SPD12: Layout Decomposition and Legalization for Double-Patterning Technology

Rani S. Ghaida, *Student Member, IEEE*, Kanak B. Agarwal, *Senior Member, IEEE*, Sani R. Nassif, *Fellow, IEEE*, Xin Yuan, Lars W. Liebmann, Puneet Gupta, *Member, IEEE*

***Abstract*—The use of multiple-patterning optical lithography for sub-20nm technologies has become inevitable with delays in adopting the next generation of lithography systems. The biggest technical challenge of multiple patterning is failure to reach a manufacturable layout-coloring solution, especially in dense layouts. This paper offers a post-layout solution for the removal of conflicts, i.e., patterns that cannot be assigned to different masks without violating spacing rules. The proposed method essentially consists of three steps: layout coloring, exposure layers and geometric rules definition, and, finally, layout legalization using compaction and multiple-patterning rules as constraints. The method is general and can be used for different multiple-patterning technologies including LELE double-patterning (DP), triple/multiple-patterning (i.e., multiple litho-etch steps), and self-aligned double patterning (SADP). For demonstration purposes, we apply the proposed method in this paper to remove conflicts in DP. We offer an O(n) layout-coloring heuristic algorithm for DP, which is up to 80X faster than the ILP-based approach. The conflict-removal problem is formulated as a linear program (LP), which permits an extremely fast run-time (less than 1 minute in real time for macro layouts). The method was tested on standard cells and macro layouts from a commercial 22nm library designed without any multiple-patterning awareness; for many cells, the method removes all conflicts without any area increase; for some complex cells and macros, the method still removes all conflicts but with a modest 6% average increase in area.**

***Index Terms*—design for manufacturability, multiple-patterning technology, double-patterning technology, lithography, layout compaction, layout legalization, design rules.**

I. Introduction

The use of double/multiple-patterning (DP/MP) optical lithography for sub-20nm technologies has become inevitable with delays in adopting the next generation of lithography systems. One of the most favorable MP alternatives is pitch-split DP where layout patterns are formed with two separate exposure and etch (or develop) steps (i.e. litho-etch-litho-etch process). Hereafter, we will use the term DP to denote pitch-split DP.

For a layout to be DP manufacturable, layout features that violate the minimum spacing of single patterning (a.k.a. minimum same-mask or same-color spacing) must be assigned to different masks. The biggest technical challenge of multiple patterning is failure to reach a manufacturable mask-assignment solution, especially in dense layouts. Layouts designed with conventional rules are generally incompatible with DP; whereas, designing layouts with DP rules is a burden for the designer and requires enormous manual effort. This paper offers an automated post-layout solution for adapting conventional layouts to MP technology.

DP mask assignment is essentially a two-color labeling problem [2] and is often referred to as DP coloring or DP layout decomposition. In DP coloring, the layout is represented with a *conflict graph*, where nodes represent layout polygons to be colored and arcs represent coloring constraints. An arc between two nodes denotes a manufacturing constraint on the two corresponding layout polygons to color with two different colors. This constraint is necessary to ensure the printability of non-touching polygons assigned to the same exposure and separated by a distance smaller than the minimum same-color spacing rule. A conflict graph is colorable with two colors and no constraint violations only when the graph contains no odd cycles, i.e. cycles with odd number of arcs; and, an odd cycle is referred to as a coloring conflict.

The difference between DP coloring and the labeling problem of graph theory is that a layout polygon can be a composite of layouts of different masks. The splitting of polygons into multiple parts on different masks is known as *stitching* and the location where the two masks join is called a *stitch*. Although stitching complicates the labeling problem, it is an efficient and practical method to conform many, originally DP-unfriendly, layout patterns to DP. In particular, stitching is used to break some odd cycles in the conflict graph getting rid of some coloring conflicts (as illustrated by the example of Figure 1). Even with stitching, many patterns cannot be assigned to the two masks without violation of the minimum same-color spacing. Such patterns are called native DP conflicts and resolving these conflicts – with certain layout perturbation – is the biggest challenge facing the deployment of DP.

Manuscript received July 10, 2012; revised August 31, 2012; accepted November 12, 2012. Date of publication XXXXX; date of current version XXXXX. The authors would like to thank Kevin W. McCullen and Rani R. Narayan from IBM for their help with setting up the minimum perturbation based VLSI artwork legalization system [1]. The authors would also like to thank Andrew B. Kahng from UCSD and Hailong Yao from Tsinghua University for providing layout test cases that allowed results comparison with their work.

R. S. Ghaida was with the University of California, Los Angeles, CA 90095, USA. He is now with GLOBALFOUNDRIES Inc., Milpitas, CA 95035, USA (e-mail: ranighaida@gmail.com).

K. B. Agarwal and S. R. Nassif are with IBM Corp., Austin, TX 78758, USA (e-mail: kba@us.ibm.com; nassif@us.ibm.com).

X. Yuan is with IBM Corp., San Jose, CA 95120, USA (e-mail: xinyuan@us.ibm.com).

L. W. Liebmann is with IBM Corp., Hopewell Junction, NY 12533, USA (e-mail: lliebman@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 00.0000/TCAD.2012.00000

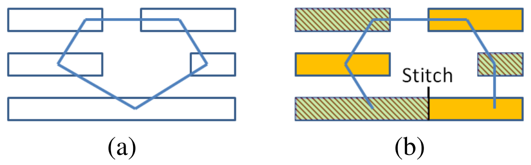

Figure 1. Example of a layout with odd cycle in its conflict graph (a) that was broken by introducing a stitch (b).

A. Prior art in DP coloring

Prior works in DP coloring differ mainly by the way stitches are dealt with. Rule-based stitching where polygons are split at certain fixed locations is proposed in [3, 4]. The drawback of this method is that many stitch locations cannot be found by the rules.

In [5, 6], the layout is segmented into rectangles, stitches that can resolve DP conflicts are determined, and the coloring problem with stitch minimization is formulated as an integer linear program (ILP). Segmentation of the layout into rectangles has many drawbacks. First, it complicates the problem as it forces the consideration a lot of extra stitch locations that should never be used. Consider the example of Figure 2. Rectangle C has same-color spacing violations with both rectangles A and B. As a result, A and B must always be assigned to the same mask to avoid a DP conflict and the stitch location at the joint of A and B is never used. The second drawback of segmentation is that it makes the handling of multiple same-color rule values difficult. Because rectangles are mapped into nodes, there is no easy way to distinguish between side-to-side (S2S), tip-to-side (T2S), and tip-to-tip (T2T) same-color spacing rules that may have different values in modern technologies. For example, the left vertical edge of shape A in Figure 2, which corresponds to a line-side, will be confused with a tip using conventional polygon-identification and manipulation engines such as Calibre. Also, handling the bottom horizontal edge of shape B is complicated as it is affected by the actual coloring; it should be considered a tip when the two shapes are colored differently or ignored when the two shapes are colored with the same color. Another drawback of the methods of [5, 6] is that ILPs are very time consuming to solve (NP-hard problem [7]). In addition, the method in [6] can only be applied to gridded layouts with a grid size equal to half the pitch, which is not the case for many layers (e.g., M1).

The work of [8] proposes a graph-reduction method to reduce the size of the coloring problem. The method avoids segmentation of the layout into rectangles and its associated drawbacks. On the downside, the method formulates the problem as a maximum-cut problem, which is an NP-complete problem, and solves it using ILP.

The more recent work of [9] formulates the coloring problem with stitch minimization as a minimum-cut problem and solves the problem in $O(n^{1.5} \log n)$. In [10], a method for DP coloring with multiple objectives including stitch minimization is proposed. The method is based on min-cut partitioning and the problem is solved in a polynomial time algorithm. These methods are also based on the segmentation of the layout into rectangles and cannot handle multiple same-color spacing rules. The work in [11] offers a method to speed up the coloring process through graph partitioning.

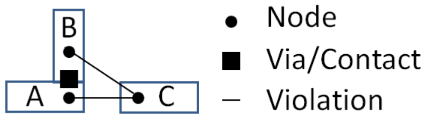

Figure 2. Illustration of the drawbacks of segmenting the layout into rectangles, which is performed in prior art of DP coloring and conflict removal.

B. Prior art in conflict removal

Prior art in layout perturbation to resolve DP conflicts [12–14] generally formulates the problem as an ILP (except [14]). Moreover, all previous works segment the layout into rectangles and move rectangles around to eliminate DP conflicts.

Working with rectangles has the same drawback discussed earlier and some additional drawbacks. The problem is further complicated because the automated layout perturbation solver (ILP or compaction) needs to maintain the connectivity of rectangles at joints (e.g., L-shape) through additional constraints. Moreover, because the constraints of the solver are defined between rectangles, overlap rules with features from the top and bottom layers cannot be handled correctly. Consider again the example of Figure 2 where an L-shape metal overlaps with a via (or contact) at the corner. If the via movement is blocked, the solver will try to move shapes A and B so that *each* covers the via *completely*. Not only these moves are unnecessary because the via is initially covered, but they can also impact the layout area and the effectiveness of the conflict removal.

