) shows one input layout, and the corresponding layout graph is in Fig. 3(b). For each edge (conflict candidate), we check whether there is an end-cut candidate. End-cut candidates (grey rectangles in Fig. 3(c)) are introduced to input layout. For each end-cut candidate $i - j$, if it is applied, then features i and j will be merged into one feature. By this way the corresponding conflict edge can be removed. We label all edges in layout graph, to indicate those conflict edges that can be removed by end-cut insertion (see blue solid edges in Fig. 3(d)). If stitch is considered in layout decomposition, some vertices in layout graph can be split into several segments. The segments in one layout graph vertex are connected through *stitch edges*. A set SE contains all these stitch edges *.

2.2 End-Cut Graph

Since all the end-cuts are manufactured through one single exposure process, they should be distributed far away from each other. That is, two end-cuts have conflicts if they are within minimum end-cut distance dis_c

*Please refer to[16] for the details of stitch candidate generation.

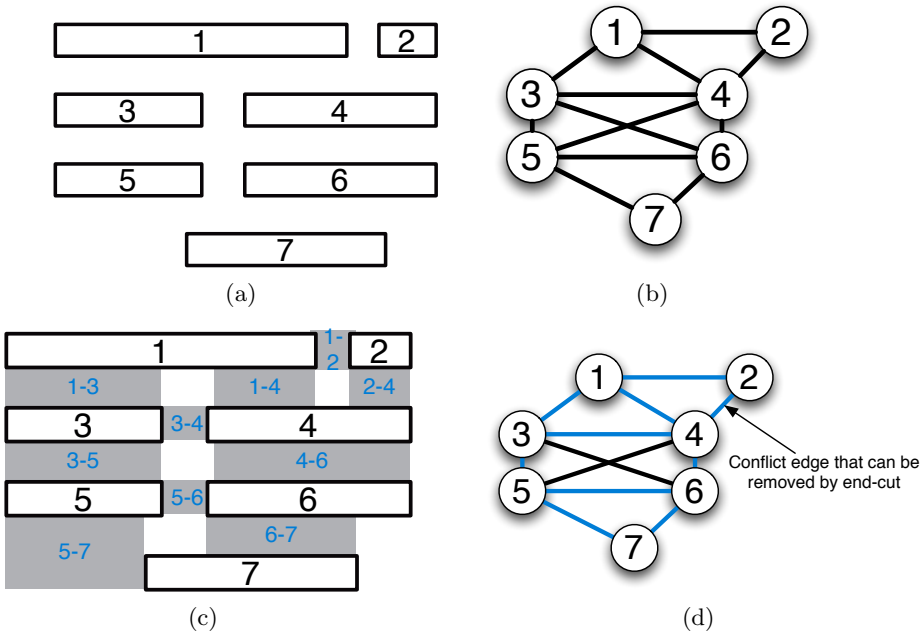


Figure 3. Layout graph construction (a) Input layout. (b) Layout graph with conflict edges. (c) End-cut candidate generation, where each grey rectangle represents one end-cut candidate. (d) Update layout graph where each blue edge means that the conflict can be removed by introducing one end-cut.

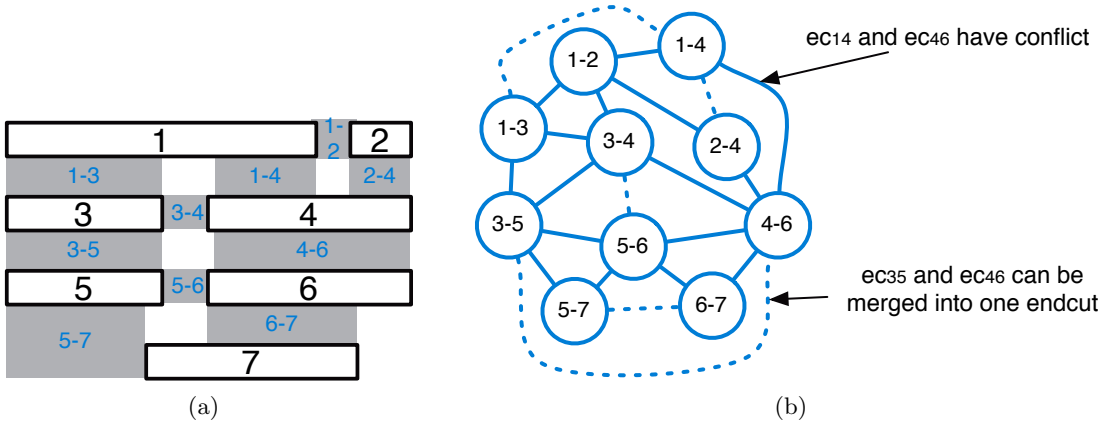


Figure 4. End-cut graph construction. (a) Input layout with end-cut candidates. (b) End-cut graph.

