Triple and Quadruple Patterning Layout Decomposition via Iterative Rounding and Linear Programming Solving

Yibo Lin, Xiaoqing Xu, Bei Yu, Ross Baldick and David Z. Pan

***Abstract*—As the semiconductor technology scales down to $10nm$ node and beyond, multiple patterning has become a competitive lithography candidate, along with other emerging techniques including EUVL, E-Beam and Directed Self Assembly (DSA). Due to the delay of EUVL, triple even quadruple patterning may be used for the front-end of line layers and lower metal layers with tight pitches in future technology nodes. To enable multiple patterning, layout decomposition, as the key step, aims at splitting the layout into various parts such that each part can be manufactured using single patterning. In this work, the triple and quadruple patterning layout decomposition issue is first formulated as the integer linear programming (ILP) problem. In order to deal with the large problem size, an iterative rounding and LP solving scheme (IRLS) is proposed to reduce the number of non-integers in the final solutions. Moreover, novel stitch insertion strategies are proposed to further control the amount of conflicts and stitches in the decomposition results.**

I. Problem Formulation and Overall Flow

A. Previous Works

Due to the resolution limits of the $193nm$ lithography tools, multiple patterning (MP) has become a viable candidate to enable the geometric scaling for advanced technology nodes [1]–[4]. The basic principle for MP layout decomposition is to split the original layout into several different masks to achieve smaller pitches than the resolution limits of the $193nm$ photolithography.

For general layout decomposition problem, the conflict graph is constructed from the original layout [5]. Each node in the graph represents a polygon feature in the layout. Then, a conflict edge is added between a pair of nodes if the corresponding features can not be assigned to the same mask and a stitch edge is added if a pair of nodes belong to the same polygon feature before the stitch candidate generation and conflict graph construction. Thus, the layout decomposition issue is reduced to the conflict graph coloring problem and the number colors allowed is equal to the number of masks for the layout decomposition. For instance, we have 3 and 4 colors for triple patterning (TP) and quadruple patterning (QP) lithography, respectively. The layout decomposition example for TP and QP are demonstrated in Fig. 1(a)-(b) and Fig. 1(c)-(d), respectively. A stitch is introduced in Fig. 1(a) to resolve coloring conflicts and achieve legal mask assignment in Fig. 1(b). Similarly, QP decomposition results are illustrated in Fig. 1(c) and (d).

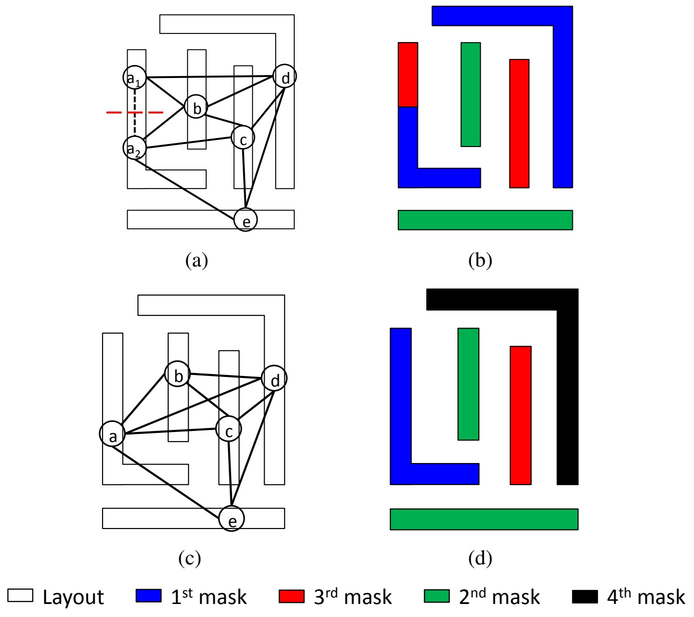


Fig. 1. Multiple patterning layout decomposition, (a) conflict graph with one stitch, (b) triple patterning layout decomposition, (c) conflict graph without stitches, (d) quadruple patterning layout decomposition.

The layout decomposition problem has been well studied for double patterning (DP) [5]–[10] and triple patterning (TP) [11]–[16].

