Double Patterning Technology Friendly Detailed Routing

Minsik Cho, Yongchan Ban, and David Z. Pan
Dept. of ECE, The University of Texas at Austin, Austin, TX 78712
{thyeros, ycban, dpan}@cerc.utexas.edu

***Abstract*— Double patterning technology (DPT) is a most likely lithography solution for 32/22nm technology nodes as of 2008 due to the delay of Extreme Ultra Violet lithography. However, it should hurdle two challenges before being introduced to mass production, layout decomposition and overlay error. In this paper, we present the first detailed routing algorithm for DPT to improve layout decomposability and robustness against overlay error, by minimizing indecomposable wirelength and the number of stitches. Experimental results show that the proposed approach improves the quality of layout significantly in terms of decomposability and the number of stitches with 3.6x speedup, compared with a current industrial DPT design flow.**

I. Introduction

To bridge the gap between current immersion lithography and again-delayed EUV lithography, double patterning technology (DPT) receives large attention from industry and is regarded as a technically and practically viable alternative to achieve high resolution for 32/22nm nodes [1], [7], [8], [11], [13], [14], [17]. The key idea of DPT is to decompose a single layout into two masks in order to increase pitch size and improve depth of focus (DOF) [9], [15]. Fig. 1 illustrates the concept of DPT. The increased pitch size brings several advantages which enables higher resolution and better printability [7]: (a) the performance of Sub-Resolution Assist Features (SRAF) and Optical Proximity Correction (OPC) algorithms will be enhanced; (b) DPT is generic to be applied for poly, metal, active, and even via layers; (c) current manufacturing infrastructures (e.g., stepper) and materials (e.g., photo-resist) can be reused without expensive modification. These advantages all make DPT as the most prominent manufacturing solution for 32/22nm nodes.

However, the deployment of DPT needs to tackle two major challenges, layout decomposition and overlay error [1], [6], [9], [15]. As shown in Fig. 1, a layout has to be decomposed (or colored differently). Unfortunately, such decomposition is not always feasible, especially for complex 2D patterns common in metal layers [1], [12], [13] owing to new spacing constraints from DPT. For indecomposable cases, a simple solution is to modify the layout, which will be highly expensive. Another solution is to split one polygon into two in order to resolve decomposition conflicts, which will introduce a stitch as shown in Fig. 2 (a). However, a stitch is highly sensitive to overlay error, potentially causing pinching or bridging issues as shown in Fig. 2 (b) [6], [12]. Therefore, it is important to make a layout more decomposable with fewer stitches.

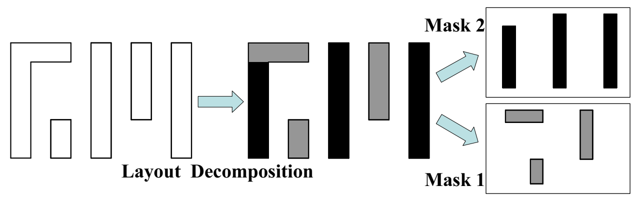
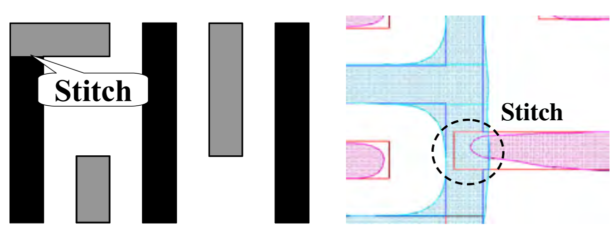


Fig. 1. In DPT, one single layer can be decomposed into two masks to effectively increase pitch size [1].

There are only a few previous works on layout decomposition mainly from a mask synthesis perspective using a commercial simulator [8], design guidelines [17], and pattern matching [14]. However, all these works mainly focus on post-design optimization, which may be too late for successful decomposition. Also, none of them minimize the number of stitches systematically. Therefore, it is in great demand to take DPT into account during design time, especially detailed routing in order to generate a highly decomposable layout with a small number of stitches due to the following reasons: (a) most of hard-to-decompose patterns are from complex 2D routing wires; (b) it is the last major design optimization step with a comprehensive view on DPT; (c) there is considerable design flexibility to find reasonable tradeoff between DPT and conventional design objectives (e.g., timing, via, wirelength).