of each other. Note that these conflict relationships among end-cuts are not available in layout graph, therefore we construct *end-cut graph* to store the relationships. Fig. 4(a) shows an example of an input layout with all end-cut candidates; the corresponding end-cut graph is shown in Fig. 4(b). Each vertex in the graph represents one end-cut. There is an solid edge if and only if the two end-cuts conflict to each other. There is an dash edge if and only if they are close to each other, and they can be merged into one larger end-cut.

2.3 Problem Formulation

Given a layout which is specified by features in polygonal shapes, the layout graph and the end-cut graph are constructed. The layout decomposition assigns all vertices in layout graph into one of two colors, and select a set of end-cuts in end-cut graph. The objectives it to minimize the number of conflict and/or stitch.

3. END-CUT CANDIDATE GENERATION

Above we have discussed how to generate the layout graph and end-cut graph. In this section we will explain the details to generate all end-cut candidates. For each conflict edge in layout graph, there are two adjacent features in corresponding input layout. To check whether one end-cut can be inserted, we classify the adjacent relationships of features into two types: *edge-edge* and *corner-corner*. The end-cut between two edge-edge features can be calculated through calculating the projection, as illustrated in Fig. 5. Note that in some cases, if the width of one end-cut is smaller than wire width threshold w_{th}, this end-cut candidate is forbidden (see Fig. 5(d)).

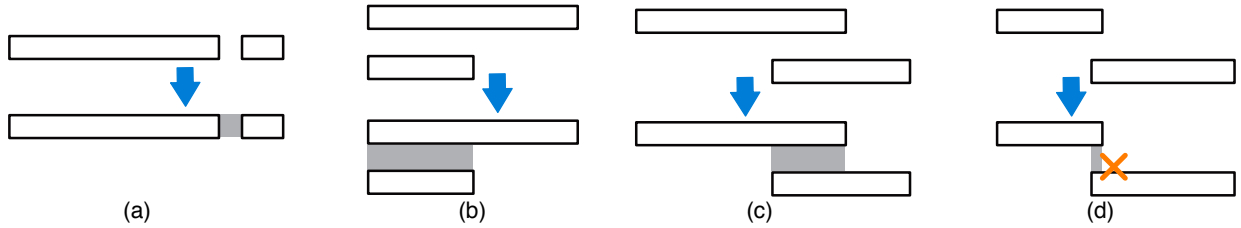


Figure 5. End-cut candidate generation for edge-edge type.

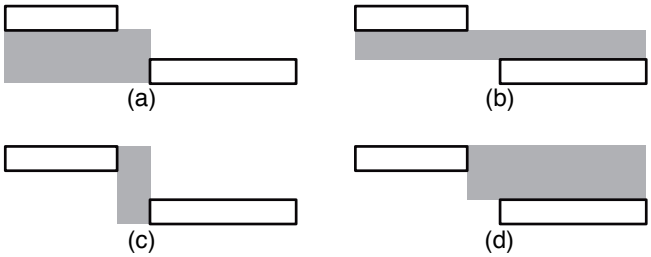


Figure 6. End-cut candidate generation for corner-corner type.

If one end-cut candidate cannot be constructed through edge-edge type, we search whether there is one legal corner-corner type end-cut. As shown in Fig. 6, there are four possible end-cut shapes. We pick up the shape with minimal area, meanwhile the shape width should be larger than width threshold w_{th}. For each end-cut candidate, if it is overlapping with any layout feature, it would be forbidden.