The ILP formulation for layout decomposition is first studied for DP [6], [17] and further extended to TP [11]. The colors are encoded with binary variables to in the ILP formulation and the objective is to minimize the total cost of the conflicts and stitches. Although some graph simplification methods have been proposed to speedup ILP, it still suffers from runtime in the real application. In particular, the integer linear programming (ILP) solution for QP layout decomposition problem has been ignored due to the NP-completeness of the mathematical formulation and large problem size.

Recently, [18] proposed layout decomposition algorithms for QP and beyond, which introduced semi-definite approximation to trade-off with exact algorithms. Other than that, few works have been done for the QP layout decomposition. While semi-definite programming (SDP) based algorithms produce good solution qualities, the runtime becomes an obstacle for its wide usage due to the increasing layout sizes.

In this work, we propose an algorithm to solve the ILP based QP layout decomposition problem via an Iterative Rounding and Linear programming Solving (IRLS) scheme [19]. In general, if an ILP formulation is relaxed to a linear programming problem, it can be solved much faster, usually in polynomial time. Some of the non-integers within the LP solution are rounded to 0/1 based on our rounding schemes. Then, the integers in the solution will be treated as constants and the LP problem is solved iteratively until no further improvement on solution quality. We expect for better decomposition results compared with previous works [18].

B. Problem Formulation

Given the target layout, the first step of layout decomposition is to generate stitch candidates and construct conflict graph with conflict edges and stitches edges. Stitch candidate generation schemes are discussed in detail in [11], [12], [14]. In this work, we assume the stitch candidates are given as the input for our coloring framework. To represent three/four colors in the TP/QP layout decomposition problem, two binary variables are introduced for each node. The ILP formulation is shown in Formulation (1). For each conflict edge in the edge set E_c, the possibility of identical colors on both vertices are forbidden by Constraints (1d)-(1e). Constraint (1a) is only used to eliminate the fourth color for TP layout decomposition. Different from the ILP formulation in [11], additional conflict or stitch edge variables are not introduced for the simplicity of the formulation. Instead of minimizing the total cost from conflicts and stitches, the target of our ILP formulation is to seek a feasible color assignment to the variables while optimizing the changeable objective function. We deal with stitch insertions in a separate step in our coloring framework.

$$
\begin{aligned}
\min \quad & \mathit{Objective} && (1)\\
\text{s.t.} \quad & x_{i1} + x_{i2} \le 1 && (1a)\\
& x_{i1} + x_{i2} + x_{j1} + x_{j2} \ge 1 \quad \forall e_{ij} \in E_c && (1b)\\
& x_{i1} + \bar{x}_{i2} + x_{j1} + \bar{x}_{j2} \ge 1 \quad \forall e_{ij} \in E_c && (1c)\\
& \bar{x}_{i1} + x_{i2} + \bar{x}_{j1} + x_{j2} \ge 1 \quad \forall e_{ij} \in E_c && (1d)\\
& \bar{x}_{i1} + \bar{x}_{i2} + \bar{x}_{j1} + \bar{x}_{j2} \ge 1 \quad \forall e_{ij} \in E_c && (1e)\\
& \bar{x}_{i1} = 1 - x_{i1} \quad \forall i \in V && (1f)\\
& \bar{x}_{i2} = 1 - x_{i2} \quad \forall i \in V && (1g)\\
& x_{i1}, x_{i2} \in \{0, 1\} \quad \forall i \in V && (1h)
\end{aligned}
$$

C. Overall Flow

The overall flow for our coloring framework is demonstrated in Fig. 2. The framework starts with the LP solving with zero objective. To deal with the non-integers in the solution from LP, additional constraints are introduced and objective function is changed for the original LP formulation. In particular, these additional constraints and objective function change will not break the feasibility of possible coloring assignment. We continue the iterative rounding and LP solving (IRLS) until no further improvement. Based on the IRLS solution, novel stitch insertion schemes are proposed to achieve final coloring assignment for each polygon feature.