In this paper, we propose the first DPT-friendly detailed routing algorithm. The key idea behind our algorithm is to perform detailed routing and layout decomposition (or



(a) A polygon can be splitted to resolve a decomposition or coloring conflict at a cost of stitch.

(b) Stitch may result in significant printability degradation due to overlay error and line-end effect.

Fig. 2. The concept of a stitch is elaborated by an example in (a), and its susceptibility to overlay error is demonstrated in (b).

This work is supported in part by NSF, SRC, IBM Faculty Award, Sun, Fujitsu, Qualcomm and equipment donations from Intel.

coloring) *simultaneously* in a correct-by-construction manner to accomplish high layout decomposability and reduce the number of overlay-error-prune stitches. Therefore, our DPT-friendly detailed routing directly outputs a decomposed layout without an extra time-consuming decomposition step.

The rest of the paper is organized as follows. Section II provides preliminaries on DPT and its challenges. Section III motivates DPT consideration during design time. Then, we propose our DPT-friendly detailed routing algorithm in Section IV. Experimental results are discussed in Section V, followed by conclusion in Section VI.

II. Preliminaries

A. Double Patterning Technology (DPT)

The difficulty of a process technology can be described by k_1 in Rayleigh Formulae [4], $k_1 = HP\frac{NA}{\lambda}$ where λ is wavelength of the light (currently $193nm$ for ArF lithography), NA is numerical aperture, and HP is minimum printable half-pitch. In order to print a feature in the $32nm$ node with the current single exposure infrastructure, we should increase k_1 above at least 0.25, which can be accomplished by various ways including the 3rd generation immersion fluid (Refraction Index (RI) > 1.8), larger lens, or Extreme Ultra Violet (EUV) light source (λ=13.5nm). However, in light of the physical and practical limitations in the above ways, the only feasible solution is to increase pitch size without changing minimum feature size by double patterning technology (DPT). By decomposing a layout into two masks as shown in Fig. 1, we can effectively double HP, theoretically enabling 65nm technology/infrastructure to print 32nm designs.

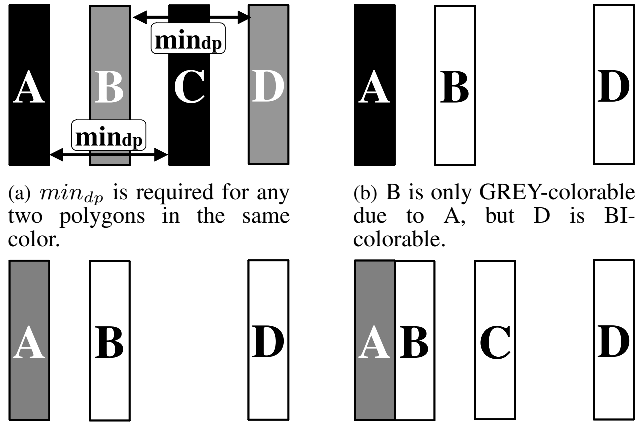
As expected, however, DPT process is highly complex, as one layer needs to be patterned by two exposures and two etching with two masks. There are several DPT lithography processes; litho1-etch1-litho2-etch2 (LELE) [6], spacer type DPT [2], and litho oriented DPT [16]. Although there are differences in different DPT processes, all are highly complex and involve multiple common challenges in both design and manufacturing sides such as layout decomposition and stitch minimization, which will be discussed in Section II-B.

B. Challenges in DPT

The two most important issues to deal with DPT are layout decomposition and overlay-error-prune stitches [12].

- **Layout Decomposition** in DPT is to decompose (color) the original design polygons into two groups or colors (BLACK or GRAY) to decide which polygon will be placed on which mask under the minimum double patterning spacing constraint.
- **Stitch Minimization** is another critical issue in DPT due to the overlay error which is caused by the mismatch between the first patterning and the second patterning. Unfortunately, a stitch is known to be highly sensitive to the overlay error, causing bridging or pinching. Fig. 2 (b) shows an example of a notching error due to a stitch.

Due to such criticality and importance, layout decomposition and stitch minimization have been considered during mask



(a) min_{dp} is required for any two polygons in the same color.

