Triple Patterning Aware Routing and Its Comparison with Double Patterning Aware Routing in 14nm Technology *

Qiang Ma Hongbo Zhang Martin D. F. Wong
Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
qiangma1@illinois.edu, hzhang27@illinois.edu, mdfwong@illinois.edu

ABSTRACT

As technology continues to scale to 14nm node, Double Patterning Lithography (DPL) is pushed to near its limit. Triple Patterning Lithography (TPL) is a considerable and natural extension along the paradigm of DPL. With an extra mask to accommodate the features, TPL can be used to eliminate the unresolvable conflicts and minimize the number of stitches, which are pervasive in DPL process, and thus smoothen the layout decomposition step. Considering TPL during routing stage explores a larger solution space and can further improve the layout decomposability. In this paper, we propose the first triple patterning aware detailed routing scheme, and compare its performance with the double patterning version in 14nm node. Experimental results show that, using TPL, the conflicts can be resolved much more easily and the stitches can be significantly reduced in contrast to DPL.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithm, Design

Keywords

Triple patterning aware routing, Double patterning aware routing, Maze routing, 14nm technology

1. INTRODUCTION

As the minimum feature size keeps shrinking, and the availability of the next generation lithography methods (EUV, e-beam direct write, etc.) is further delayed, Double Patterning Lithography (DPL) is commonly recognized as a feasible lithography process for 20nm technology nodes [3, 5, 6, 12]. DPL increases pitch size and enhances resolution by decomposing the features on a critical layer into two masks. The conflicting features with spacing between them less than a predefined threshold d_{min} have to be assigned onto different masks. Whenever necessary, a feature can be further sliced to resolve conflicts, which introduce *stitches*. However, in the circumstance of high density layout, it is still possible that some conflicts cannot be resolved with stitches [7]. Moreover, the introduced stitches can lead to yield loss due to overlay error [3]. The problem of layout decomposition with conflicts and stitches minimization for DPL has been extensively studied [7, 13]. Recently, Tang *et al.* have shown in [11] that this problem is polynomial time solvable and provided an optimal algorithm for it.

As technology continues to scale to 14nm node, the scalability of DPL is further challenged. The complexity of layout decomposition grows higher, and stitches become more costly as they are more sensitive to overlay error. As DPL is being pushed to its limit, Triple Patterning Lithography (TPL) is a considerable and natural extension along the paradigm of DPL to alleviate the situation. Industry has already explored the test-chip patterns with triple patterning or even quadruple patterning [1]. Yu *et al.* studied the layout decomposition problem for TPL in [14] and formulated the problem as Integer Linear Programming. With an extra mask to accommodate the features, TPL can be used to (1) eliminate the unresolvable conflicts and reduce the number of stitches while maintaining the same pitch size as DPL (for 14nm node); (2) further increase the pitch size and improve the depth of focus (DOF) (for 10nm node and beyond).

A layout configuration without considering TPL/DPL at design stage can make the layout hard to decompose, and redesigning an indecomposable layout requires high ECO efforts. It is observed that most hard-to-decompose patterns originate from routing wires [4], so considering TPL/DPL during design time, especially detailed routing stage, can significantly benefit the layout decomposition step. Figure 1 gives an example for illustration in the context of TPL. Figure 1(a) shows an indecomposable layout consisting of four nets. All four nets conflict with each other so they need to be assigned to different masks (colors), however, only three masks (colors) are available. If net 2 is implemented with an alternative route, the layout can be decomposed at the expense of introducing a stitch on net 3, as shown in Figure 1(b). In Figure 1(c), net 3 is rerouted, and the layout can be decomposed without stitching. Several previous works studied the problem of optimizing routing for double patterning. Cho *et al.* proposed the first double patterning aware detailed router in [4]. They developed a heuristic which is a modified Dijkstra's shortest path algorithm to take into account the conflicts and stitches when routing one net. Lin and Li developed a double patterning aware gridless router in [8]. They maintain a conflict graph during routing and use it to guide the routes of the incoming nets. Sun *et al.* explored in [10] post-routing layer assignment for DPL optimization. As far as we know, no previous work has addressed triple patterning aware routing. Note that throughout this paper, we interchangeably use the terminologies

*This work was partially supported by the National Science Foundation under grant CCF-1017516 and a grant from the semiconductor Research Corporation (SRC).