In [12], DP requirements are added to the ILP constraints to perform DP-aware layout migration while minimizing area and layout perturbation. In addition to the problems with segmentation, the method leads to unmanageable number of constraints, excessive runtime to solve the ILP, and does not work well when the layout contains DP conflicts initially (i.e. not migrated from a previous generation).

In [13], wire spreading is proposed to remove DP conflicts. All wire-spreading options that reduce DP conflicts are precomputed and conflicts and wire moves are minimized in the ILP. In addition to the problems common to all prior works that are discussed earlier, wire spreading can reduce the number of conflicts by a modest amount (as the results in [13] show). Many conflicts can be resolved with edge-location adjustment and wire-width reduction but not with wire spreading. Moreover, to avoid creating new DP conflicts, the method only moves segments when their spacing from all neighboring wires after the movement is at least equal to the same-color spacing. In many actual cases however, we may be able to move the segment to a closer distance from its neighbors – equal to the different-color spacing (typically half the same-color spacing) – and still avoid creating new conflicts[1]. The method of [13] cannot detect such cases and unnecessarily limits the wire spreading because, otherwise, the entire graph will have to be checked for newly created conflicts for every wire-spreading option.

Rather than solving the problem with an ILP, the work in [14] applies traditional layout compaction – based on minimum-area metric – iteratively as long as DP conflicts are reduced. At each iteration, the process of DP-compliance

[1]When the segment is assigned a different color than its neighbors.

checking, which includes pattern projection [5], segmentation, conflict graph generation, and odd cycle detection, is performed initially. DP constraints at odd cycles only are then generated and a trial compaction is performed. The DP-compliance check is repeated and, if the number of odd cycles is reduced, the DP-constraints are permanently committed. In addition to the problems associated with segmentation into rectangles and the large runtime of iterative compaction and performing the DP-compliance check twice at each iteration, the method is not effective in removing DP conflicts and keeps a large number of conflicts unresolved (as reported in [14]). Because DP constraints are generated only at odd cycles, resolving one conflict may create a new conflict in other parts of the layout. As a result, the iterative compaction may stop without removing many DP conflicts that otherwise could have been resolved. In our work, we were able to remove DP conflicts efficiently, effectively, and simultaneously across all layers. This was made possible by essentially defining DP constraints all over the layout in terms of DRs – after an initial coloring that minimizes the number of conflicts – and applying linear programming-based layout compaction once across all layers.

C. Our approach

In this paper, we extend our work presented in [15]. This work offers an automated post-layout solution for adapting conventional layouts to MP technology. The proposed method essentially consists of three steps: layout coloring, exposure layers and geometric rules definition, and, finally, layout legalization using compaction and multiple-patterning rules as constraints. The method is general and can be used for different multiple-patterning technologies including LELE DP, tripe/multiple-patterning with multiple litho-etch steps, and self-aligned double patterning (SADP). For demonstration purposes, we apply the proposed method in this paper for DP in LELE process.

We follow a different approach for the DP coloring than prior works. Specifically, we use DR-dependent projection to determine the features that may cause DP conflicts and their actual, possibly non-rectangular, shapes (as in [8]). We then formulate the problem as a labeling problem that we solve in a $O(n)$ heuristic algorithm. In our method, all candidate stitches that can be useful are automatically identified and are reduced by the algorithm. Because we use all candidate stitches, our method guarantees a conflict-free coloring solution when the layout has no native conflicts (i.e., conflicts that cannot be resolved with stitching).

Using a linear program (LP), DP conflicts are removed and the layout is legalized simultaneously across multiple layers by edge-based layout perturbation. This layout legalization is performed through layout compaction formulated as a *minimum perturbation problem*, unlike [14] that uses minimum-area metric for compaction[2]. The proposed methodology allows the layout designer to design with conventional single-patterning layers and DRs, masking the complexity of dealing with double-patterning layers and requirements.

Our proposed methodology for designing DP-compatible layouts is depicted in Figure 3. Using existing non DP-compatible layouts or layouts designed from scratch using conventional rules, we perform an optional step of layout simplification at DP layers for the possible sacrifice of non-crucial parts as described in Section IV. We then carry out DP coloring while considering all candidate stitch locations. If the layout contains DP native conflicts, the conflicts are removed and the layout is legalized simultaneously across all layers while minimizing layout perturbation using a LP and maintaining the same area as the original layout. Optionally, in case some native conflicts remain unresolved, the LP-based layout legalization is repeated while allowing an area increase to remove more DP conflicts (all conflicts are removed after this step in most cases).

[2]The advantages of minimum layout-perturbation metric over the minimum area metric for layout compaction are discussed in [1, 16].

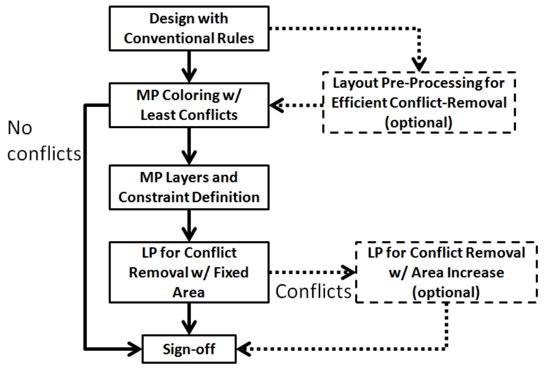
Design with Conventional Rules
Layout Pre-Processing for Efficient Conflict-Removal (optional)
MP Coloring w/ Least Conflicts
No conflicts
MP Layers and Constraint Definition
LP for Conflict Removal w/ Fixed Area
LP for Conflict Removal w/ Area Increase (optional)
Conflicts
Sign-off

Figure 3. The flow for our proposed method to achieve DP-enabled layout design.

We make the following contributions.

- We offer a framework for DP coloring and legalization while *minimizing layout perturbation*. The framework guarantees the legalization of the layout across all layers simultaneously and achieves conflict-free standard-cell/macro layouts that are fully compatible with DP.
- We propose a $O(n)$ heuristic algorithm for DP coloring that guarantees a conflict-free solution for layouts without native conflicts by using all candidate stitch locations.
- We formulate the problem of conflict removal as a LP, which can be solved in polynomial time [17], as opposed to prior art of conflict removal that formulate the problem as an ILP, which is NP-hard [7].
- We handle, during coloring as well as legalization, complex same-color spacing rules including tip-to-tip and tip-to-side in addition to the minimum spacing, unlike previous works that only handles a single same-color rule value.

The remaining paper is organized as follows. Section II describes our coloring approach, which handles complex same-color spacing rules and guarantees conflict-free solution for layouts without native conflicts. Section III presents our method for DP conflict removal and layout legalization based on minimum perturbation. A method to improve the effectiveness of the conflict removal is described in Section IV. Section V shows how DP-compatible designs can be achieved after applying the proposed methodology for standard cells. Section VI presents the experimental results, while Section VII concludes the paper.

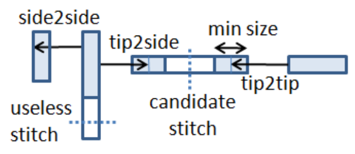

Figure 4. DR-dependent projection to identify violating parts and stitch locations. Violating parts are the blue features and non-violating parts are the clear features.

II. DP Coloring

We follow a different approach for the DP layout decomposition than what is presented in the literature. We first find all parts of the layout that have same-color spacing violations with neighboring features and, then, we assign these violating parts to the two masks. In this way, candidate locations of stitches are automatically defined and can be easily minimized as we show later in this section. In the end, non-violating parts can be assigned to either mask. If a non-violating part touches features of the same mask, we assign it to that same mask to avoid introducing extra stitches; whereas, if a non-violating part touches features of different masks, we assign it to both masks to maximize the overlap region of the masks. The details of this implementation follow.

A. *Multiple-spacing rules projection*

We start with DR-dependent projection to identify violating parts as illustrated in Figure 4. From each edge in the layout, side or tip, we project to the neighboring features and determine neighboring edges with which the corresponding same-color spacing rule is violated[3]. From the violating edges and based on the values of the corresponding spacing violation, we determine the exact parts of the layout that violate the same-color spacing with their neighbors. Violating parts that are smaller than the minimum feature size allowed on a single mask are grown within polygons of the original layer to meet the minimum requirement.

Unlike previous works that can only allow a single same-color spacing rule, we allow three same-color spacing rules with different values: side-to-side (S2S), tip-to-side (T2S), and tip-to-tip (T2T). When a single same-color spacing rule is allowed, the largest spacing rule value must be used as the minimum same-color spacing to ensure no DP conflicts are missed. The advantage of allowing multiple same-color spacing rule-values is crucial whenever the values of spacing rules differ, which is the case in latest technologies. The importance of allowing different values for the different rules will be quantified in Section VI.

B. *Coloring objectives*

The main objective of DP coloring is to assign features to the two different masks with the minimum number of conflicts. A secondary objective is to minimize stitches, which may increase yield loss due to overlay error between the first and second exposure layers. Because stitches can remove certain conflicts (as illustrated in Figure 1), we consider all possible stitch locations during coloring and get rid of stitches that do

[3]This can be done using existing DRC tools and in a similar fashion as in [5].