(b) B is only GREY-colorable due to A, but D is BI-colorable.

(c) B is only BLACK-colorable due to A, but D is BI-colorable.

(d) B is BI-colorable, and the color of C depends on that of B.

Fig. 3. This example shows the key concepts in DPT.

synthesis/manufacturing [8], [14], [17], but cannot be effectively addressed due to their high design dependency.

C. Definitions

We explain the key definitions in DPT with Fig. 3: min_{dp}, BLACK-colorable, GREY-colorable, and BI-colorable. During layout decomposition, as mentioned earlier, polygons will be divided into two masks or two colors (GREY or BLACK). And, two polygons on the same mask (thus in the same color) should maintain minimum double patterning spacing or min_{dp}. For example, since A and C are in BLACK, min_{dp} is required between two as shown in Fig. 3 (a). Such min_{dp} sometimes enforces a specific color for some polygon, if there is an already colored polygon nearby. Consider Fig. 3 (b). Since A is already in BLACK, B should be colored as GREY not to violate the min_{dp} constraint, thus B is only GREY-colorable. Similarly in Fig. 3 (c), B is only BLACK-colorable. In both Fig. 3 (b) and (c), D can be colored in either way as it has enough spacing from B, so called BI-colorable.

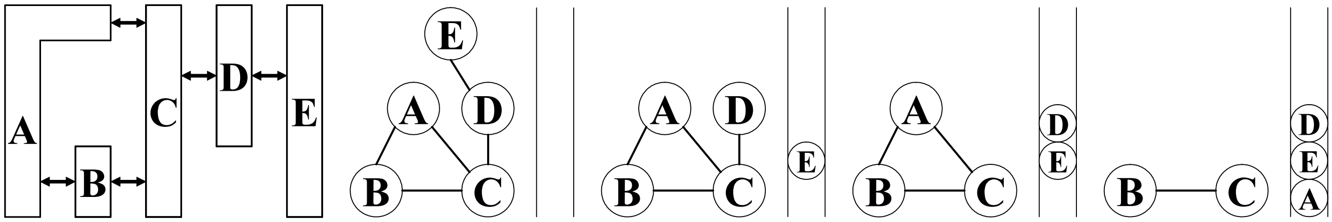
An interesting case is in Fig. 3 (d) where A and B are abutted. For this case, B is BI-colorable, because coloring B as GREY does not violate the min_{dp} constraint (as, A and B can be treated as one bigger polygon) and coloring B as BLACK is still fine at a cost of a stitch. The color of C depends on how B will be colored. If B is in GREY eventually, then C will be BLACK-colorable (otherwise GREY-colorable).

III. Motivations

In this section, we illustrate the complexity of layout decomposition in Section III-A. Then, we further motivate why detailed routing can make significant impact on layout decomposition as well as the number of stitches in Section III-B.

A. Complexity of Layout Decomposition

At the first glance, layout decomposition for DPT seems identical to the phase-assignment problem [3], as both can



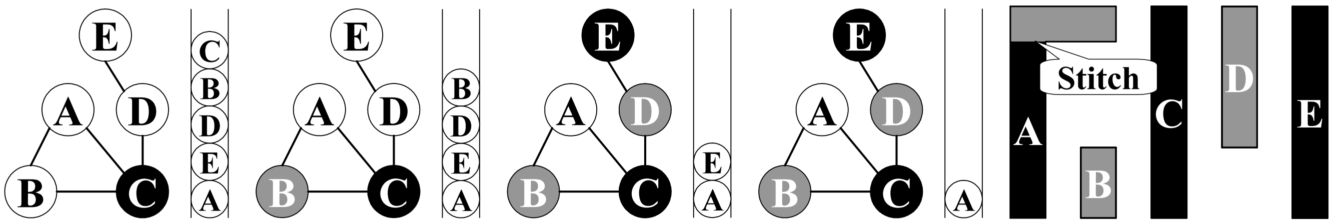
(a) An example for layout decomposition for DPT is shown with five conflicts among polygons.

(b) A conflict graph for 2-coloring can be built from (a) with a double-ended queue.

(c) The node E with degree<2 is detached and pushed into the top of the queue.