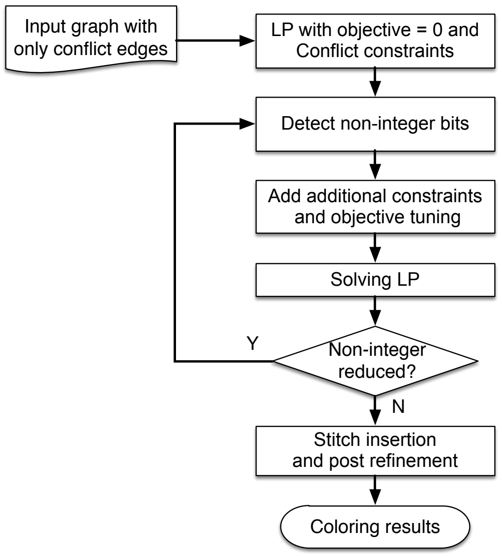

Input graph with only conflict edges
LP with objective = 0 and Conflict constraints
Detect non-integer bits
Add additional constraints and objective tuning
Solving LP
Non-integer reduced?
Y
N
Stitch insertion and post refinement
Coloring results

Fig. 2. Overall flow for our coloring framework.

II. Iterative Rounding and Linear Programming

The details for this section will be added in the final report.

A. Linear Programming Solving

The ILP formulation in (1) is relaxed to LP by removing constraint (1h). The critical issue from LP solving is that it may introduce many non-integers in the solution. For instance, with the constraint (1h), a trivial feasible solution with $x_{i1} = 0.5, x_{i2} = 0.5 \; \forall i \in V$ can be achieved. As shown in Fig. 3, the feasible region for the LP solving is denoted as the dash light green. The dash red line denotes the objective function with optimal value. The grids consisting of dashed black lines are possible solutions with integer bits. We can see that the optimal solution from LP solving is $(0.5, \; 0.5)$. Efficient techniques are needed to push the LP solution to those blue dots in the feasible region with integer while being close to the optimal point.

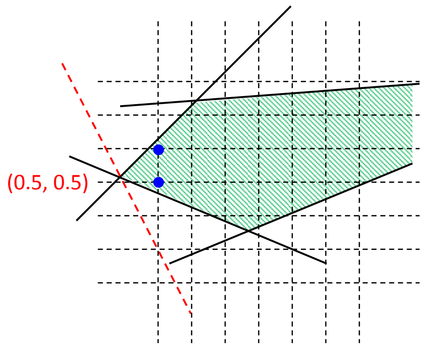


Fig. 3. The polyhedron for feasible linear programming solutions.

B. Iterative Rounding

1) Odd cycle constraints: An odd cycle in a graph needs at least three colors. For the odd cycle example shown as Fig. 4, if the first bits of the vertices are equal, i.e. $x_{i1} = x_{j1} = x_{k1}$, it is not possible to obtain a solution without conflicts by adjusting x_{i2}, x_{j2}, x_{k2}. The LP relaxation will produce all 0.5 solutions for x_{i2}, x_{j2}, x_{k2}. To avoid such kind of solutions, the first bits of the vertices should not be equal. We can avoid the situation of equality of the first bits by adding constraints in Eqn. 2, which forbids the cases of all zeros and all ones. Similar techniques can be applied to the second bits.

$$x_{i1} + x_{j1} + x_{k1} \geq 1 \tag{2a}$$
$$(1 - x_{i1}) + (1 - x_{j1}) + (1 - x_{k1}) \geq 1 \tag{2b}$$

For a general odd cycle ($cycle$), we have the following constraints.

$$\sum_{l \in cycle} x_{l1} \geq 1, \qquad \sum_{l \in cycle} (1 - x_{l1}) \geq 1 \tag{3a}$$
$$\sum_{l \in cycle} x_{l2} \geq 1, \qquad \sum_{l \in cycle} (1 - x_{l2}) \geq 1 \tag{3b}$$

These constraints prune invalid solutions without losing the feasibility of the LP problem.

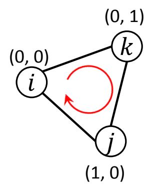


Fig. 4. One possble odd cycle in the conflict graph.