4. ILP ALGORITHMS

After the construction of layout graph and end-cut graph, LELEEC layout decomposition problem can be transferred into an optimization problem on graphs. Note that conflict graph can not be guaranteed to be planar, so some face based methodology[17] cannot be applied here. Therefore, we formulate integer linear programming (ILP) to solve the optimization problem. For convenience, some notations in the ILP formulation are listed in Table 1.

Table 1. Notations in ILP

CE	set of conflict edges
EE	set of end-cut conflict edges
SE	set of stitch edges.
r_i	the i_{th} layout feature
x_i	variable denoting the coloring of r_i
ec_{ij}	0-1 variable, $ec_{ij} = 1$ when the end-cut between r_i and r_j
c_{ij}	0-1 variable, $c_{ij} = 1$ when a conflict between r_i and r_j
s_{ij}	0-1 variable, $s_{ij} = 1$ when a stitch between r_i and r_j

4.1 ILP Formulation without Stitch Insertion

When no stitch candidate is considered, the layout decomposition problem can be formulated as shown in Eq. (1). The objective is to minimize the conflict number.

$$\min \sum_{e_{ij} \in CE} c_{ij} \tag{1}$$

$$\begin{aligned}
\text{s.t.} \quad & x_i + x_j \leq 1 + c_{ij} + ec_{ij} && \forall e_{ij} \in CE && (1a)\\
& (1 - x_i) + (1 - x_j) \leq 1 + c_{ij} + ec_{ij} && \forall e_{ij} \in CE && (1b)\\
& ec_{ij} + ec_{pq} \leq 1 && \forall e_{ijpq} \in EE && (1c)\\
& ec_{ij} + x_i - x_j \leq 1 && \forall e_{ij} \in CE && (1d)\\
& ec_{ij} + x_j - x_i \leq 1 && \forall e_{ij} \in CE && (1e)
\end{aligned}$$

where x_i is a variable for the colors of feature r_i, c_{ij} is a binary variable for conflict edge $e_{ij} \in CE$. ec_{ij} is a binary variable for end-cut candidate. Constraints $(1a)$ and $(1b)$ are used to evaluate the conflict number and end-cut number. If two adjacent features r_i and r_j are assigned same colors (masks), and the end-cut candidate ec_{ij} is not applied ($ec_{ij} = 0$), then there is one conflict reported ($c_{ij} = 1$). If end-cuts ec_{ij} and ec_{pq} are conflict with each other, constraint $(1c)$ makes sure that at most one of them will be applied. Constraints $(1d)$ and $(1e)$ are used to forbid useless end-cut. That is, if features x_i and x_j are in different colors, $ec_{ij} = 0$.

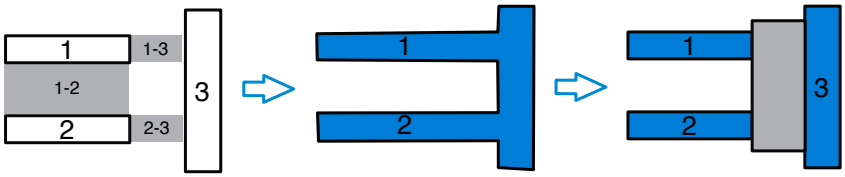


Figure 7. End-cut candidate generation for corner-corner type.

At first glance, the ILP formulation (1) works well. However, it may report some unnecessary conflicts. An example is shown in Fig. 7, where three end-cut candidates $ec_{12}, ec_{13}, ec_{23}$ are considered among features r_1, r_2 and r_3. Note that ec_{13} and ec_{23} could be merged into one large end-cut. If these two end-cuts are applied, all features can be manufactured through one feature (see Fig. 7(b)). However, based on the constraints $(1a)$ and $(1b)$, since x_1 equals to x_2, and end-cut ec_{12} is not applied, conflict c_{12} will be reported. We can see that our initial ILP formulation ignore this wire merging character. To overcome this limitation, we modify the constraints $(1a)$ and $(1b)$ as follows:

$$\begin{aligned}
x_i + x_j &\leq 1 + c_{ij} + ec_{ij} + ec_{ik} \cdot ec_{jk}\\
(1 - x_i) + (1 - x_j) &\leq 1 + c_{ij} + ec_{ij} + ec_{ik} \cdot ec_{jk}
\end{aligned} \tag{2}$$