(d) The node D with degree<2 is detached and pushed into the top of the queue.

(e) The node A with the largest degree is detached and put into the bottom of the queue.



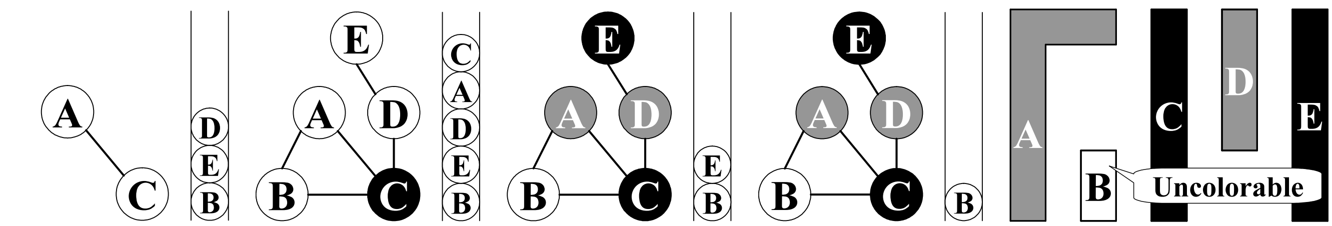
(f) The node C on the top of the queue is colored as BLACK and popped out.

(g) The node B on the top of the queue is colored as GRAY and popped out.

(h) The node E on the top of the queue is colored as BLACK and popped out.

(i) The node A on the top of the queue cannot be colored due to the conflicts with B and C.

(j) However, the node A can be colored with BLACK and GRAY with a stitch, resulting in successful decomposition for DPT.



(k) The node B with the largest degree is detached and put into the bottom of the queue.

(l) The node C on the top of the queue is colored as BLACK and popped out.

(m) The node E on the top of the queue is colored as BLACK and popped out.

(n) The node B on the top of the queue cannot be colored due to conflicts.

(o) The node B cannot be colored due to A in GRAY and C in BLACK, resulting in decomposition failure.

Fig. 4. This example describes a layout decomposition approach based on 2-coloring of a conflict graph in (a)–(j), but further shows that the same layout cannot be decomposed by the same 2-coloring approach as shown in (k)–(o), depending on how to spill nodes. Therefore, layout decomposition for DPT is not equivalent to but much more complex than 2-coloring, while phase-assignment is equivalent to 2-coloring [10].

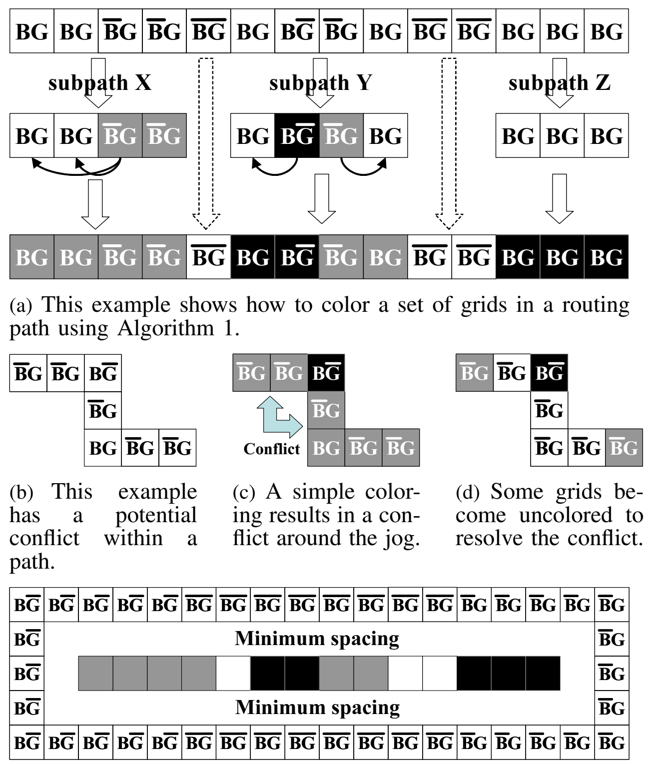
be formulated as a 2-coloring problem. However, there are two key differences. Phase-assignment is for the space between polygons, but layout decomposition for DPT is for the polygons. More importantly, resolving a conflict in phase-assignment needs to involve layout modification (e.g., increasing spacing) [3], but not necessarily in DPT, as a polygon can be severed into multiple polygons without altering a layout.