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

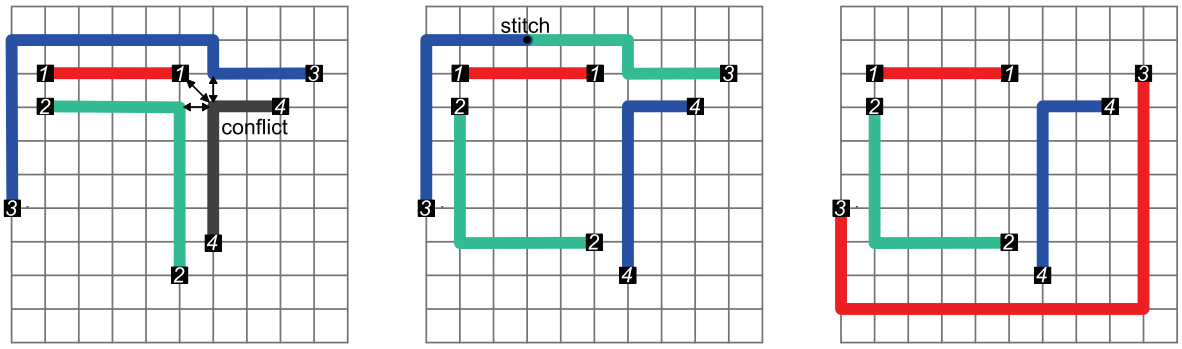


Figure 1: (a) The routing solution cannot be successfully decomposed due to the conflict; (b) The routing solution can be decomposed at the expense of introducing a stitch; (c) The routing solution can be decomposed without stitching.

triple patterning and triple patterning lithography (TPL), as well as double patterning and double patterning lithography (DPL).

In this paper, we study triple patterning aware routing and compare it with double patterning aware routing in 14nm technology node. We first propose a graph model that correctly models the cost of conflicts and stitches in TPL. By replacing each vertex in the routing grid with the graph model and performing shortest path algorithm on the expanded graph, the optimal path with mask/color assignment for one net can be computed, in presence of previously routed and colored nets. Our proposed graph model is a unified model that can be extended to handle multiple patterning and can also be tailored for double patterning. We then develop a negotiated congestion based scheme to resolve conflicts. The regions with conflicts are penalized and the nets are iteratively rerouted and re-colored. Experimental results show that this scheme is very effective and all conflicts can be resolved within a small number of iterations in our test cases. An effective heuristic is also developed to ensure approximately balanced utilization of the three masks. To the best of our knowledge, this is the first triple patterning aware detailed router developed in literature. We also implement a double patterning aware detailed router by adapting our graph model for double patterning, and compare it with the triple patterning version in 14nm node. Experimental results show that using TPL the conflicts can be resolved much more easily and the stitches can be significantly reduced in contrast to DPL.

The rest of this paper is organized as follows: Section 2 states the triple patterning aware routing problem; Section 3 introduces our proposed graph model and describes how to route a single net on the expanded routing graph; Section 4 presents our negotiated congestion based scheme to resolve conflicts, as well as the heuristic to balance the routing on the three masks; Section 5 reports the experimental results, and Section VI concludes this paper.

2. PROBLEM FORMULATION

In TPL, we have three masks available to accommodate all the features within one layer. For the ease of discussion, we use three colors, namely, RED, GREEN and BLUE, to represent the three masks. In TPL aware detailed routing, we perform simultaneous routing and color assignment. If the spacing between two pieces of wires assigned with the same color is smaller than a predefined minimum spacing requirement d_{min}, a conflict is caused and the two pieces of wires will contribute to the total conflicting wire length. If a piece of wire changes its color assignment at some point, as net 3 in Figure 1(b), a stitch is introduced. We want to minimize the total conflicting wire length as well as the number of stitches. The triple patterning aware detailed routing problem is stated as follows.

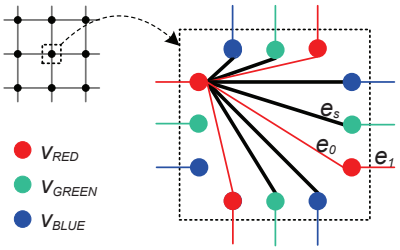


Figure 2: Graph model for a non-pin vertex in the routing grid.

DEFINITION 1. **Triple Patterning Aware Detailed Routing** - *Given a netlist, a routing grid, a minimum spacing requirement d_{min} and three colors (RED, GREEN and BLUE), detailed routing with simultaneous color assignment is performed such that the total conflicting wire length and number of stitches are minimized.*

An immediate subproblem is how to perform maze routing for a single net on the routing grid in the presence of a set of previously routed nets. When we route a net, it is desired to compute a path p with color assignment that produces minimum conflicting wire length and minimum number of stitches. Of course, the wire length of the net, as a conventional metric, also needs to be minimized. Therefore, the weighted sum $l_w^p + \alpha l_{con}^p + \beta n_s^p$ is a good cost metric to minimize when we route a single net, where l_w^p, l_{con}^p and n_s^p denote the wire length, the conflicting wire length and the number of stitches produced by the path p computed, respectively, and α and β are user defined parameters that specify the relative importance between them. We define the triple patterning aware maze routing problem below.