Since $ec_{ik} \cdot ec_{jk}$ is non-linear, we introduce boolean variables $\gamma_{ik,jk}$ to replace it, and enforce the following artificial constraints in our ILP formulation:

$$\begin{aligned}
ec_{ik} + ec_{jk} &\leq \gamma_{ik,jk} + 1\\
ec_{ik} &\geq \gamma_{ik,jk}\\
ec_{jk} &\geq \gamma_{ik,jk}
\end{aligned} \tag{3}$$

After replacing constraints (2) with (3), our formulation can be linearized. Therefore, the modified ILP formulation is as follows:

$$\min \sum_{e_{ij} \in CE} c_{ij} \tag{4}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 + c_{ij} + ec_{ij} + \gamma_{ik,jk} \qquad \forall e_{ij} \in CE \tag{4a}$$

$$(1 - x_i) + (1 - x_j) \leq 1 + c_{ij} + ec_{ij} + \gamma_{ik,jk} \qquad \forall e_{ij} \in CE \tag{4b}$$

$$(1c) - (1e), (3)$$

4.2 ILP formulation with Stitch Insertion

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

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

$$\text{s.t.} \quad x_i - x_j \leq s_{ij} \qquad \forall e_{ij} \in SE \tag{5a}$$

$$x_j - x_i \leq s_{ij} \qquad \forall e_{ij} \in SE \tag{5b}$$

$$(1c) - (1e), (3), (4a) - (4b)$$

5. SPEEDUP TECHNIQUES

ILP is a well-known NP-hard problem that it may suffer from long runtime penalty to achieve the results. In this section, we provide a set of speedup techniques. Note that these techniques can keep optimality. In other words, with these speedup techniques, ILP formulation can achieve the same results comparing with those not applying speedup.

5.1 Independent Component Computation

The first speedup technique is called *independent component computation*. By breaking down the whole layout graph into several independent components, we partition the initial layout graph into several small ones. Then each component can be resolved through ILP formulation independently. At last, the overall solution can be taken as the union of all the components without affecting the global optimality. Note that this is a well-known technique which has been applied in many previous studies (e.g.,[16,18,19]).

5.2 Bridge Computation

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

5.3 End-Cut Pre-Selection

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

6. EXPERIMENTAL RESULTS

We implement our algorithms in C++ and test on an Intel Xeon 3.0GHz Linux machine with 32G RAM. ISCAS 85&89 benchmarks from[7] are used. GUROBI[20] is chosen as the ILP solver. We set the minimum coloring distance dis_m as $2W_{min} + 3S_{min}$, where W_{min} and S_{min} denote the minimum wire width and the minimum spacing, respectively. The width threshold w_{th}, which is used in end-cut candidate generation, is set as dis_m.

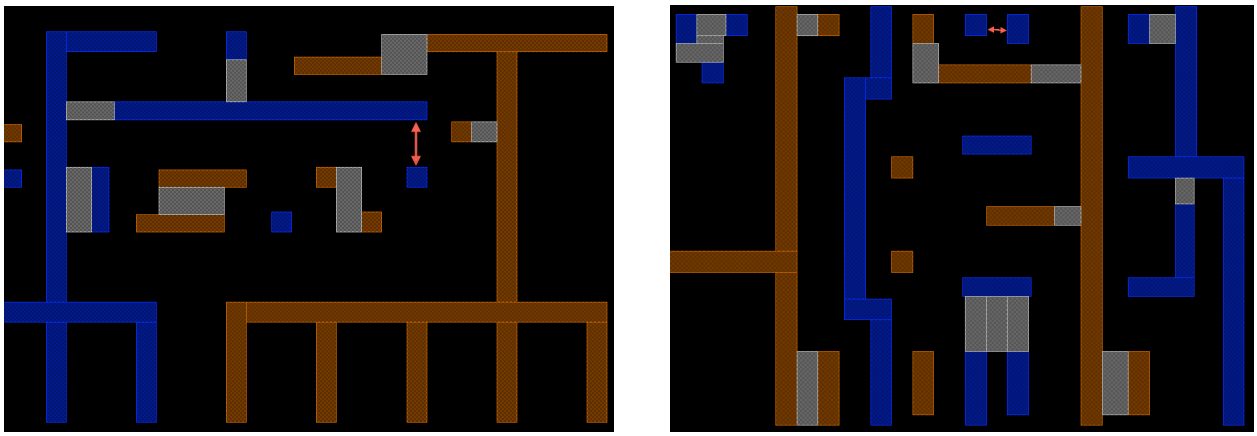
Table 2. Comparison of Runtime and Performance