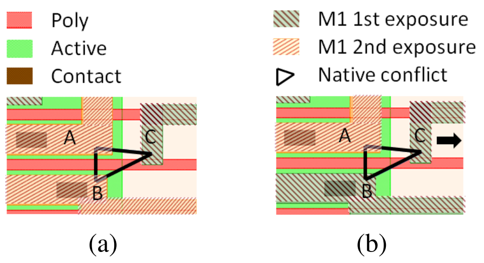

Figure 5. Odd cycle coloring can affect the efficiency of conflict removal. In (a), the conflict is on M1 between shapes A and B and can only be fixed if the gates are spaced apart and area is increased; in (b), the conflict is on M1 between shapes B and C and can be fixed by moving C in the direction of the arrow without increasing area.

not affect the number of conflicts. If a stitch is introduced inside any violating part, then one of the stitch's sides will have to be assigned to the same mask as the neighboring part that created the violation, which leads to a new DP conflict (as in Figure 2). As a result, stitches should be located only in non-violating parts. Since DP conflicts are between violating parts only, stitches are beneficial (i.e. may reduce the number of conflicts) only if placed in non-violating parts that separate two or more violating parts. In other words, a single stitch is sufficient in such non-violating parts and stitches in a non-violating part that connect to a single violating part is useless because we can always assign such non-violating part to the same mask as the connected violating part (see example of Figure 4). In addition, stitches that cannot guarantee the minimum overlap length of the two masks are disregarded (by joining the connected violating parts).

Although an odd cycle will always result in a DP conflict no matter the coloring, deciding what features go on the same mask can affect the efficiency of the conflict removal. To see how, consider the example of Figure 5. This layout contains an odd cycle between shapes A, B, and C. In Figure 5(a), the coloring solution leads to a conflict between shapes A and B that can be resolved only if the gates are spaced apart and, consequently, the layout area is increased; whereas, in Figure 5(b), the coloring solution results in a conflict between shapes B and C that can be resolved by moving C to the right without increasing the layout area. To take advantage of this observation, we make violations in the orthogonal orientation of gates (vertical violations for our layouts) more critical than the ones in other orientations (horizontal and diagonal violations for our layouts). Similarly, we make horizontal violations more critical than diagonal violations because the latter typically require less additional separation to fix.

C. *Implementation details*

The coloring of violating parts is straightforward and its first-stage initial coloring is performed in $O(n)$, where n is the number of violations and candidate stitches. An example that illustrate the coloring steps is given in Figure 6 and the details of the algorithm are presented in Figure 7.

We start by constructing the conflict graph, where violating parts are represented by nodes and violations and stitches are represented by arcs. We represent vertical violations by solid arcs, horizontal violations by dotted arcs, diagonal violations by double-line arcs, and stitches between two shapes by arcs

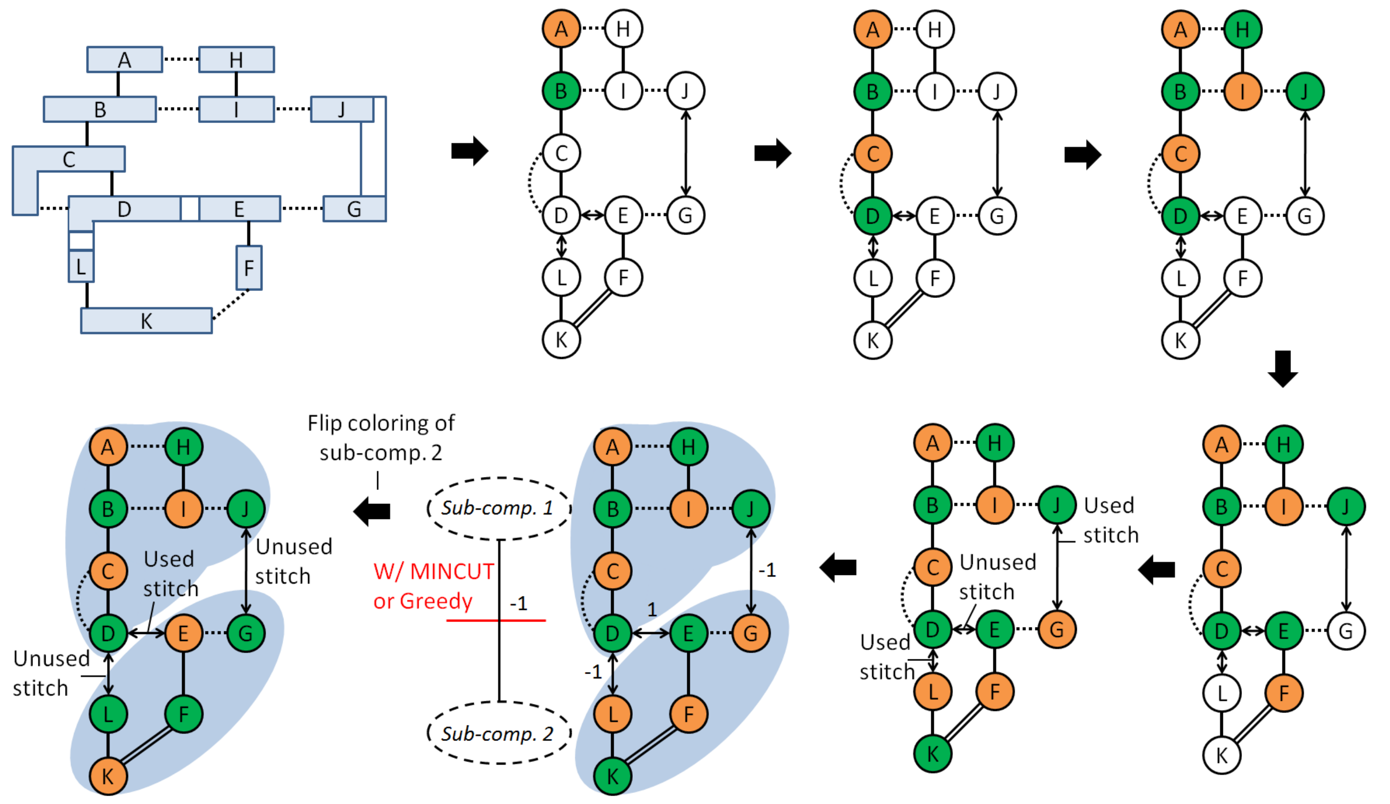

Figure 6. An illustrating example showing each step of the coloring process for an isolated region of the layout.

with two-sided arrows. For each connected component (identifying connected components is $O(n)$), we pick a violation-arc with preference to vertical over horizontal and horizontal over diagonal arcs and assign the two connected nodes to different masks. Whenever a new node is assigned, its connected arcs get added to first-in-first-out (FIFO) queues of the different types of violations and stitches to be processed next. A new arc (possibly a stitch-arc) is popped from the different queues with preference to violation-arcs over stitch-arcs and the same preference for the different violation-arcs as before. This process is repeated until all arcs in the component are processed. Each node is assigned only once: when a violation-arc is processed, the two nodes are assigned to different masks and, when a stitch-arc is processed, the two nodes are assigned to the same masks.

We perform a second-stage coloring where the initial coloring is possibly flipped to further reduce the number of used stitches. Each part of a component that is connected with violation-arcs only (without stitches) is called a sub-component and stitches connect different sub-components (see Figure 6). Each sub-component has a flipping score based on which the coloring of its nodes is flipped or preserved. When a stitch is processed, we record the connection of the two connected sub-components; if the stitch is used, the flipping score is decremented by one (the score being zero initially); if the stitch is unused, the flipping score is incremented by one. So, by flipping a sub-component with a negative score, the number of stitches is reduced by the amount of the score.

We follow two approaches to determine the sub-components where color-flipping is beneficial. The first is based on a greedy heuristic algorithm where sub-components with negative scores are flipped in a decreasing order of scores. When a sub-component is flipped, the sub-component and its neighbors are prevented from future flipping (i.e., flipping is locked). Although this algorithm may reach a sub-optimal solution, it ensures $O(n)$ running time for the overall coloring procedure. The most suboptimal solution occurs for the case shown in Figure 8. Here, the greedy algorithm will only color-flip the center node with the highest flipping-score N. The optimal solution is to flip every other neighbor of the center node with a score of $(N-1)$. Since the center node has N neighbors, the optimal solution results in $N\frac{N-1}{2}$ less stitches than the initial coloring solution and $N\frac{N-3}{2}$ stitches less than the solution obtained with the greedy algorithm. This worst case and similar bad scenarios are uncommon for actual layouts, however, and the sub-optimality of the greedy flipping is limited in practice as we will show in our experimental results (Section VI).