2) Anchoring highest degree vertex: During color assignment, one coloring solution can actually rotate to generate another coloring solution. To reduce the solution space, it will not do harm to the optimality if one vertex of the graph is pre-colored. In the layout of a modern design, power/ground wires are usually assigned with the same color, shown as Fig. 5(a). These wires will be merged into one vertex which eventually results in a high-degree vertex, shown as Fig. 5(b), for power/ground wires are usually very long; As a high-degree vertex has a large set of neighbors, the solution space will be largely reduced if its color is pre-determined. Therefore, when constructing the mathematic formulation, we anchor the color of the vertex with highest degree.

3) Objective function bias: To eliminate the non-integer results in an LP solution, one heuristic is to push the corresponding variables to 0 or 1 by adjusting the objective function. For example, if x_{i1} turns out to be 0.6, it indicates that x_{i1} has the tendency to 1; hence, we add $(1 - x_{i1})$ to the objective function so that x_{i1} tends to be pushed to 1 during the next iteration. The generalized rule is as follows.

1) If $x_i > 0.5$, obj $\leftarrow$ obj $+ (1 - x_i)$.
2) If $x_i < 0.5$, obj $\leftarrow$ obj $+ x_i$.

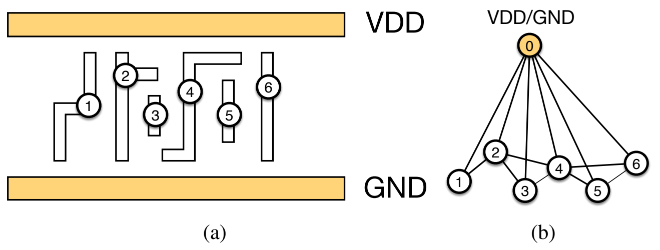


Fig. 5. An example of (a) pre-colored VDD/GND and (b) anchored vertex.

4) Binding constraints analysis: One drawback for the objective function biasing technique cannot handle non-integers like 0.5. Therefore, we propose a method to round those vertex with coloring solution $(x_{i1}, x_{i2}) = (0.5, 0.5)$ pairwisely by analyzing the related binding constraints. For a constraint in LP, if the inequality turns out to be equality according to the LP solution, we call it binding and this constraint is called binding constraint. Fig. 6 shows an example of constraints for a vertex whose solution is $(0.5, 0.5)$. Let S_{i1} be the set of constraints only related to x_{i1}, S_{i2} be the set of constraints only related to x_{i2} and the set of shared constraints are S_{ic}. Assume each constraint is formatted in a way that all variables are on the left side of the inequality operator and only constants are on the right side. At the same time, the coefficient for x_{i1} is positive. If all constraints in S_{i1} share the same kind of operators (all "$\leq$" or "$\geq$"), then these constraints will not be violated if x_{i1} is pushed from 0.5 to 1. The condition also holds for x_{i2} by checking all constraints in S_{i2}. With the analysis above, we can generate a candidate rounded solution for (x_{i1}, x_{i2}). The solution will not be accepted unless the rounded solution also satisfies all constraints in S_{ic}. For the example in Fig. 6, we can generate a candidate rounded solution $(0, 1)$ and then check if constraints in S_{ic} are satisfied as well. If true, (x_{i1}, x_{i2}) are rounded to $(0, 1)$. This technique will not affect the feasibility of the LP.

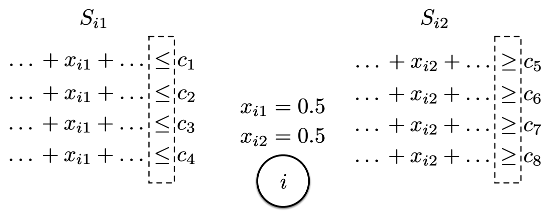

S_{i1}
$\ldots + x_{i1} + \ldots \leq c_1$
$\ldots + x_{i1} + \ldots \leq c_2$
$\ldots + x_{i1} + \ldots \leq c_3$
$\ldots + x_{i1} + \ldots \leq c_4$
$x_{i1} = 0.5$
$x_{i2} = 0.5$
i
S_{i2}
$\ldots + x_{i2} + \ldots \geq c_5$
$\ldots + x_{i2} + \ldots \geq c_6$
$\ldots + x_{i2} + \ldots \geq c_7$
$\ldots + x_{i2} + \ldots \geq c_8$

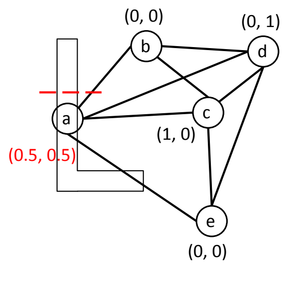
Fig. 6. An example of binding constraints analysis.