Circuit	Wire#	Sub-G#	ILP w/o. stitch				ILP w. stitch			
			conflict#	stitch#	cost	CPU(s)	conflict#	stitch#	cost	CPU(s)
C432	1109	30	0	0	0	1.66	0	0	0	1.77
C499	2216	64	0	0	0	3.7	0	0	0	4.15
C880	2411	102	0	0	0	5.3	0	0	0	5.56
C1355	3262	104	0	0	0	5.5	0	0	0	5.72
C1908	5125	155	1	0	1	8.4	0	0	0	8.62
C2670	7933	299	0	0	0	16.2	0	0	0	16.75
C3540	10189	417	0	0	0	22.5	0	0	0	23.33
C5315	14603	601	0	0	0	32.8	0	0	0	34.01
C6288	14575	475	9	0	9	27.5	8	11	9.1	32.54
C7552	21253	788	0	0	0	44.3	0	0	0	45.57
S1488	4611	194	0	0	0	10.3	0	0	0	10.73
S38417	67696	2285	3	0	3	149.6	0	1	0.1	159.68
S35932	157455	4469	47	0	47	411.3	5	1	5.1	380.73
S38584	168319	5659	14	0	14	502.3	2	0	2	477.27
S15850	159952	5417	16	0	16	473.9	5	4	5.4	452.58
avg.	-	-	6	0	6	114.3	1.33	1.13	1.45	110.6
ratio	-	-	4.5	0	**1.0**	**1.0**	1.0	1.0	**0.24**	**0.97**

6.1 With or Without Stitch

In the first experiment, we show the decomposition results of the ILP formulation. Table 2 compares two ILP formulations "ILP w/o. stitch" and "ILP w. stitch". Here "ILP w/o. stitch" is the ILP formulation based on the graph without stitch edges, while "ILP w. stitch" considers the stitch insertion in the ILP. Columns "Wire#" and "Sug-G#" reports the total feature number, and the divided sub-graph number, respectively. For each method we report the conflict number, stitch number, and computational time in seconds("cpu(s)"). "Cost" is the cost function, which is set as conflict# +0.1× stitch#. From Table 2 we can see that both ILP formulation is effective that only a few conflicts are reported. Compared with "ILP w/o. stitch", when stitch candidates are considered in the ILP formulation, the cost can be reduced by 76%, while the runtime is similar.

Fig. 8 shows two conflict examples in decomposed layout, where conflict pairs are labeled with red arrows. We can observe that both of the two conflicts come from via shapes. One possible reason is that it is hard to find end-cut candidates around via, comparing with long wires.



(a) (b)

Figure 8. Conflict examples in decomposed results.

6.2 LELEEC v.s. LELELE

In the next experiment, we compare two TPL type layout decompositions. The state-of-art LELELE decomposer[9] is applied for the comparison. Fig. 9 and Fig. 10 compare the conflict number and the stitch number under two lithography processes, respectively. We can observe that through applying end-cutting (LELEEC), both the conflict and stitch number are reduced dramatically. The reasons are mainly twofold: (1) 4-clique conflict, which is a common type in standard layout, can be resolved in LELEEC; (2) Compared with wire shapes, most of the end-cuts are smaller, therefore the trim mask can contain more features.