DEFINITION 2. **Triple Patterning Aware Maze Routing** - *Given a set of previously routed nets together with their color assignment on a routing grid, as well as two pins of a net, the objective is to compute a path p with color assignment between the two pins such that the weighted sum $l_w^p + \alpha l_{con}^p + \beta n_s^p$ is minimized.*

3. ROUTING A SINGLE NET

In this section, we propose a graph model that correctly captures the cost of conflicts and stitches, and show that the triple patterning aware maze routing problem can be optimally solved by performing shortest path algorithm on an expanded routing graph constructed using the graph model. We then discuss a practical extension of the proposed graph model to forbid stitch at corner.

3.1 Triple Patterning Aware Maze Routing

Suppose we are given a routing grid G, which can be viewed as a routing graph if we regard every intersection of four line segments

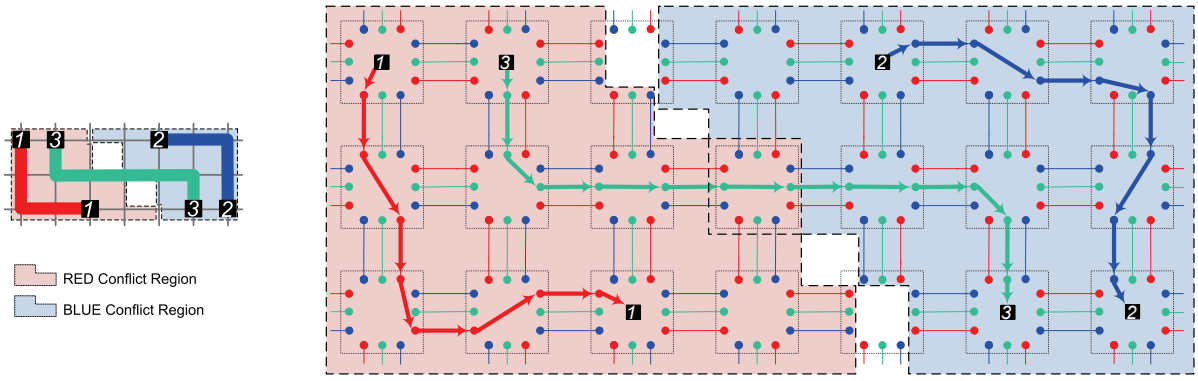


Figure 4: (a) Routes on the original routing grid; (b) Routes on the expanded routing graph.

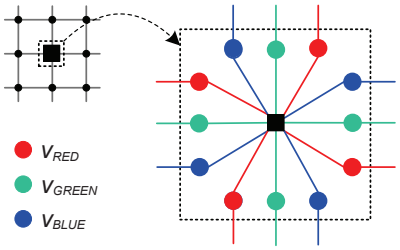


Figure 3: Graph model for a pin vertex in the routing grid.

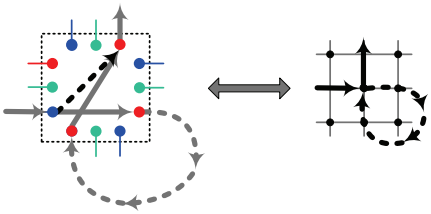


Figure 5: A path in the expanded routing G' may correspond to a walk with loop in the routing grid G.

as a vertex and the short segments between vertices as edges. In the triple patterning aware maze routing problem, the cost of wire length and conflicting wire length can be easily captured by assigning cost to the edges of G. However, the cost of stitches cannot be directly captured. In order to capture the cost of stitches as well, we split each vertex v of G into 12 vertices and construct a graph model on them, as shown in Figure 2. The detailed construction is described as follows:

- The 12 vertices fall into three categories, namely, v_{RED}, v_{GREEN} and v_{BLUE}, which correspond to the three colors RED, GREEN and BLUE, respectively. On each of the four boundaries of the graph model, there are one v_{RED}, one v_{GREEN} and one v_{BLUE}, as shown in Figure 2. On the two adjacent boundaries of two neighboring graph models, the two vertices of the same color are connected.
- Within the graph model, two vertices are connected by an edge if and only if they are not lying on the same boundary. Note that in Figure 2, only the edges adjacent to the vertex v_{RED} on the left boundary are displayed. This graph model works like a switch box. A route can come into the graph model at any boundary and go out at any other boundary. The color of the route can be changed within the model.
- All the edges can be categorized into three types, namely, e_0, e_s and e_1. A type e_0 edge with cost 0 connects two vertices of the same color within the graph model. A type e_s edge with cost β (the cost of a stitch), as indicated by a thick edge in Figure 2, connects two vertices of different colors within the graph model. It corresponds to a stitch. A type e_1 edge connects two same color vertices of two neighboring graph models. It corresponds to an edge in the original routing grid. The base cost of a type e_1 edge is 1, which indicates the cost of the unit wire length. However, when routing on a type e_1 edge causes a coloring conflict, the cost of this edge will be updated to $(1+\alpha)$, where α is the cost of the unit conflicting wire length. Note that the type e_1 edges are shared by the neighboring graph models.

Note that if a vertex v in G is a pin of a net, we will construct the graph model for v as shown in Figure 3 instead, where v is still split into the 12 vertices, and another vertex representing the pin will be added and connected to the 12 vertices.

We now obtain an expanded routing graph G' from the original routing grid G through the construction described above. When we route one net, we simply apply Dijistra's shortest path algorithm on G'. Figure 4 gives an example for illustration. Figure 4(a) displays the routes of three nets on the original routing grid, while Figure 4(b) demonstrates the corresponding routes on the expanded routing graph. Suppose, without loss of generality, the net ordering during routing is net 1, net 2, then net 3. Net 1 and net 2 are routed and assigned to RED and BLUE, respectively. The two shaded regions are respectively RED conflict region and BLUE conflict region. The type e_1 edges connecting two v_{RED} (v_{BLUE}) vertices within the RED (BLUE) conflict region will have their cost be updated as $1+\alpha$, indicating that using these edges for routing will cause conflicts. Therefore, when net 3 is routed, the shortest path algorithm will find the path colored GREEN as shown in Figure 4.

To show the optimality of our graph expansion based approach, we first show that the shortest path p' between two pins on G' must correspond to a path p between the two pins on G. Suppose the shortest path p' on G' does not correspond to a path p on G, and it corresponds to a walk with loop on G instead. It follows that p' enters a graph model at least twice on G'. Let us assume that, w.l.o.g., p' enters a graph model twice as shown in Figure 5. However, the part of path p' that produces a loop can always be shortcut by an edge inside the graph model. This indicates that p' is not a shortest path, which is contradictory to the assumption. Note that the shortest path p' on G' also determines the color assignment of its corresponding path p on G. Then, from our graph model construction and cost setting, it is easy to see that the $l_w^p + \alpha l_{con}^p + \beta n_s^p$ value of a path p with color assignment on G is the same as the cost of its corresponding path p' on G'. Thus, the shortest path p'

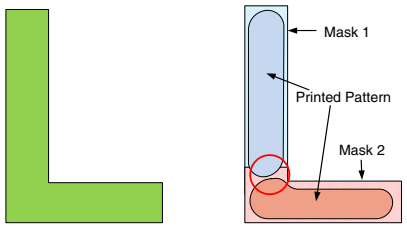


Figure 6: (a) Pattern to be printed; (b) Stitching at corner results in significant printability degradation due to overlay error and line-end effect.

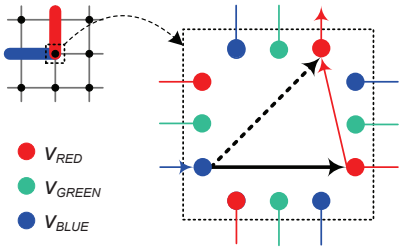


Figure 7: Corner stitches cannot be prevented by simply removing the edges that directly generate corner stitches.

between two pins on G' corresponds to a colored path p of smallest $l_w^p + \alpha l_{con}^p + \beta n_s^p$ value between the two pins on G. Based on the above analysis, we can conclude that the triple patterning aware maze routing problem on a routing grid G can be optimally solved by computing the shortest path between the two pins on the expanded routing graph G'.

3.2 Forbidding Stitch at Corner

A routing path can change its color assignment at some point to avoid conflicts, which introduces a stitch. Stitches are sensitive to overlay errors and thus lead to printability degradation. In particular, when a stitch occurs at a turning point, or corner, of a route, the situation is even worse and the degradation of printability can be much more significant [2], as illustrated in Figure 6. Therefore, stitch at corner is highly undesirable. In this subsection, we show that our graph model can be extended to disallow stitch at corner.