III. Stitch Insertion Schemes

A. Stitch Insertion

Non-integer may still exist in the solution from IRLS. With the help from stitch insertions, we can further obtain feasible coloring solutions. An example is demonstrated in Fig. 7. The IRLS solution yields the non-integer bits, namely $(0.5,\ 0.5)$, for node a. However, the stitch candidate for node a can be applied to achieve legal color assignment in Fig. 7(b).

After IRLS, the coloring solution for the graph without stitch edges is mapped back to the original graph which contains both conflict and stitch edges. For vertices with non-integer color assignments in the original graph (usually comes from vertices with stitches), we sort them by the number of conflict edges. Vertices with larger number of conflict edges are greedily assigned legal colors before that with fewer conflict edges.



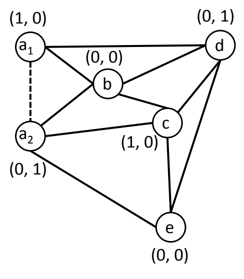


Fig. 7. Stitch insertion to resolve non-integer bits. (a) Non-integer bits from IRLS. (b) Stitch insertion to obtain a feasible coloring solution.

B. Post Refinement

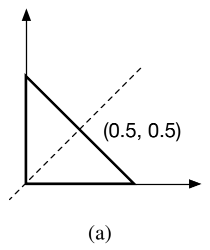
After stitch insertion, it is still possible to improve the results by locally flipping colors. For each vertex pair with conflicts, we check the neighbors of these two vertices and re-assign colors for these two vertices to resolve conflicts. For each vertex pair with stitches, similar approach is applied to resolve stitches.

IV. Experimental Results

We implemented the ILP and IRLS based coloring schemes in C++ and all experiments are performed on the Linux machine with 3.4GHz Intel(R) Core and 32GB memory. Gurobi [20] is used as the ILP and LP solver.

Table I shows the results without stitches. We can see that for small benchmarks (benchmarks with the name start with C) IRLS carries out almost the same number of conflicts as ILP and similar runtime. But for large benchmarks, IRLS produces slightly more conflicts than ILP with smaller runtime. The difference is not very obvious in TPL, but QPL shows the effectiveness of IRLS in the speedup of coloring. For QPL, IRLS achieves more than 200 times speedup than ILP with 1.3% more conflicts.

To explain the reason why IRLS works better in QPL than in TPL, we draw an example of cut planes for the solution spaces, illustrated as Fig. 8. Since TPL only has three colors, the corresponding binary variables are only allowed to be (0, 0), (0, 1) and (1, 0). The solution space is constrained by 45 degree and 135 degree cutlines, because the coefficients for all variables are 1 for all the constraints. The example in Fig. 8(a) shows that with one 45 degree cut line, a non-integer boundary point is generated. However in QPL, both 45 degree and 135 degree cut line are necessary to generate a non-integer boundary point. In other words, it is more likely for TPL to have non-integer boundary points than QPL. Considering that LP solver usually goes through the boundary points and finds a legal solution, the number of non-integer boundary points will affect the solution quality. If there are large number of non-integer boundary points, IRLS needs to spend more iterations to resolve these non-integer solutions as well. This explains why the runtime of IRLS in TPL is very close to that of ILP, while it outperforms ILP significantly in QPL.



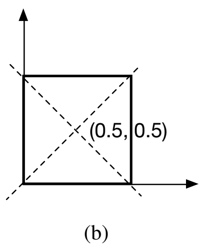


Fig. 8. An example of (a) TPL solution space and (b) QPL solution space.