In the second flipping approach, the flipping problem is formulated as a minimum-cut problem (inspired by the minimum-cut formulation for graph coloring of [8, 9]). We construct a flipping graph where nodes represent sub-components and every edge represents one or more candidate stitches between two-sub-components. A used stitch is associated with a flipping score of -1 and an unused stitch is associated with a flipping score of 1. The sum of flipping scores of all candidate stitches between any two-sub-components determines the edge's weight as shown in Figure 6. Now, the problem is equivalent to partitioning the graph into two parts: one part where the coloring will be flipped and another part where the coloring will be preserved. It is easy to see that the problem is solved optimally by partitioning the graph based on the

```
1: Perform DR-dependent projection (see Section II-A).
       Identify violating parts.
       Identify all useful candidate stitches.
2: Construct conflict graph with nodes representing violating parts and
   four types of arcs representing vertical violations, horizontal viola-
   tions, diagonal violations, and stitches.
3: Create separate FIFO queues for the different types of arcs.
4: Determine connected components.
5: Determine connected sub-components (i.e. without stitch connections).
6: for all Connected components do
7:     Pick any violation-arc with preference to vertical over horizontal
       and horizontal over diagonal and assign its nodes to different
       masks.
8:     Push all arcs connected to the assigned node into FIFO queues of
       the different types of arcs.
9:     Pop an arc (possibly a stitch-arc) from the different queues with
       preference to violation-arcs over stitch-arcs and the same prefer-
       ence for the different violation-arcs as above.
10:    Assign the two nodes connected to the arc to different masks if
       the arc is for a violation and to the same mask if the arc is for a
       stitch.
11:    if Arc is a stitch then
12:        Record the connection of the two sub-components (the two
           nodes connected to the stitch-arc belong to different sub-
           components).
13:        If the stitch is used (i.e. connected nodes were assigned to
           different masks), increment the flipping-score of the two sub-
           components by one.
14:        If the stitch is unused (both connected nodes assigned to same
           mask), decrement the flipping-score of the two sub-components
           by one.
15:    end if
16:    Repeat steps 9 to 15 until all arcs in the component are processed.
17: end for
18: for all Sub-component with a positive flipping score sorted by
    descending score do
19:    Skip if already processed or marked not to be processed
20:    Mark as flipped and processed and mark its neighbors as processed
       (to prevent future flipping)
21: end for
22: for all Nodes do
23:    Flip node if it belongs to a flipped sub-component
24: end for
```

Figure 7. Overview of the $O(n)$ coloring procedure with greedy algorithm for color-flipping[4].

minimum cut with the smallest sum of weights (as it was also shown in [9]). In case the conflict graph has no DP conflicts (i.e., no odd-cycles), such solution gives the minimum number of stitches; in case the graph has DP conflicts, however, optimal flipping may not correspond to optimal overall number of stitches because the way the odd cycle is colored can affect the number of stitches.

For finding the minimum cut, we use Stoer and Wagner's MINCUT algorithm [18], which has a running time of $O(n \log(n))$. If the minimum cut has a negative value, the coloring of sub-components of one of the partitions (i.e., from one side of the cut) is flipped to reduce the number of used stitches by the absolute value of the cut.

It is important to note that, for the MINCUT algorithm we used [18] to work properly and to enhance the run-time, the algorithm is applied for each connected component of the conflict graph separately since the coloring of connected components can be performed independently.

[4]Although the loop of line 18 to 21 is theoretically higher than $O(n)$, it takes much less time to execute than the $O(n)$ loop of line 6 to 17 because neighbors of flipped sub-components are skipped and the number of neighbors for a sub-component is less than 3 in most cases and at most 10 in practice.

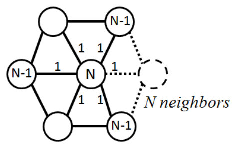

Figure 8. Case of most suboptimal solution for greedy-based flipping.

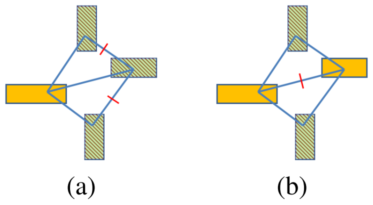

Figure 9. Example showing two coloring solutions that our method may give for the same layout (rare case with all diagonal violations) depending on the propagation order of the coloring: (a) with two DP violations and (b) with a single DP violation.

D. Stitches vs. conflicts and special cases

Stitches are manufacturable, DP conflicts are not. When the requirement for the minimum mask overlap is met, stitches are safe to manufacture and their minimization is recommended rather than required. Moreover, stitches may occur in millions in large layouts and reducing the number of stitches by few percents does not have a significant impact on the manufacturing yield. On the other hand, a layout with a single DP conflict can never be manufactured. As a result, our primary objective in this work was to achieve a coloring solution with the least number of DP conflicts. Although our method minimizes the number of stitches, it does not guarantee achieving the minimum number of stitches. Most importantly, because we consider all candidate stitch locations, our method guarantees to reach a solution with DP violations only at the locations of *native conflicts* (i.e. conflicts that cannot be resolved with stitching) and a conflict-free solution for layouts without *native conflicts*. A DP native conflict is defined as an odd cycle in the conflict graph that cannot be resolved with stitching. By performing the coloring with a conflict graph that includes all candidate stitches and while ensuring any two nodes with a violation that are not part of an odd cycle are assigned to different masks, our method leads to a solution with zero *non-native conflicts* (i.e. conflicts that are resolvable with stitches).

For some special cases with two or more native-conflict odd cycles share some of their arcs, our method may lead to a solution with non-minimum number of *DP violations* at such native conflicts depending on the propagation order of the coloring. One such special case is shown in Figure 9. Because we set a propagation preference with purely vertical violations first, diagonal violations second, and purely horizontal violations last, all violations in this four-tip configuration must be diagonal violations for the method to result in the coloring solution with two DP violations (Figure 9(a)) for some propagation order; otherwise the method will result in the coloring solution with a single DP violation as in Figure 9(b). Besides the peculiarity of this layout, such four-tip configuration may never occur because contacts/vias are on tracks in actual layouts. Furthermore, the number of DP violations may not reflect the amount of effort needed to remove the violations with layout perturbations and, in this special case, the coloring with two DP violations may be easier to fix than that with a single violation depending on the layout (at the same layer as well as the top and bottom layers).

III. Conflict Removal while Minimizing Perturbation

Our layout legalization naturally applies for DP, triple/multiple-patterning (TP/MP), and SADP. We focus on the application of the methodology for DP and show how it can be extended for TP and SADP.

After the DP layout decomposition with the minimum number of conflicts is complete, our objective is to make the layout compatible with DP and resolve the conflicts while minimizing layout perturbation. We use the method proposed in [1] for layout legalization with minimum perturbation as the objective. The layout is represented as a constraint graph where nodes correspond to the layout edges and arcs correspond to the DRs that need to be met between any two layout edges. Arcs are assigned weights that correspond to the values of rules as illustrated in Figure 10. Layer-to-layer connectivity is maintained through the DRs between the layers, which are represented in the graph by arcs between nodes of the different layers.

The two mask layouts of any double-patterned layer are defined as stand-alone layers. Same-color spacing rules, between features of the same mask, including side-to-side, tip-to-side, and tip-to-tip are mapped into arcs between the nodes of the stand-alone mask layer in the constraint graph. DRs that define the interaction between the two mask layouts (e.g., minimum overlap length) are mapped into arcs between the nodes of the two stand-alone mask layers. For the interactions across different layers in the stack (e.g., M1 and contacts), we define any double-patterned layer as the union of its two mask layouts and map across-layers DRs into arcs between nodes of the union layers[5].

As in layout compaction, the two-dimensional minimum perturbation problem is simplified by solving the one-dimensional problem successively (in x and y directions). It is important to note that the order in which the two-dimensional problem is solved, i.e. x or y direction first, can give different results; in our experiments, we solve the problem in both possible orders and choose the best solution. The 1D minimum perturbation problem is formulated as a LP as follows.

$$\begin{aligned} &Minimize && \sum_i W_i |X_i - X_i^{init}| \\ &Subject\ to: && X_j - X_i \geq d_{ij}, \forall A_{ij}, \end{aligned}$$

where X_i and X_i^{init} are the current location and the initial location of node i and W_i is the weight for the perturbation of node i from its initial location. W is normally assigned a value of 1. It can be assigned a larger value to penalize the movement of edges that are less desirable (e.g., edges near the cell boundary) or prevent edges at certain layers from moving (e.g., diffusion/poly layers). A_{ij} is the arc between nodes i and j, which represents the DR constraint between the two layout elements, and d_{ij} is the weight of arc A_{ij}, which represent the value of the DR.

[5]Rather than using layers of the mask layouts and have the same problem highlighted in Figure 2.

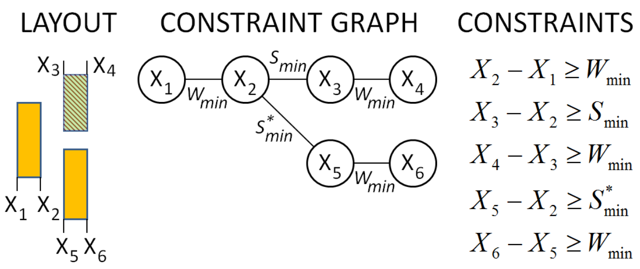

Figure 10. Example of x-direction constraint graph construction and constraint definition for a double-patterned layer. W_{min} is the minimum width rule, S_{min} is the side-to-side different-color spacing rule, and S^*_{min} is the side-to-side same-color spacing rule.