Intuitively, stitch at corner can be prohibited by removing the type e_s edges that introduce corner stitches in the graph model. For example, in Figure 7, the edge connecting v_{BLUE} on the left boundary and v_{RED} on the top boundary (shown in dashed line) will be removed. However, this does not work as v_{BLUE} on the left boundary can still reach v_{RED} on the top boundary by taking a detour as shown by the solid edges in Figure 7. To address this issue, we further split each vertex v in the graph model into two vertices v^{in} and v^{out}. v_{RED} (v_{GREEN}, v_{BLUE}) is split into v_{RED}^{in} (v_{GREEN}^{in},v_{BLUE}^{in}) and v_{RED}^{out} (v_{GREEN}^{out}, v_{BLUE}^{out}). We also make the edges directed. The edges are coming into the graph model through v_{in}, and going out of the graph model through v^{out}. Within the graph model, an edge will be directed from v^{in} to v^{out}. The new graph model that disallows stitch at corner is shown in Figure 8. Note that only the edges adjacent to the vertices on the left boundary are displayed. A minor flaw of using this new graph model is that a loop similar to the one shown in Figure 5 may be produced by applying the shortest path algorithm. A path may take a detour outside the graph model to reach v_{RED}^{out} on the top boundary from v_{BLUE}^{in} on the left boundary, but this time the detour can no longer be shortcut as there is no connection between the two vertices within the new graph model. However, according to our experiment, this situation rarely happens. When it happens, we simply block the place where the detour occurs and run the shortest path algorithm again. This technique turns out to be very effective.

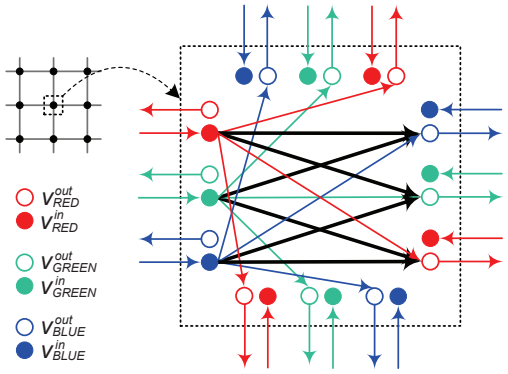


Figure 8: Graph model for TPL that disallows stitch at corner.

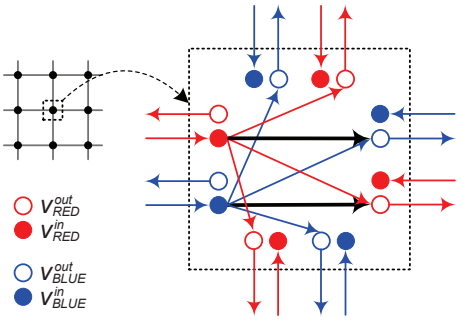


Figure 9: Graph model for DPL that disallows stitch at corner.

Our proposed graph model is a unified graph model, which can be extended for multiple patterning lithography, and can also be tailored for double patterning lithography. The graph model for DPL is shown in Figure 9, which uses two colors *RED* and *BLUE*.

4. OVERALL ROUTING SCHEME

In this section we present our overall routing scheme for the triple patterning aware detailed routing problem. We adopt a negotiated congestion based scheme to resolve the coloring conflicts over iterations of rip-up and reroute/re-color. Since the nets are routed and colored one by one sequentially, the solution obtained within one single pass is not good enough, and coloring conflicts may exist. The negotiated congestion based scheme is able to dynamically refine the routing and coloring solution over iterations. This scheme can significantly reduce the adverse effect of improper net ordering. We also propose a way to balance the features over the masks, which is beneficial for manufacturing. Our overall routing scheme also works for double patterning aware detailed routing.

4.1 Negotiated Congestion based Scheme to resolve Coloring Conflicts

The negotiated congestion based routing scheme [9] has been widely used in FPGA routing and global routing to resolve routing congestions. In this routing scheme, routability is achieved by forcing all the nets to negotiate for a resource and thereby determine which net needs the resource most. Some nets may use shared resources that are in high demand if all alternative routes utilize resources in even higher demand; other nets will tend to spread out and use resources in lower demand. All the nets are iteratively rerouted until no more resources are shared.

We adapt this negotiated congestion based scheme to resolve coloring conflicts in our triple patterning aware detailed routing problem. We let the nets negotiate for the color assignment by adding a history cost to the type e_1 edges in the expanded routing graph G'. The cost of a type e_1 edge is computed by the following formula:

$$cost(e_1) = 1 + \alpha \times (isConflict? \; h_c : 0),$$

where $isConflict$ is a boolean variable indicating if this edge lies in the conflict region produced by the previously routed nets, and h_c denotes the history cost. This means that if routing on this type e_1 edge causes a conflict, the cost of this edge will be set to $1+\alpha \times h_c$; otherwise, the cost will be set to 1. h_c is initialized as 1.