Table II shows the results when stitch insertion is allowed. Due to the existence of stitches, there is performance degradation for IRLS. For TPL, it produces an average of 37.5% more conflicts and 50% more stitches than ILP with 1.3 times speedup. For QPL, although the speedup is more significant, the performance degradation cannot be ignored. On average, there are about 6 times more conflicts than ILP and 2 times more stitches. We also compare our results with that of SDP [21] in Table II. The SDP approach produces almost the same results as ILP with much smaller runtime. We can see the trade-off between IRLS and SDP in conflicts and stitch numbers and speed.

V. Conclusion

To the best of our knowledge, this is the first work to propose the IRLS based decomposition framework for both TP and QP patterning lithography. Novel stitch insertion schemes are presented to further resolve the coloring conflicts and control the amount of stitches in the decomposition results. The experimental results show that this appoach is very suitable for graphs without stitches in QPL.

References

[1] L. Liebmann, D. Pietromonaco, and M. Graf, “Decomposition-aware standard cell design flows to enable double-patterning technology,” in *Proc. of SPIE*, vol. 7974, 2011.

[2] 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.

[3] 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.

[4] R. Merritt, “Intel sees quad-patterned path to 10-nm chips,” September 2012, http://www.eetimes.com/document.asp?doc_id=1262512.

[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.

TABLE I. COMPARISON ON BENCHMARKS WITHOUT STITCHES

Circuit	ILP for TPL		IRLS for TPL		ILP for QPL		IRLS for QPL	
	cn#	CPU(s)	cn#	CPU(s)	cn#	CPU(s)	cn#	CPU(s)
C432	4	0.077	4	0.067	2	0.094	2	0.07
C499	0	0.109	0	0.112	5	0.197	5	0.124
C880	7	0.132	7	0.121	1	0.166	1	0.123
C1355	3	0.175	3	0.167	4	0.204	4	0.183
C1908	1	0.27	1	0.251	4	0.285	4	0.306
C2670	6	0.382	6	0.379	6	0.463	6	0.428
C3540	9	0.506	9	0.478	3	0.538	3	0.57
C5315	9	0.73	9	0.744	14	0.882	14	0.765
C6288	205	1.209	205	0.899	9	0.874	9	0.788
C7552	22	0.895	22	1.06	15	1.213	15	0.888
S1488	2	0.225	2	0.179	6	0.275	6	0.277
S38417	95	4.876	98	3.704	567	33.018	571	4.173
S35932	157	9.758	162	9.318	1792	240.699	1832	10.555
S38584	230	10.155	236	9.756	1691	107.853	1700	11.473
S15850	212	10.124	213	10.199	1500	8241.31	1511	9.842
avg.	64	2.642	65	2.496	374	575.204	379	2.672

TABLE II. COMPARISON ON BENCHMARKS WITH STITCHES

Circuit	ILP for TPL			IRLS for TPL			TPL [TCAD'15, Yu+]			ILP for QPL			IRLS for QPL			QPL [DAC'14, Yu+]		
	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)
C432	4	0	0.099	8	0	0.066	4	0	0.1	0	2	0.1	0	2	0.058	0	2	0.24
C499	0	0	0.272	0	0	0.111	0	0	0.1	3	1	0.144	4	1	0.104	4	1	0.16
C880	3	0	0.132	5	0	0.119	7	0	0.1	0	1	0.126	1	2	0.105	0	1	0.02
C1355	2	0	0.182	2	0	0.165	3	0	0.1	1	0	0.158	1	0	0.133	4	0	0.1
C1908	0	0	0.257	0	0	0.242	1	0	0.1	2	2	0.565	2	2	0.208	3	2	0.28
C2670	2	0	0.381	4	0	0.337	6	0	0.2	2	0	0.329	3	0	0.296	6	0	0.16
C3540	4	1	0.519	4	1	0.483	9	1	0.4	1	1	0.419	1	1	0.419	3	1	0.09
C5315	5	0	0.709	5	0	0.696	9	0	0.4	6	1	0.679	7	2	0.557	13	1	0.6
C6288	111	0	1.981	139	19	0.945	213	1	1.7	0	9	0.66	3	9	0.58	0	9	0.36
C7552	10	0	1.069	9	1	1.021	26	0	0.6	9	2	0.905	6	12	0.808	13	2	0.6
S1488	1	0	0.231	1	0	0.213	3	0	0.1	0	0	0.197	0	0	0.179	6	0	0.05
S38417	20	27	3.026	28	35	2.588	57	20	1.8	66	20	15.937	50	66	2.65	551	20	6.6
S35932	42	77	8.613	86	102	7.401	60	46	6.4	257	46	887.633	259	261	6.728	1745	50	28.7
S38584	45	81	7.216	78	99	6.924	131	36	5.8	N/A	N/A	N/A	173	183	9.946	1653	41	21.1
S15850	57	57	6.783	75	73	6.414	119	34	5.8	N/A	N/A	N/A	185	148	9.185	1462	42	18
avg.	20	16	2.098	30	22	1.848	43	9	1.58	27	7	69.835	46	46	2.130	364	11	5.137