Figure 10 shows the construction of the constraint graph and the definition of constraints in the x-direction for an example double-patterned layout.

In today's technology, design rules can be complex. For example, spacing rules can depend on the edge type (e.g., tip, side, side of a fat wire, etc). To handle such complex rules, we pre-process the layout before the construction of the constraint graph to identify/label the type of each edge; this information is then used to determine what rule is applied when defining the constraint between any two edges. It is worth noting that pattern-based design rules, which may be introduced at future technologies, are expected to be a challenge for our methodology as well as compaction methods in general. Handling of such rules is beyond the scope of this work.

We obtain an equivalent formulation to the original problem with a linear objective function by introducing two new variables L and R for each node i as follows (details in [1]):

$$\begin{aligned} &Minimize && \sum_i W_i (R_i - L_i) \\ &Subject\ to: && X_j - X_i \geq d_{ij} \quad \forall A_{ij} \\ & && L_i \leq X_i, L_i \leq X_i^{init} \quad \forall i \\ & && R_i \geq X_i, R_i \geq X_i^{init} \quad \forall i. \end{aligned}$$

This formulation permits the application of the method for practical layouts that use a discrete manufacturing grid for the coordinates. According to the total unimodularity property [19], when all X_i^{init} and d_{ij} are integers, the solution of the problem consists of integers only. The handling of gridded design rule constraints can be achieved as in [20]. The target on-grid locations are determined and on-grid constraints are relaxed to spacing constraints between the target locations and the cell boundary. After this relaxation, the problem is still formulated and solved as a linear program as detailed in [20].

To handle infeasible constraints, we relax the unsatisfied arc constraints such that all constraints are feasible and a penalty is added in the objective function for the originally infeasible constraint. Section IV gives more details about the handling of infeasible constraints and in-depth details can be found in [1].

Our formulation of the problem maintains all inter and intra layer connectivity, which are represented as constraints in the graph. Internal connectivity of double-patterned layers at stitches is maintained through the minimum overlap length constraint. The conflict-removal problem is solved globally for the entire layout using a single LP with constraints on all layout layers. As a results, our method permits the removal of DP conflicts across all layout layers simultaneously. And,

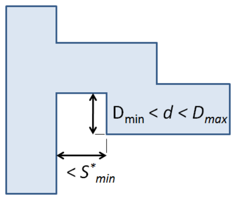

Figure 11. Characteristics of notches that will be removed. S^*_{min} is the side-to-side same-color spacing rule, D_{min} is the notch-depth below which the notch is manufacturable with a single exposure, and D_{max} is the depth beyond which notch-filling is not performed to avoid creating fat wires with larger spacing requirements.

because we formulate the problem as a minimum perturbation problem, our method ensures that the removal of a conflict in one part of the layout will not create a new conflict or design-rule violation in another part of the layout.

The way we formulate the conflict-removal problem permits the application of our methodology for layout legalization for TP and SADP. To apply the methodology for TP, TP coloring is performed instead of DP coloring and the three mask-layouts of any triple-patterned layer are treated as three stand-alone layers. All TP rules that define the interaction between these three mask-layouts (i.e., spacing and overlap rules) are mapped into constraints between the stand-alone layers. And, rules that define the interactions between the triple-patterned layer as a whole and the top/bottom-level layers (e.g., contacts/VIA layers) are mapped into constraints between edges of the union of the three mask-layouts and the edges of the top/bottom-level layers. In a similar fashion, the methodology can be applied for SADP; all that is needed is a SADP-coloring method as [21] and a set of design rules to ensure SADP compatibility of the layout as in [22].

IV. Layout Pre-Processing for More Efficient DP Conflict Removal

In actual layouts, we observe that some DP conflicts can be avoided by simple notch removal prior to coloring. In addition, many conflicts on the M1 layer are caused by segments that are added to cover redundant contacts/vias or to maximize the pin-access region. Redundant contacts and vias improve manufacturability, but they are *not absolutely required*. The same is true for pin segments that extend beyond the minimum requirement to ensure pin-accessibility. The addition of these extra segments is considered a good layout practice to maximize the pin-access region and, consequently, improve the routing efficiency. We take advantage of these observations and, *as an option*, we perform notch filling and allow the *possible* sacrifice of redundancy and extra pin segments to improve the results of the DP conflict removal framework[6]. Specifically, we pre-process the layout prior to the coloring to mark potential sacrificial features. During the legalization and conflict-removal, these features are recovered whenever possible without creating any new violations.

[6]Note if these features are necessary to meet a certain requirement on yield/routability score, it would be possible to mark a fraction of these features for possible sacrifice or simply avoid this optional step altogether.

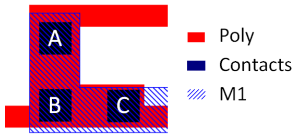

Figure 12. Group of redundant contacts connecting to the Poly layer. Contact B has more flexibility of movement than contacts A and C and, thus, we pick B as the required contact and A and C as redundant contacts that may be sacrificed if necessary to resolve conflicts.

A. Small notch removal

Small notches, or small-depth U-shapes, with depth less than a certain value, D_{min}, may be manufactured with a single exposure. Deeper notches, however, require the two segments of the notch to be assigned to different exposures (i.e. colors). In some layouts, assigning the two segments of a notch different colors is not possible without creating a coloring violation and, in such case, the notch contributes to the number of DP native conflicts. The removal of DP conflicts caused by notches may not be possible during layout legalization when the layout area is fixed and may lead to an area overhead when the layout area is allowed to increase for legalization. As a result, getting rid of notches that cannot be manufactured in a single exposure prior to coloring is a good practice and makes layout legalization for DP more efficient.

An effective way to remove small notches is by joining their two segments so as to fill the notches. Filling a notch requires little layout modifications: it does not create extra color violations and adding extra material (i.e. metal) does not affect other layers. To avoid creating "fat wires" that have peculiar spacing requirements with their neighboring features, we fill notches prior to coloring only when the depth of the notch is smaller than a specified value, D_{max}. Figure 11 depicts the characteristics of notches that will be filled in our DP-enablement framework.

B. Sacrifice of redundant contacts/vias when necessary

The process of identifying redundant vias is similar to the process of identifying redundant contacts and, for brevity, we only describe the latter process. We start by finding overlap regions of the top layer (M1) and the bottom layer (Polysilicon or active). If a single polygon of the overlap region interacts with two or more contacts, these contacts are identified as a group of redundant contacts. Next, we choose one of the contacts from each group to be a required contact/via and add all such required contacts to single contacts to form a new layer of required contacts. The remaining contacts that were not chosen as required contacts are considered redundant. The choice of the required contact among a group is made with preference to the contact with the highest flexibility of movement as illustrated in Figure 12. Contacts that were considered redundant are assigned to a new layer.

If M1 is double patterned, the line-end part of M1 that covers a redundant contact is removed, as shown in Figure 13, and overlapping redundant contacts with M1 is specified as a recommended, but not required, constraint. The LP of the conflict removal method will meet this recommended constraint only when possible without creating a DP conflict or any DR violations. In other words, redundant contacts will be sacrificed only when necessary to resolve conflicts. To ensure

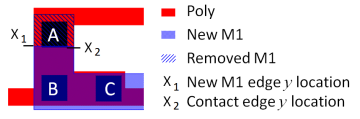

Figure 13. Illustration of M1 simplification for possible sacrifice of contacts.

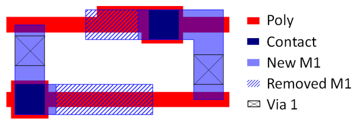

Figure 14. Illustration of M1 simplification for possible sacrifice of pin segments.

recommended contacts still get a chance to be covered by M1 after the layout is perturbed, we add a required constraint to keep redundant contacts at the same spacing and aligned to the corresponding required contact chosen among the group of redundant contacts.

C. Sacrifice of pin segments when necessary

M1 pin segments that do not connect to any other layer in the layout stack are removed for possible sacrifice as shown in Figure 14. To allow the layout perturbation to recover the removed parts when possible without creating violations, the original M1 layer is kept and a recommended constraint is added to the LP problem to minimize the distance between the new M1 edge and the original M1 edge.

The removal of M1 pin segments and M1 parts that cover redundant contacts/vias is performed before the DP coloring. This way, because violations are reduced, extra candidate stitches can be identified and taken advantage of to reduce DP conflicts. When the sacrifice is not necessary to resolve conflicts, these extra stitches will be removed by the coloring algorithm (by coloring violating parts of a stitch with the same color) and the layout perturbation will recover the sacrificed parts as described earlier.