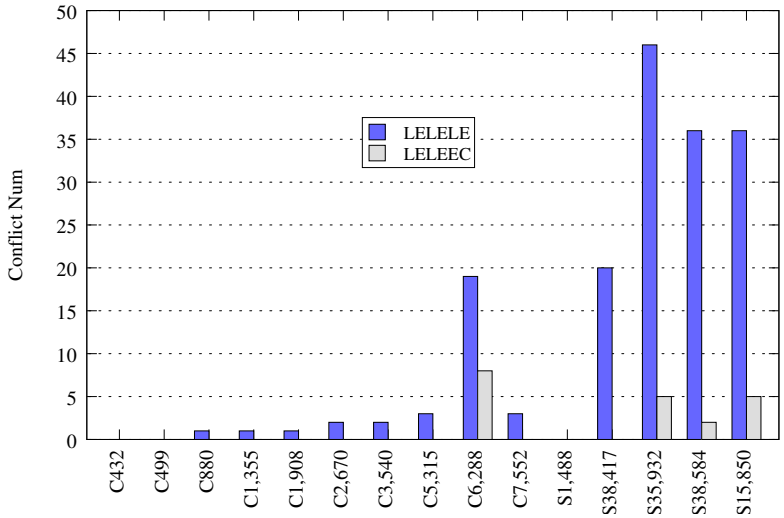


Figure 9. Conflict number comparison

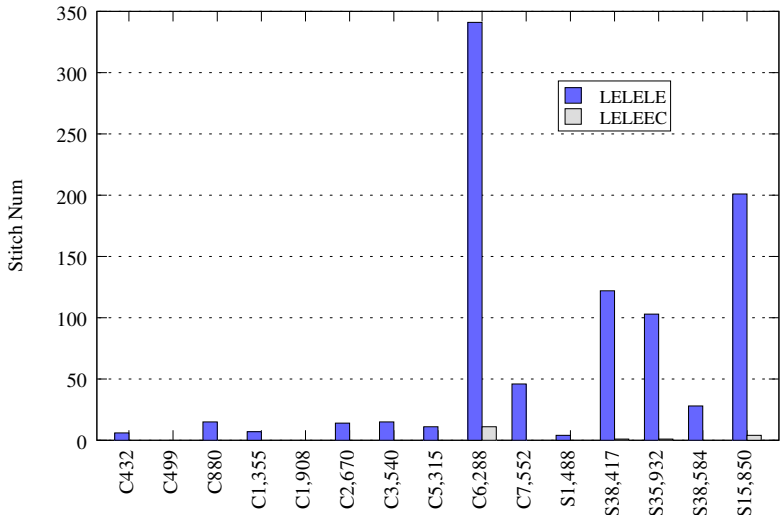


Figure 10. Stitch number comparison

7. CONCLUSION

In this paper, we propose the first study for the LELEEC layout decomposition. The problem is translated into an optimization problem on the layout graph and end-cut graph. Integer linear programming (ILP) is then applied to search the solutions. The experimental results show the effectiveness of our algorithms. In addition,

our preliminary results show that compared with traditional LELELE type TPL, LELEEC can reduce both the conflict number and stitch number dramatically. As LELEEC may be adopted by industry for 14nm/11nm nodes, we believe more research will be needed to enable LELEEC-friendly design and mask synthesis.

Acknowledgment

This work is supported in part by NSF, SRC, Oracle, and NSFC.

REFERENCES

[1] 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.

[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] Y. Arisawa, H. Aoyama, T. Uno, and T. Tanaka, "EUV flare correction for the half-pitch 22nm node," in *Proc. of SPIE*, vol. 7636, 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.

[5] B. Yu, J.-R. Gao, and D. Z. Pan, "L-Shape based layout fracturing for e-beam lithography," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2013.

[6] 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.

[7] 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.

[8] 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, 2011.

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

[10] 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.

[11] Q. Ma, H. Zhang, and M. D. F. Wong, "Triple patterning aware routing and its comparison with double patterning aware routing in 14nm technology," in *ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 591–596.

[12] P. Lei and X. Ying, "Thermoelectric Heat Pump Drying Temperature Control System on the Basis of 89C51", in *International Conference onComputer Science and Electronics Engineering*, 2012.

[13] Y.-H. Lin, B. Yu, D. Z. Pan, and Y.-L. Li, "TRIAD: A triple patterning lithography aware detailed router," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012.

[14] C. Ausschnitt and P. Dasari, "Multi-patterning overlay control," in *Proc. of SPIE*, vol. 6924, 2008.

[15] B. J. Lin, "Lithography till the end of moore's law," in *ACM International Symposium on Physical Design (ISPD)*, 2012.

[16] 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.

[17] Y. Xu and C. Chu, "A matching based decomposer for double patterning lithography," in *ACM International Symposium on Physical Design (ISPD)*, 2010, pp. 121–126.

[18] 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.

[19] 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.

[20] "GUROBI," http://www.gurobi.com/html/academic.html.