This scheme works as follows. We start with a global routing solution without color assignment. In the initial iteration, each net is rerouted on the expanded routing graph G'. When the initial iteration terminates, the routes on G' provide a color assignment of all the nets. Note that during routing one net, the access to the graph models occupied by other nets is denied, so that no crossing will be generated and routability is guaranteed. If coloring conflicts exist, iterations of rip-up and reroute will be performed. When a net i is rerouted, we first remove its current route, as well as the conflict region(s) it produces. If a type e_1 edge was lying in net i's conflict region of the same color, its flag $isConflict$ will be updated as *false* since the conflict region is now gone. The shortest path algorithm is then performed to compute a new path for net i, and the conflict region(s) it produces will be updated. If a type e_1 edge falls into a conflict region representing the same color, its flag $isConflict$ will be set as *true*. In addition, if the path of this net causes coloring conflicts with previously routed nets, the type e_1 edges that are responsible for the conflicts will have their history cost h_c incremented by 1. In this way, the regions with coloring conflicts grow more expensive over iterations, and those nets with more options will tend to choose alternative routes or colors in subsequent iterations, so that the conflicts can potentially be resolved. In our implementation, this procedure will terminate when either no more conflicts exist or enough iterations of rerouting have been performed. Our experimental results show that this scheme is very effective in resolving conflicts.

The example in Figure 10 demonstrates how this negotiated congestion based scheme works. There are 6 nets, where net 1-5 are previously routed with color assignment, and net 6 is the current net to be routed, as shown in Figure 10(a). It is easy to see that net 6 will cause conflict with other nets no matter what color it is assigned. Net 6 is colored in RED by the shortest path algorithm, which causes conflicts with net 4, as shown in Figure 10(b). As a result, the type e_1 edges that are responsible for the conflicts have their history cost h_c incremented. Therefore, in the next iteration, net 4 is rerouted and colored in BLUE, so that the conflicts are effectively resolved, as shown in Figure 10(c).

4.2 Balancing Features on Three Masks

TPL provides three masks to accommodate the features. Balancing the features on the three masks ensures that each mask is fully utilized, so that none of the masks is unnecessarily dense. This helps the printability enhancement during manufacturing. We develop a heuristic that can effectively control the balancing on the fly. During routing, we maintain three variables l_{RED}, l_{GREEN} and l_{BLUE}, which respectively keep track of the wire length colored in RED, GREEN and BLUE. The cost of the type e_1 edges will be adjusted according to the current distribution of the total wire length on the three masks. For example, when relatively more wires are assigned to GREEN, the cost of the type e_1 edges of *GREEN* color should be increased, so that the router tends to favor the routes using other colors. In our implementation, we let l_{RED} be the reference. The cost of the type e_1 edges of GREEN color is scaled by $\frac{l_{GREEN}}{l_{RED}}$, and the cost of those of BLUE color is scaled by $\frac{l_{BLUE}}{l_{RED}}$. This technique is shown to be effective through the experiments.

The pseudocode of our overall routing scheme is listed below.

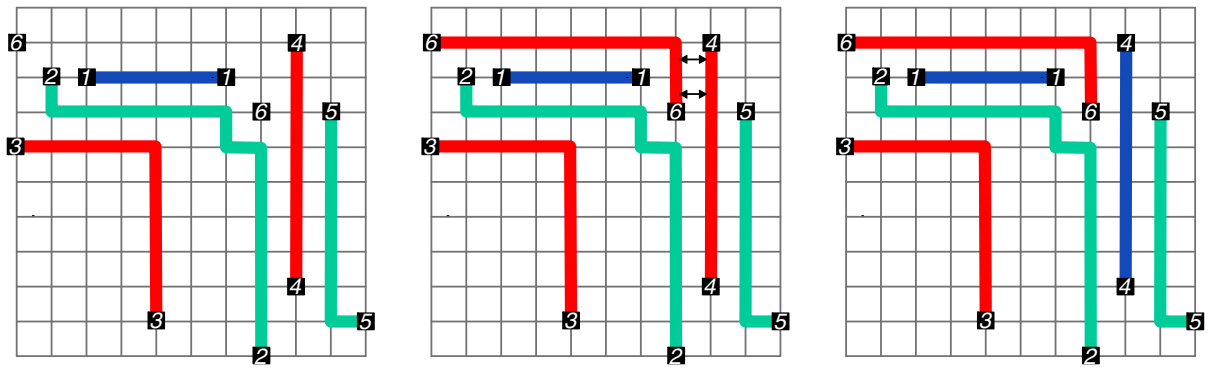
```
ALGORITHM TPL AWARE DETAILED ROUTING(G,N):
  Construct the expanded routing graph G' from G;
  Set h_c as 1 for all the type e_1 edges;
  for each net i in N do      //initial iteration
      Reroute net i using shortest path algorithm;
      Update flag isConflict for the affected e_1 edges;
      Update l_RED, l_GREEN and l_BLUE;
  while ∃ conflicts do
      for each net i in N do
          Remove the route of net i;
          Update flag isConflict for the affected e_1 edges;
          Update l_RED, l_GREEN and l_BLUE;
          Reroute net i using shortest path algorithm;
          Update flag isConflict for the affected e_1 edges;
          Increment h_c for the conflicting e_1 edges;
          Update l_RED, l_GREEN and l_BLUE;
```