Consider a layout in Fig. 4 (a) where five disconnected polygons are shown along with five conflicts in double-headed arrows. We can formulate layout decomposition of Fig. 4 (a) as a 2-coloring problem by building a corresponding conflict graph and performing 2-coloring (BLACK or GRAY) based on Chatin's algorithm [5]. In Fig. 4 (b), a conflict graph for the layout in (a) is constructed and a double-ended queue for coloring is prepared. As in Chatin's algorithm, a node with degree <2 is repeatedly detached from the graph and pushed into the top of the queue. In Fig. 4 (c), the node E is detached, which successively reduces the degree of the node D to 1, resulting in Fig. 4 (d). Since there is no node with degree 1 in Fig. 4 (d), we decide to spill the node A, thus insert to the bottom of the queue as in Fig. 4 (e). Then, as both B and C have degree 1, we can push B, then C into the queue.

Once all the nodes are stored in the queue, we can pop out one node from the top of the queue at a time for coloring. As in Fig. 4 (f), we pop out C and color it as BLACK. Next, we can pop out B and color it as GRAY not to conflict with C as in Fig. 4 (g). After several steps including Fig. 4 (h), we encounter the situation in Fig. 4 (i) where A cannot be colored due to the conflicts with B and C. In a 2-coloring problem, such situation implies this graph is uncolorable, which requires layout modification in phase-assignment [10], but not necessarily in DPT. As in Fig. 4 (j), layout decomposition can be completed by splitting the polygon A into two parts at a cost of stitch on A.

Let us also consider the result of not selecting A in Fig. 4 (e). Although we decide to spill the node B instead of A as shown in Fig. 4 (k), it is still impossible to make the graph 2-colorable as in Fig. 4 (n). However, this will make the layout indecomposable as shown in Fig. 4 (o).

As a result, differently from the phase-assignment prob-



(a) This example shows how to color a set of grids in a routing path using Algorithm 1.

(b) This example has a potential conflict within a path.

(c) A simple coloring results in a conflict around the jog.

(d) Some grids become uncolored to resolve the conflict.

(e) Once the routing path is color, neighboring grids need to be shadowed by Algorithm 2.

Fig. 5. A routing path can be efficiently colored while minimizing the number of stitches, and its neighboring grids are shadowed for remaining unrouted/uncolored nets.

lem [3], the fact that a conflict graph is not 2-colorable does not guarantee the infeasibility of layout decomposition for the corresponding layout, because some conflicts can be resolved by stitches. The complexity of a layout decomposition for DPT with the minimum number of stitches is unknown yet, but we believe it is NP-hard, as there are many places for stitches.

B. DPT Consideration during Design

Layout decomposition is the most critical step for DPT, as discussed in Section II, especially in metal layers due to 2D patterns (while the poly layer has 1D patterns mostly). However, layout decomposition itself can be very complex and cannot be solved by a 2-coloring algorithm as discussed in Section III-A, which clearly requires design time consideration, more specifically during detailed routing. Current industrial effort is to first finish detailed routing, then perform layout decomposition (coloring all the polygons either in BLACK or GRAY) for DPT. If there is any uncolorable polygon, ripup/rerouting should be performed repeatedly to fix the conflict, resulting in long design-turn-around-time [6]. A detailed routing oblivious to DPT may generate highly complex patterns which may increase the uncolorable wirelength. Additionally, finding a decomposable layout is not sufficient for successful DPT processes; the number of stitches should be minimized to make a layout robust against overlay error. Therefore, it is critical to consider DPT in a correct-by-construction manner during detailed routing.

TABLE I
GRID STATE FOR DPT COLORING

grid state	description	grid color
BG	BI-colorable	Nearest color
$B\overline{G}$	BLACK-colorable	BLACK
$\overline{B}G$	GRAY-colorable	GRAY
$\overline{BG}$	Uncolorable	No color

IV. DPT-Friendly Detailed Routing

In this section, we propose our DPT-friendly detailed routing algorithm. As a first step, we propose a routing path coloring algorithm to minimize the number of stitches in Section IV-A, which provides two key observations for DPT-friendly detailed routing in Section IV-B.