D. Handling recommended constraints during legalization

Recommended constraints are handled in the LP formulation of the conflict removal framework in a similar way as infeasible constraints are handled, i.e. by introducing a new variable to relax the constraint and minimizing this relaxation variable in the objective function. This is illustrated through the example of Figure 13 where M1 covering the redundant contact A is set as a recommended constraint. Here, M1 is shrunk until it slightly overlaps with the redundant contact A. X_1 is the location of the M1 edge overlapping the redundant contact A and X_2 is the location of the bottom edge of the redundant contact (underneath M1). The constraint is then $X_2 - X_1 + r_{12} = \textit{contact width rule} + \textit{M1 overlap past contact rule}$. r_{12} is included in the objective function so that it is minimized. The minimization of relaxation variables for recommended constraints is given less priority than the minimization of the relaxation variables of infeasible required constraints (by assigning a smaller weight in the objective function). This way, recommended constraints are met only when possible without creating any DP conflicts, DR violation, or area increase.

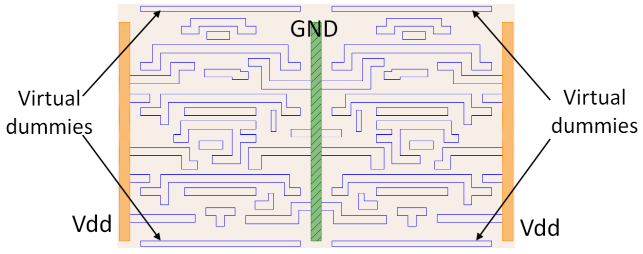

Figure 15. Handling cell-boundary conflicts during placement (virtual dummies not on mask).

It is worth noting that the minimization of the relaxation variables can be weighted according to the importance of what is being sacrificed (e.g., pin-access metric such as in [23]).

V. DP-Compatible Design

Our DP coloring and legalization framework primarily targets standard-cell-based designs. The framework can also be used, however, for small full-custom macro designs as we demonstrate later in the paper in Section VI.

To create a DP-compatible standard-cell-based design, the framework is used to build a standard-cell library. The design is then synthesized using the new library and placement and routing are carried out to create the layout of the entire design. This method preserves the abstraction and common practice in the design of state-of-the-art systems, where standard-cell libraries are commonly developed separately from the physical design and must meet all manufacturing constraints before their release.

With our method, DP conflict-free cells are achieved and they are designed so that no DP conflicts can occur at cell boundaries after placement. Consider the example where M1, which is usually the most complex and dense layer, is double-patterned. The coloring of the Vdd and GND power rails is fixed in all cells as shown in Figure 15. During placement, cells are possibly flipped with respect to the vertical axis so that cells in the same column have Vdd and GND on the same sides (with fixed coloring) and cells of one column and the next/previous columns share the same power rails (as shown in Figure 15). DP conflicts between cells of the same column are prevented as follows. Two dummy M1 wires are added before the coloring process at the cell edges (as in Figure 15) so that all features at the cell sides are assigned the same color (similar to [12]). These dummies are removed after the conflict removal flow is complete and they do not appear on the masks.

Two versions of each cell are provided, one with the initial coloring (after conflict removal) and another with flipped coloring (excluding the power rails coloring) that guarantees no conflict with power rails. If the placement of two cells in the same column results in a DP conflict at the cell boundary, we simply use the version of one of the cells where the coloring is flipped.

Timing differences between the coloring-versions of the same cell may occur due to Critical Dimension (CD) variation of the different exposures and different overlay impacts. Such timing variations are expected to be insignificant, however, since intra-cell M1 wires are naturally short [24]. Consequently, timing analysis needs not be aware of the exact

Table I
RESULTS OF OUR DP COLORING AT THE M1 LAYER WITH TWO SETS OF SAME-COLOR SPACING RULES AND COMPARISON WITH OUR GREEDY AND MINCUT-BASED FLIPPING APPROACHES. THE MINIMUM DIFFERENT-COLOR SPACING RULE IS 65NM, WHILE THE MINIMUM OVERLAP LENGTH IS 10NM.

		S2S = 110, T2S = 120, T2T = 130nm						S2S = T2S = T2T = 130nm					
M1 Layer		**Greedy Flipping**			**MINCUT Flipping**			**Greedy Flipping**			**MINCUT Flipping**		
Design	**Instances**	**Viol**	**Stitches**	**Secs**	**Viol**	**Stitches**	**Secs**	**Viol**	**Stitches**	**Secs**	**Viol**	**Stitches**	**Secs**
OR1200	3,077	1,007	2,152	0.17	995	2,096	7.20	2,119	1,711	0.15	2,119	1,649	0.54
TV80	6,429	3,692	5226	0.48	3,692	5226	0.52	5,667	5,494	0.48	5,667	5,317	9.29
AE18	10,556	9,053	9,597	1.04	9,053	9,597	1.14	14,578	9,376	1.03	14,578	9,376	1.03
MIPS789	19,868	21,273	26,753	2.35	21,273	26,753	3.27	28,582	29,061	2.42	28,578	28,098	405.31

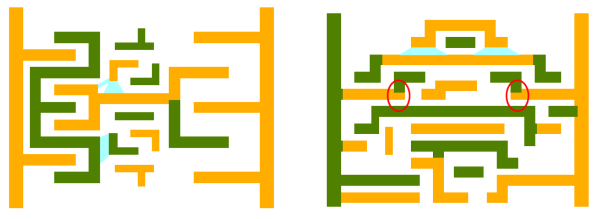
(a) (b)

Figure 16. Violation count based on shape movement requirement: (a) layout example with three spacing violations counted as two conflicts only and (b) layout example with two spacing violations between the same two polygons counted as two conflicts. The highlighted regions correspond to possible issues with newly created tips at stitch locations (this will be discussed in Section VI-E).

coloring of cell-instances. If for some types of designs timing analysis is required to model such effects, the two versions of the cell can be dealt with as completely separate cells and their timing can be characterized separately. This is not desirable as it effectively doubles the number of cells in the library and it should only be used for DP-aware timing analysis with high-accuracy requirement.

VI. EXPERIMENTAL SETUP AND RESULTS

The DP coloring was implemented using Calibre SVRF code [25], for performing projection and forming the final mask layout, and C++ with OpenAccess database/API for the actual coloring of non-violating parts of the layout. An implementation from Boost C++ Libraries [26] was used for identifying connected components and sub-components and the implementation for the Stoer and Wagner MINCUT algorithm from OGDF library [27] was used to find the optimal color-flipping of sub-components. The DP conflict removal with layout perturbation was implemented and integrated into the minimum perturbation based VLSI artwork legalization system [1].

A. *Reporting DP conflicts in the legalization results*

We verify post-coloring DP conflicts (or DP violations) by running design rule check (DRC) on the colored layout using Calibre nmDRC. Same-color spacing violations are effectively DP coloring conflicts. In this case, multiple violations may exist between any two polygons.

The natural way of counting post-coloring conflicts is to count every same-color spacing violation as a conflict. Such count is suitable for evaluating the layout coloring; it is not a good metric, however, for quantifying the effectiveness of the conflict-removal framework. Consider the example of Figure 16(a). If every spacing violation (every aqua-blue polygon in the figure) is counted as a conflict, the layout would have three conflicts. The conflict-removal framework will only need to fix two conflicts, however, since fixing one of the two spacing violations on the top will most likely fix the other automatically (by moving the bottom edge of the top layout polygon up). Yet, not all spacing violations between the same two layout polygons can be treated as a single violation. In the case of Figure 16(b), fixing the two violations will certainly require the movement of two edges (the bottom edges of the top polygon up). Therefore, for better accuracy in reporting the conflict-removal results, we inspect the spacing violations visually to determine how many conflicts they correspond to.

B. *Coloring results*

We test our greedy-based flipping and MINCUT-based flipping coloring methods on M1 layouts with the same runtime environment (2GHz CPU and 18GB RAM). The layouts correspond to designs from [28] with number of cell-instances ranging from 3K to 20K. We perform this testing with two sets of same-color spacing rules. The first set is with different values for the different rules and the second set is with the same value for all same-color spacing rules[7].

The results, depicted in Table I, show that the greedy-based flipping leads to at most 3.8% larger number of stitches than that achieved with the MINCUT-based flipping, which has the same run-time complexity as [9]. And, for many of the cases, both methods lead to the same number of stitches; those are the cases when no color-flipping is necessary.

Previous works on DP coloring allow a single same-color spacing rule-value. Consequently, even if the manufacturing process permits the first set of rules, the coloring of previous works must be performed with the second set of rules (i.e., all rules are equal to the largest of all rules) to ensure no DP conflicts are ignored during the coloring. The results of Table I show that, in this case, the number of violation increases by 38% on average and by 53% in the worst case. Hence, allowing multiple rule-values to be used in the coloring is crucial when the rules have different values, which is the case in latest technologies.

Table I show that, for the cases where the color-flipping was not necessary, the run-time results of both methods are comparable. When coloring flipping is necessary (cases in Table I where the number of stitches is different for the two methods), however, the run-time of the greedy-based flipping

[7]The rule values are assumed and may be different in an actual process. Yet, whenever the rule values are different and no matter the actual values, allowing multiple same-color spacing rules during the coloring and layout legalization leads to a reduced number of violations (or at least the same) compared to when a single same-color spacing rule is allowed.