5. EXPERIMENTAL RESULTS

We implement a TPL aware detailed router as well as a DPL version, and compare their performance in 14nm node on two sets of benchmarks. The graph model that disallows stitch at corner is adopted for both triple patterning and double patterning. Our program is implemented in C++ and all the experiments are performed on a Linux machine with 2.0GHz CPU and 2GB RAM.

We first compare the two routers on a set of benchmarks derived from industrial data. The result of comparison is shown in Table 1. The column "Init Con. WL" shows the conflicting wire length after the initial iteration. The column "#iter" shows the total number of rerouting iterations performed. The column "Con. WL" reports the final conflicting wire length. We can see that our negotiated congestion based routing scheme is very effective in resolving conflicts. All the conflicts can be resolved within a few iterations for both DPL and TPL. The column "#Stitch" and the column "#Via" report the number of stitches and the number of vias in the final solution, respectively. We can see that TPL, with an extra mask to accommodate the features, can remove almost all of the stitches (99.9% on average) that are necessary for DPL. TPL can also reduce the number of vias by 37% on average compared with DPL. The column "Wirelength" reports the total wire length together with the ratio of the wire length on each mask. The results show that our approach can balance the utilization of the masks very well. We then compare the two routers in terms of ability to resolve conflicts on a set of denser benchmarks. The result is displayed in Table 2. The maximum number of rerouting iterations is set to be 30. We can see from the result that the double patterning router generates a lot more conflicts from the beginning and cannot resolve all of them after 30 iterations, while the triple patterning router can easily bring the conflicting wire length down to zero within a small number of iterations.

By comparing DPL and TPL under the same printing condition, we would like to answer the multiple choice question about which patterning technique to use in 14nm technology node. During the experiment and with help of our proposed algorithm, we have set up the comparison between DPL and TPL fairly enough: (1) the same test benches; (2) the same technology node setup; (3) the routing-coloring co-optimized designs for both DPL and TPL. Note that since the stitches can be almost completely avoided (in Table 1),



(a) (b) (c)

Figure 10: An example of using the negotiated congestion based scheme to handle conflicts. (a) Net 1-5 are routed and colored, and net 6 is not routed yet; (b) Net 6 is routed in color red, causing conflicts with net 4; (c) In the next iteration, net 4 is rerouted in color blue due to the penalty exerted on the red edges.

Table 1: Comparison of TPL and DPL in 14nm technology

Test Cases	#Net	Size (um^2)	Init Con. WL(um)		#Iter		Con. WL		#Stitch		#Via		Wirelength (um)		Runtime (s)	
			DP	TP	DP	TP	DP	TP	DP	TP	DP	TP	DP (R:B)	TP (R:G:B)	DP	TP
test1	1K	31.36	7.39	0.28	16	3	0	0	49	0	648	476	354(1:1.03)	354(1:1.005:0.99)	57	17
test2	2K	70.56	9.07	0.39	9	3	0	0	96	0	1076	750	720(1:1.04)	723(1:1.016:1.0007)	143	68
test3	4K	165.9	8.23	0.28	4	3	0	0	185	0	1622	878	1444(1:1.02)	1461(1:1.02:1.04)	273	301
test4	6K	245.9	12.15	0.56	16	2	0	0	247	0	2252	1496	2130(1:1.012)	2162(1:0.97:1.003)	2431	438
test5	8K	321.1	19.08	0.67	7	3	0	0	365	0	3426	2036	2873(1:0.999)	2911(1:1.01:1.02)	1790	1194
test6	10K	384.2	30.41	2.13	15	4	0	0	419	1	4452	2782	3593(1:1.016)	3625(1:0.998:1.01)	5673	2210
Avg			14.56	0.716	11.2	3	0	0	227	0.17	2246	1403	1852	1873	1728	705
Diff			1	-95%	1	-73%	0	0	1	-99.9%	1	-37%	1	+1.13%	1	-59%