A. Routing Path Coloring

For DPT-friendly detailed routing, it is critical to color a routed path with fewer stitches and shorter uncolored wirelength. Hence, we introduce a two-bit variable for each detailed routing grid to maintain colorability which will be one of the four states in Table I. As a grid with BG can be in either BLACK or GRAY, we have to find the best color for the grid in order to minimize the number of stitches.

Algorithm 1 Coloring_Path

```
Require: a path p
split p into a set of colorable subpaths by the \overline{BG} state
for each path t ∈ S do
   for each ordered grid d ∈ t do
      if d.state == \overline{B}G then
         Color d as GRAY
      else if d.state == B\overline{G} then
         Color d as BLACK
      end if
   end for
   for each ordered grid d ∈ t do
      if d.state == BG then
         Color d with the nearest color
      end if
   end for
end for
for each ordered grid d ∈ p do
   for each grid x whose distance from d < min_dp do
      if d.state == x.state and both colored and any
      uncolored grid or stitch exists between d and x then
         Uncolor x
      end if
   end for
end for
Color_Shadow(p)
```

Algorithm 2 Color_Shadow

Require: A path p

for each ordered grid $d \in p$ **do**
  for each grid x whose distance from $d < min_{dp}$ **do**
    if $x \notin p$ **then**
      if d is in BLACK **then**
        if $x.state == BG$ **then**
          $x.state == \overline{B}G$
        else if $x.state == B\overline{G}$ **then**
          $x.state == \overline{BG}$
        end if
      else if d is in GRAY **then**
        if $x.state == BG$ **then**
          $x.state == B\overline{G}$
        else if $x.state == \overline{B}G$ **then**
          $x.state == \overline{BG}$
        end if
      end if
    end if
  end for
end for

Algorithm 3 DPT-Friendly Detailed Routing

Require: A set of blockages B, a set of nets N

1: layout decomposition and color shadowing of B
2: **for** each net $n \in N$ **do**
3:   s = source grid of n
4:   t = target grid of n
5:   A priority queue $Q = \{s\}$
6:   **while** Q is not empty **do**
7:     x = dequeue from Q
8:     **if** x==t **then**
9:       break
10:     **end if**
11:     **for** each adjacent grid d of x **do**
12:       $cost = x.cost + 1 + A^{*}cost$ //unit wirelength is 1
13:       **if** $x.state$==$B\overline{G}$ and $d.state$==$\overline{B}G$ **then**
14:         $cost+=\alpha$ //to discourage a stitch
15:       **else if** $x.state$==$\overline{B}G$ and $d.state$==$B\overline{G}$ **then**
16:         $cost+=\alpha$ //to discourage a stitch
17:       **else if** $d.state$==$\overline{BG}$ **then**
18:         $cost+=\beta$ //to reduce uncolorable wirelength
19:       **end if**
20:       **if** x and d not on the same layer **then**
21:         $cost+=\gamma$ //to discourage too many vias
22:       **end if**
23:       **if** $d.cost > cost$ **then**
24:         $d.cost = cost$
25:         $d.prev = x$
26:         enqueue d to Q
27:       **end if**
28:     **end for**
29:   **end while**
30:   p = Backtrace from x to s of n
31:   Coloring_Path(p)
32: **end for**

Our coloring algorithm for a routing path is proposed in Algorithm 1. To reduce the problem size, we slice a path into multiple subpaths in line 1, if there is any grid in the $\overline{BG}$ state. Next, we color grids in either the $B\overline{G}$ or $\overline{B}G$ state, as they have a single option in lines 2–9. For remaining grids which are in the BG state, we color each one with the nearest color along the corresponding subpath in lines 10–15. Since there can be within-path conflicts, we also perform post-processing in line 16–22. Once a path is colored, we shadow around the path in line 23, which is described in Algorithm 2, to update the states of nearby grids. We visit grids which are within min_{dp} distance from the path in order to update their colorability.