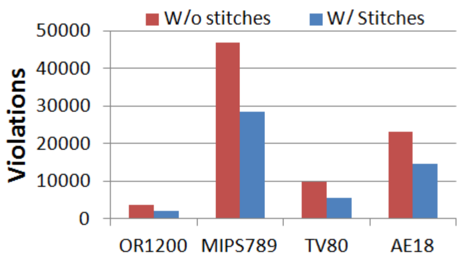

Figure 17. Comparison of the number of same-color spacing violations when stitches are forbidden and when they are allowed.

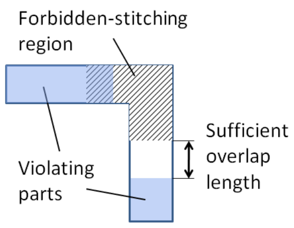

Figure 18. Region of forbidden stitching at corner, which will be merged with the overlapping violating part on the left to prevent stitching at the corner. Note if the overlap length in the non-violating part is smaller than the minimum rule, the non-violating part will also be merged with the violating parts and no candidate stitches will be available in this shape.

is significantly faster than that of the MINCUT-based flipping (up to 167× faster for the largest design).

Since color-flipping does not affect the number of violations in the conflict graph, the number of violations resulting from both coloring methods, i.e. the greedy-based and MINCUT-based flipping, should ideally be the same. In some cases however, the number of violations obtained with both methods may be slightly different (e.g., OR1200 design with the first set of rules and MIPS789 design with the second set of rules in Table I) due to a coloring problem that will be explained in Section VI-E.

To quantify the benefit of stitches in reducing the number of violations, we repeat the DP coloring while forbidding stitches and compare the results to that from Table I (case of S2S = S2T = T2T = 130nm). The results, depicted in Figure 17, show that allowing for stitches reduces the number of violations by roughly 40%. Therefore, forbidding stitching entirely in DP is expected to significantly restrict the layout. It is important to note that stitching may be forbidden at certain locations as in [4] (e.g., at corners). Forbidding candidate stitch locations can be applied easily with our coloring approach by merging regions of forbidden-stitching with overlapping violating parts as illustrated in Figure 18. If the region of forbidden stitching has no overlapping violating part, no measure needs to be taken as stitching will only occur at the interface with a violating part, which is outside the forbidden region in this case.

Different methods for measuring violations can result in significantly different number of violations. For example, corner-to-corner same-color spacing violations may be considered as violations in one method but ignored in another; also, one method may consider same-color spacing violations between vertical segments of any U-shape, but another method may ignore these if one or both vertical segments are shorter than a certain length. Also, since stitches are used to remove DP conflicts/violations, if the number of identified violations is different, the number of stitches will also be different. Hence, to compare the number of violations and number of stitches of our coloring approaches with that of a previous work, the same method must be used to detect violations in layouts colored with our methods and layouts colored with methods from the previous work.

Table II
Comparison of number of violations and stitches in a M2 layout colored with the pure ILP approach of [5] and the same layout colored with our greedy/MINCUT-based approaches.

	ILP [5]		**Our Greedy**		**Our MINCUT**	
	Viol	**Stitches**	**Viol**	**Stitches**	**Viol**	**Stitches**
AES45	1,793	1,848	887	1,779	887	1,764

Five layout test cases from [5] were available to us: four at the Poly layer in uncolored form and one at the M2 layer in colored form. For the colored test case, we use our violation-measuring method to determine the number of violations in the initial coloring, performed using the pure ILP approach of [5], and the coloring performed using our coloring approaches and the same DP rules (same-color spacing and overlap length). Table II reports the number of violations and number of stitches for the different coloring methods. Compared with the layout colored using the ILP-based method of [5], the layouts colored with our coloring approaches contain roughly half the number of violations, while the number of stitches is proportionate[8]. For the uncolored test cases as well as the colored test case, we compare the running times of our coloring approaches with those reported in [5] for the pure ILP approach and the conflict cycle detection (CCD) approach, which artificially removes odd-cycles prior to solving the ILP. The results are shown in Table III. The layouts are at the Polysilicon (Poly) and M2 layers with rules from FreePDK 45nm process [29] for designs with cell-instances ranging from 26K to 300K and synthesized using Nangate 45nm Open Cell Library [30]. The same assumption of minimum same-color spacing (1.2× minimum-spacing relaxation for Poly and 1.1× minimum-spacing relaxation for M2) and overlap length were used for all coloring methods.

Our DP coloring method with greedy-based flipping is the fastest: up to 80× faster than the pure ILP approach, 63× faster than the CCD approach of [5], and 19× faster than our coloring method with MINCUT flipping. Our MINCUT-based flipping method results in up to 9% smaller number of stitches compared with our greedy-based flipping. It is also up to 9× faster than the pure ILP approach and 7× faster than the CCD approach of [5].

C. DP conflict-removal and legalization results

Our DP conflict removal framework was tested on a commercial 22nm standard-cell and macro layouts (on a 2GHz CPU/18GB RAM machine). We assume M1 is double patterned and apply the conflict removal method for layouts that have DP conflicts. The M1 minimum different-color spacing

[8]This result may not be generalizable, however, as it is based on the single layout that was available in colored form. Moreover our violation-measuring method that was used to report the results may differ from that of [5] and, hence, many of the violations may not have been considered while performing the coloring of [5].

Table III
RESULTS OF OUR DP COLORING AT THE POLY AND M2 LAYERS (MINIMUM-SPACING RELAXATION OF 1.2× FOR POLY AND 1.1× FOR M2) AND COMPARISON WITH METHODS FROM PREVIOUS WORK OF [5]. "VIOL" REFERS TO NUMBER OF VIOLATIONS, "SECS" REFERS TO NUMBER OF SECONDS FOR USER TIME, AND "MIN" REFERS TO MINIMUM OVERLAP LENGTH).

				ILP [5]	CCD [5]	Our approach - Greedy		Our approach - MINCUT	
Design	**Instances**	**Min**	**Viol**	**Secs**	**Secs**	**Stitches**	**Secs**	**Stitches**	**Secs**
ART-A 45(70%)	100,000	8	5,976	565	379	29,692	11	27,909	81
ART-B 45(70%)	300,000	10	17,912	2,887	2,317	93,262	36	84755	702
ART-A 45(90%)	100,000	13	5,976	612	391	33,139	12	30,548	57
ART-B 45(90%)	300,000	10	17,912	2,892	2,355	98,053	37	89,734	722
AES45 M2 Layer	26,026	20	887	23.5	5.5	1,779	0.8	1,764	0.7

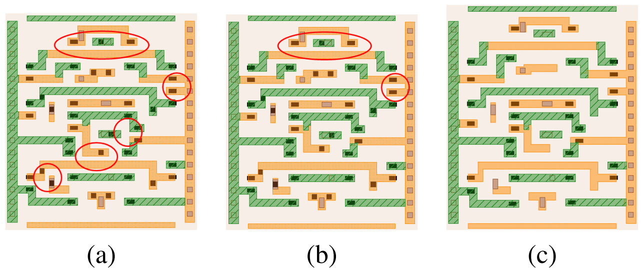
Figure 19. Sample results for a cell layout: (a) before DP conflict removal, (b) after conflict removal with fixed area, and (c) after conflict removal with area increase.

is 40nm and we use a value of 15nm for the minimum overlap length and 80nm for the side-to-side, tip-to-side, and tip-to-tip same-color spacing rules[9].

In one experiment, we apply our DP conflict removal method to standard cells. The results show that DP conflicts in many cells were completely removed without area increase and any DR violations. For some other cells, few DP conflicts remain unresolvable when the area is fixed. We give two options to deal with such stubborn conflicts. The first option is to keep these conflicts and report their locations so that the layout designer fixes them manually. The second option is to run the conflict removal framework a second run with non-fixed area so that all conflicts are removed. Figure 19 shows an example layout where M1 is double-patterned before and after the layout perturbation to remove conflicts. As Figure 19(b) depicts, the conflict removal method with fixed area is able to remove three out of the four conflicts in the original layout of Figure 19(a). The remaining stubborn conflict is removed when Poly and active are allowed to move and area is allowed to increase as shown in Figure 19(c). In this case, the restrictive DR of Poly on grid are met by modifying the LP program as described in [20]. A summary of the results is given in Table IV. For all cells, the runtime for the entire conflict removal flow (coloring plus conflict removal) is less than 10 seconds in real time. In five out of nine cells, all DP conflicts were removed without any area increase or the removal of redundant contacts. In the remaining four cells, few DP conflicts remain after applying our method with fixed area. When we allow the layout area to increase, all conflicts are removed in these cells with an average 6% area overhead (at most 9.1% overhead) and with the sacrifice of a single redundant contacts in just one of the cells.