[6] 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. 185–196.

[7] Y. Xu and C. Chu, "GREMA: graph reduction based efficient mask assignment for double patterning technology," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2009, pp. 601–606.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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, pp. 1185–1190.

[13] 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.

[14] J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *ACM/IEEE Design Automation Conference (DAC)*, 2013, pp. 69:1–19:6.

[15] 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.

[16] 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.

[17] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition approaches for double patterning lithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, pp. 939–952, June 2010.

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

[19] X. Tang, X. Yuan, and M. S. Gray, "Practical method for obtaining a feasible integer solution in hierarchical layout optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007, pp. 99–104.

[20] Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com, 2014.

[21] B. Yu, K. Yuan, D. Ding, and D. Pan, "Layout decomposition for triple patterning lithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 3, pp. 433–446, 2015.

TABLE III. COMPARISON ON BENCHMARKS WITH STITCHES (ILP AND IRLS)

Circuit	ILPfor TPL			IRLS+Greedy for TPL			IRLS+ILP for TPL			ILP for QPL			IRLS+Greedy for QPL			IRLS+ILP for QPL		
	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)	sn#	cn#	CPU(s)
C432	4	0	0.099	8	0	0.066	4	0	0.066	0	2	0.1	0	2	0.058	0	2	0.058
C499	0	0	0.272	0	0	0.111	0	0	0.111	3	1	0.144	4	1	0.104	3	1	0.104
C880	3	0	0.132	5	0	0.119	3	0	0.119	0	1	0.126	1	2	0.105	0	1	0.105
C1355	2	0	0.182	2	0	0.165	1	1	0.165	1	0	0.158	1	0	0.133	1	0	0.133
C1908	0	0	0.257	0	0	0.242	0	0	0.242	2	2	0.565	2	2	0.208	2	2	0.208
C2670	2	0	0.381	4	0	0.337	2	0	0.337	2	0	0.329	3	0	0.296	3	0	0.296
C3540	4	1	0.519	4	1	0.483	4	1	0.483	1	1	0.419	1	1	0.419	1	1	0.419
C5315	5	0	0.709	5	0	0.696	4	2	0.696	6	1	0.679	7	2	0.557	6	2	0.557
C6288	111	0	1.981	139	19	0.945	120	11	0.945	0	9	0.66	3	9	0.58	1	9	0.58
C7552	10	0	1.069	9	1	1.021	8	1	1.021	9	2	0.905	6	12	0.808	5	11	0.808
S1488	1	0	0.231	1	0	0.213	1	0	0.213	0	0	0.197	0	0	0.179	0	0	0.179
S38417	20	27	3.026	28	35	2.588	17	36	2.588	66	20	15.937	50	66	2.65	42	53	2.65
S35932	42	77	8.613	86	102	7.401	44	96	7.401	257	46	887.633	259	261	6.728	173	201	6.728
S38584	45	81	7.216	78	99	6.924	39	96	6.924	N/A	N/A	N/A	173	183	9.946	114	137	9.946
S15850	57	57	6.783	75	73	6.414	52	71	6.414	N/A	N/A	N/A	185	148	9.185	107	123	9.185
avg.	20	16	2.098	30	22	1.848	20	21	1.84	27	7	69.835	46	46	2.130	31	36	2.13