Assume that a routing path with 14 uncolored grids at various states as shown in Fig. 5 (a). We begin by splitting the path into three subpaths, X, Y, and Z, as in line 1 of Algorithm 1. For each subpath, we first color grids in the $B\overline{G}$ or $\overline{B}G$ state. Then, we color remaining grids in the BG state, by identifying the nearest color within the same subpath, as shown by the arrows in Fig. 5 (a). When a subpath consists of only grids in the BG state like subpath Z, we color them randomly. Finally, we assemble subpaths and grids in the $\overline{BG}$ state into one colored path.

For some case, there can be conflicts within a path. Consider Fig. 5 (c) where there is a jog. If we color the path in Fig. 5 (b) as done in Fig. 5 (a), we will have Fig. 5 (c) where there is a conflict. Therefore, as the routing path is given and fixed, we need to detect the conflict and further resolve it by uncoloring some grids in GRAY as shown in Fig. 5 (d), which is done in lines 16–22 of Algorithm 1. Fig. 5 (e) shows the states of nearby grids after color shadowing. Note that a grid which is close to both BLACK and GREY becomes in the $\overline{BG}$ state.

TABLE II
LOOKUP TABLE FOR DPT ROUTING

case	current grid state	next grid state	penalty
1	$B\overline{G}$	$\overline{B}G$	α (stitch)
2	$\overline{B}G$	$B\overline{G}$	α (stitch)
3	any state	$\overline{BG}$	β (uncolorable)

We can make two observations with the example in Fig. 5: (a) having a grid in the $\overline{BG}$ state on a path will result in layout decomposition failure; (b) having two grids in the $B\overline{G}$ and $\overline{B}G$ states adjacent along a path will result in a stitch.

B. Detailed Routing Algorithm

According to the observations in Section IV-A, we will penalize three cases in Table II during detailed routing as shown in Algorithm 3. In line 1, we perform layout decomposition for existing routing blockages (e.g., pins, power/ground, clock, and so on) using Chatin's algorithm [5] as done in Section III-A. When we need to spill a node, we pick one corresponding to the largest polygon. Next, we perform color shadowing

TABLE III

PERFORMANCE OF THE PROPOSED DPT-FRIENDLY DETAILED ROUTING ALGORITHM.

design	nets	area	router	wirelen(mm)			via	runtime (sec)			double patterning		Ratio		
		(um^2)		M1	M2	sum	M1-M2	router	decomposition	sum	stitch[a]	failure(um)[b]	via	runtime	stitch
test1	6K	15K	DR+LD	1.97	8.7	10.6	189	91.7	1099.6	1191.3	109	7.75	1	3.27	21.8
			DPFR	1.98	8.7	10.7	216	364.6	0	364.6	5	0.15	1.14	1	1
test2	8K	12K	DR+LD	0.54	10.2	10.8	57	120.2	1517.1	1697.3	92	10.60	1	4.33	92
			DPFR	0.54	10.2	10.8	59	392.9	0	392.9	1	0	1.04	1	1

[a] The number of stitches.

[b] The uncolored wirelength due to irresolvable conflicts.

around the colored blockages to guide detailed routing. Then, we perform a typical detailed routing algorithm based on A* search as found in line 12. However, to find a DPT-friendly path, we modify cost from lines 13 to 22. From lines 13–16, we add α penalty to the routing cost to discourage stitches from the case 1 and 2. And, in line 18, we also increase the routing cost by β to minimize the number of uncolored grids. In line 21, we can see one more penalty term γ which is to minimize the number of vias, as decomposability or stitch count can be improved at a cost of via. Once the minimum cost path is found, we can apply Algorithm 1 as in line 31.

V. Experimental Results

We implement our DPT-friendly routing in C++ and test on a 3.0 GHz Linux machine with 16G RAM. We scale down two industrial ASIC designs from 65nm to 32nm for evaluation.

For though comparison, we prepare two detailed routing algorithms for DPT, **DR+LD** (**D**etail **R**outing + **L**ayout **D**ecomposition) and **DPFR** (**D**ouble **P**atterning **F**riendly **R**outing). For layout decomposition in **DR+LD**, we use the same function in Algorithm 3 (See Section IV-B). We first run a grid-based detailed router followed by layout decomposition in **DR+LD** which is according to the current industrial effort [6], but layout decomposition and detailed routing are simultaneously performed by Algorithm 3 in **DPFR**.