Table 2: Comparison of TPL and DPL in resolving conflicts

Test Cases	#Net	Size (um^2)	Init Con. WL(um)		#Iter		Con. WL(um)	
			DP	TP	DP	TP	DP	TP
test1d	1.2K	31.36	18.47	0.82	30	3	7.68	0
test2d	2.5K	70.56	24.50	1.15	30	5	8.35	0
test3d	5K	165.9	28.40	0.96	30	4	6.24	0
test4d	7.5K	245.9	48.92	1.32	30	4	12.46	0
test5d	10K	321.1	71.26	1.89	30	5	20.79	0
test6d	12K	384.2	87.41	4.26	30	8	37.61	0
Avg			46.49	1.73	30	4.83	15.52	0
Diff			1	-96%	1	-84%	1	-100%

the overlay in TPL will not be as harmful as in DPL where stitches are commonly seen. From the comparison (Table 1 and Table 2), we can see that choosing between DPL and TPL in 14nm node is intrinsically a trade-off between mask/process cost and printability. In the circumstances where stitches are highly likely to cause yield loss and the extra mask cost from TPL is still affordable, with the help of our proposed algorithm, the solution of TPL with simultaneous routing and coloring would be a wise choice in 14nm technology node.

6. CONCLUDING REMARKS

In this paper, the problem of TPL aware routing is studied. A unified graph model is proposed to solve the maze routing problem in the context of multiple patterning lithography. An overall routing scheme is developed to resolve the coloring conflicts and balance the utilization of the masks. Triple patterning aware routing is compared with double patterning aware routing in 14nm node. The results show that TPL can resolve the conflicts more easily and significantly reduce the number of stitches compared with DPL.

7. REFERENCES

[1] International Technology Roadmap for Semiconductors 2011, http://www.itrs.net.

[2] D. Abercrombie, P. Lacour, O. El-Sewefy, A. Volkov, E. Levine, K. Arb, C. Reid, Q. Li, and P. Ghosh. Double patterning from design enablement to verification. In *Proc. SPIE vol. 8166*, 2011.

[3] 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. In *Proc. SPIE vol. 6521*, 2007.

[4] M. Cho, Y. Ban, and D. Z. Pan. Double patterning technology friendly detailed routing. In *Proc. ICCAD*, pages 506–511, 2008.

[5] J. Huckabay, W. Staud, R. Naber, A. van Oosten, P. Nikolski, S. Hsu, R. J. Socha, M. V. Dusa, and D. Flagello. Process results using automatic pitch decomposition and double patterning technology (DPT) at $k_{1eff} < 0.20$. In *Proc. SPIE vol. 6349*, 2006.

[6] Y. Inazuki, N. Toyama, T. Nagai, T. Sutou, Y. Morikawa, H. Mohri, N. Hayashi, M. Drapeau, K. Lucas, and C. Cork. Decomposition difficulty analysis for double patterning and the impact on photomask manufacturability. In *Proc. of SPIE, vol. 6925*, 2008.

[7] A. B. Kahng, C. H. Park, X. Xu, and H. Yao. Layout decomposition for double patterning lithography. In *Proc. ICCAD*, pages 465–472, 2008.

[8] Y.-H. Lin, and Y.-L. Li. Double patterning lithography aware gridless detailed routing with innovative conflict graph. In *Proc. DAC*, pages 398–403, 2010.

[9] L. McMurchie, and C. Ebeling. Pathfinder: a negotiation-based performance-driven router for fpgas. In *Proc. FPGA*, pages 111–117, 1995.

[10] J. Sun, Y. Lu, H. Zhou, and X. Zeng. Post-routing layer assignment for double patterning. In *Proc. ASPDAC*, pages 793–798, 2011.

[11] X. Tang and M. Cho. Optimal layout decomposition for double patterning technology. In *Proc. ICCAD*, 2011.

[12] 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*, 2008.

[13] Y. Xu, and C. Chu. GREMA: Graph reduction based efficient mask assignment for double patterning technology. In *Proc. ICCAD*, pages 601–606, 2009.

[14] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan. Layout decomposition for triple patterning lithography. In *Proc. ICCAD*, 2011.