We compare **DPFR** and **DR+LD** on four test designs with $\alpha = 9$, $\beta >> 10$, and $\gamma = 6$ as shown in Table III, which demonstrates the effectiveness of **DPFR**, a simultaneous layout decomposition and detailed routing for DPT. With negligible overhead in wirelength, we can improve the quality of layouts in terms of double patterning; the number of stitches for every design is reduced by at least 21x and up to 92x, and the uncolorable wirelength is at most 0.15μm while **DR+LD** has at best 7.75μm. Note that the uncolorable wirelength from **DPFR** is due to DPT-oblivious pin locations. Via overhead is 9% on average. Even though **DPFR** is slower than the routing portion of **DR+LD**, **DPFR** is at least 3x faster considering the overall flow. It is mainly because **DR+LD** has to work on a larger conflict graph for the final layout decomposition.

VI. Conclusion

Double patterning technology (DPT) is the current forerunner lithography solution for 32/22nm technology nodes, due to delayed deployment of EUV for mass production. In this paper, we present the first DPT friendly detailed routing algorithm which performs routing and layout decomposition in one shot, in a correct-by-construction manner. Experimental results show that our approach outperforms the current industrial sequential approach (routing, and then layout decomposition) by wide margin, for both quality of results and runtime. We plan to research on DPT compatible standard cell design techniques and DPT aware placement algorithms.

References

[1] 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 6521*, 2007.

[2] C. Bencher, Y. Chen, H. Dai, W. Montgomery, and L. Hul. 22nm half-pitch patterning by cvd spacer self alignment double patterning (sadp). In *Proc. of SPIE*, volume 6924, 2008.

[3] P. Berman, A. B. Kahng, D. Vidhani, H. Wang, and A. Zelikovsky. Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks. In *Proc. Int. Symp. on Physical Design*, Apr 1999.

[4] M. Born and E. Wolf. *Principles of Optics, 6th Edision*. Pergamon Press, 1980.

[5] G. J. Chatin. Register Allocation & Spilling via Graph Coloring. In *Proc. Symp. on Compiler Construction*, 1982.

[6] M. Drapeau, V. Wiaux, E. Hendrickx, S. Verhaegen, and T. Machida. Double Patterning Design Split Implementation and Validation for the 32nm Node. In *Proc. SPIE 6521*, 2007.

[7] 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 6349*, 2006.

[8] 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*, volume 6925, 2008.

[9] A. B. Kahng. Key Directions and a Roadmap for Electrical Design for Manufacturability. In *Proc. European Solid-State Circuits Conf*, 2007.

[10] A. B. Kahng, S. Vaya, and A. Zelikovsk. New Graph Bipartizations for Double-Exposure, Bright Field Alternating Phase-Shift Mask Layout. In *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2001.

[11] Y. Ku. Lithography challenges and solution for 32nm node and beyond. In *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2008.

[12] K. Lucas, C. Cork, A. Miloslavsky, G. Luk-Pat, L. Barnes, J. Hapli, J. Lewellen, G. Rollins, V. Wiaux, and S. Verhaegen. Interactions of double patterning technology with wafer processing, opc and design flows. In *Proc. of SPIE*, volume 6924, 2008.

[13] J. Park, S. Hsu, D. V. D. Broeke, J. F. Chen, M. Dusa, R. Socha, J. Finders, B. Vleeming, A. van Oosten, P. Nikolsky, V. Wiaux, E. Hendrickx, J. Bekaert, and G. Vandenberghe. Application Challenges with Double Patterning Technology (DPT) beyond 45nm. In *Proc. SPIE 6349*, 2006.

[14] J. Rubinstein and A. Neureuther. Post-decomposition assessment of double patterning layout. In *Proc. of SPIE*, volume 6924, 2008.

[15] A. Sezginer and B. Yenikaya. Double Patterning Technology: Process-Window Analysis in a Many-Dimensional Space. In *Proc. SPIE 6521*, 2007.

[16] A. Vanleenhove and D. Steenwinckel. A litho-only approach to double patterning. In *Proc. of SPIE*, volume 6521, 2007.

[17] V. Wiaux, S. Verhagen, 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*, volume 6924, 2008.