[9]Because DP was assumed for M1 in the process and to avoid making inadequate assumptions on the differences between the spacing values that we cannot justify, we use the same value for the different rules in the experiments. Nevertheless, the benefits of handling multiple same-color spacing rule values were shown through the DP coloring results.

In another experiment, we apply our DP conflict removal method for two macro layouts, two local clock buffer controllers that consists of multiple latches and inverters with roughly 82 transistors for the first macro and 460 transistors for the second. The results are given in the bottom two rows of Table IV. The method reduces the number of DP conflicts from 13 to 7 for the first macro and from 53 to 31 for the second without increasing the layout area. When the area is allowed to increase, the method removes all remaining conflicts with an average area increase of 8.7% and a total of six sacrificed redundant contacts. The runtime of the entire flow for the largest macro layout is less than one minute in real time (< 2 seconds CPU time).

D. Effects of preferred coloring and sacrifice of redundant CA and pin segments

The use of the preferred coloring method and the possible sacrifice of non-crucial layout features including redundant contacts and pin segments makes the conflict removal more effective. To quantify the impact of these two methods, we run our framework with fixed area and while enabling or disabling the two methods. The number of conflicts results for the different cases are reported in Figure 20 and are compared to the original number of conflicts in the layout before applying the legalization framework and with non-preferred coloring. Up to $\sim$4X smaller number of conflicts can be achieved when the methods are applied for cell layouts and up to $\sim$2X smaller number conflicts can be achieved when the methods are applied for macro layouts. It can also be clearly observed from the results that the conflict removal is effective only when both methods are applied at the same time.

E. A coloring problem solved automatically with legalization

When the tip-to-tip and/or tip-to-side same-color spacing rules are larger than the side-to-side different-color spacing rule plus the minimum width rule, unforeseen post-coloring DP violations may be introduced between newly created tips

Table IV
RESULTS OF APPLYING OUR DP CONFLICT REMOVAL METHOD WITH AND WITHOUT AREA INCREASE TO CELLS FROM A COMMERCIAL 22NM LIBRARY ("N. AREA" STANDS FOR NORMALIZED AREA, "CF" STANDS FOR CONFLICTS, "SRCA" STANDS FOR SACRIFICED REDUNDANT CONTACTS, AND "AREA INC." STANDS FOR AREA INCREASE).

	Original		**Conflict Removal w/o Area Increase**		**Conflict Removal w/ Area Increase**		
Cell/Macro Layouts	**N. Area**	**CF**	**CF**	**SRCA**	**CF**	**SRCA**	**Area Inc.**
LCB + latch 1	1	**1**	**0**	0	-	-	-
latch1	1.6	**3**	**2**	0	0	0	9.1%
oai	1.6	**2**	**0**	1	-	-	-
scan latch	2.3	**5**	**3**	0	**0**	0	6.2%
xor	2.4	**2**	**0**	0	-	-	-
latch2	4.3	**19**	**8**	0	**0**	0	3.3%
nand4	4.7	**4**	**0**	0	-	-	-
latch3	5.3	**4**	**3**	0	**0**	0	5.4%
nand3	6.7	**7**	**0**	0	-	-	-
LCB control. 1 (82 transistors)	13.7	**13**	**7**	4	**0**	4	8.3%
LCB control. 2 (460 transistors)	50.3	**53**	**31**	1	**0**	2	9.1%

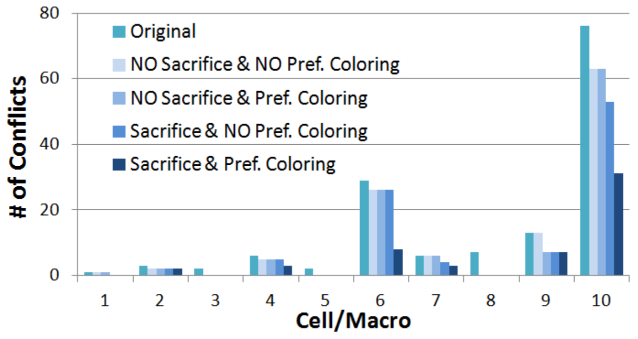

Figure 20. Number of conflicts with the fixed-area flow for the different cell and macro layouts showing the effects of using preferred coloring (see Figure 5) and the possible sacrifice of redundant contacts and M1 pin segments.

at stitch locations and the neighboring shapes as shown by the highlighted regions in Figure 16. This problem may occur with any coloring method that rely on edge-based DRC for checking for color violations and perform projection; it is the reason why we obtained different number of violations with the greedy-based and the MINCUT-based flipping methods in Table I. Our post-coloring legalization approach is advantageous in handling this issue because compaction on already-colored layouts implicitly fixes these newly introduced DP violations.

VII. CONCLUSIONS

We proposed a novel framework to enable DP in the design. Our coloring method guarantees a conflict-free solution for layouts without native conflicts and is performed with a $O(n)$ heuristic algorithm. The automated DP conflict removal and layout legalization are performed simultaneously across all layout layers while minimizing perturbation using a LP. The method enables designing with conventional DRs and masks the designer from the complexity in dealing with DP layers and requirements. The way we formulate the problem allowed us to achieve high-quality results with extremely fast run-time (less than 10 seconds in real time for typical cells). The method targets primarily standard-cell layouts but it can also be applied for small full-custom layouts and interconnect layers in complete designs as we showed in the paper. Although we demonstrate the method on LELE DP, the method is more general and can be naturally extended for other MP technologies including triple/quadruple-patterning in multiple litho-etch steps process as well as SADP. For SADP, all that is needed is a layout-coloring method as [21] and a set of design rules to ensure SADP compatibility of the layout as in [22].

REFERENCES

[1] F. L. Heng, Z. Chen, and G. Tellez, "A VLSI artwork legalization technique based on a new criteria of minimum layout perturbation," in *ACM/IEEE Intl. Symp. on Physical Design*, 1997, pp. 116–121.
[2] 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*, vol. 6521, 2007, pp. 652 109–1.
[3] T.-B. Chiou *et al.*, "Development of layout split algorithms and printability evaluation for double patterning technology," in *Proc. SPIE*, vol. 6924, 2008, pp. 69 243M–1.
[4] A. Tritchkov, P. Glotov, S. Komirenko, E. Sahouria, A. Torres, A. Seoud, and V. Wiaux, "Double-patterning decomposition, design compliance, and verification algorithms at 32nm HP," vol. 7122, pp. p. 71 220S–1, 2008.
[5] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition approaches for double patterning lithography," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 939–952, 2010.
[6] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *Intl. Symp. on Physical Design*, 2009, pp. 107–114.
[7] M. R. Garey and D. S. Johnson, *A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[8] Y. Xu and C. Chu, "GREMA: Graph reduction based efficient mask assignment for double patterning technology," in *Intl. Conf. on Computer-Aided Design*, Nov. 2009, pp. 601–606.
[9] X. Tang and M. Cho, "Optimal layout decomposition for double patterning technology," in *Intl. Conf. on Computer-Aided Design*, Nov 2011, pp. 9 –13.
[10] 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 Asian and South Pacific Design Automation Conference*, 2010, pp. 637–644.
[11] W.-S. Luk and H. Huang, "Fast and lossless graph division method for layout decomposition using SPQR-tree," in *IEEE Intl. Conf. on Computer-Aided Design*, 2010, pp. 112–115.
[12] C.-H. Hsu, Y.-W. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 284–294, Feb. 2011.
[13] K. Yuan and D. Z. Pan, "WISDOM: wire spreading enhanced decomposition of masks in double patterning lithography," in *IEEE Intl. Conf. on Computer-Aided Design*, 2009, pp. 32–38.
[14] S.-Y. Fang, S.-Y. Chen, and Y.-W. Chang, "Native-conflict and stitch-aware wire perturbation for double patterning technology," *IEEE Trans.*

on *Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 703–716, may 2012.

[15] R. S. Ghaida, K. B. Agarwal, S. R. Nassif, X. Yuan, L. W. Liebmann, and P. Gupta, "A framework for double patterning-enabled design," in *Intl. Conf. on Computer-Aided Design*, Nov. 2011, pp. 14–20.

[16] J. Zhu, F. Fang, and Q. Tang, "Calligrapher: a new layout-migration engine for hard intellectual property libraries," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1347–1361, 2005.

[17] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[18] M. Stoer and F. Wagner, "A simple min-cut algorithm," *J. ACM*, vol. 44, no. 4, pp. 585–591, Jul 1997.

[19] B. D. Cullity, *Combinatorial optimization: algorithms and complexity*. New York, NY: Prentice-Hall, 1996.

[20] X. Yuan, K. W. McCullen, F.-L. Heng, R. F. Walker, J. Hibbeler, R. J. Allen, and R. R. Narayan, "Technology migration technique for designs with strong ret-driven layout restrictions," in *ACM/IEEE Intl. Symp. on Physical Design*, 2005, pp. 175–182.

[21] Y. Ban, A. Miloslavsky, K. Lucas, S.-H. Choi, C.-H. Park, and D. Z. Pan, "Layout decomposition of self-aligned double patterning for 2d random logic patterning," in *Proc. SPIE*, vol. 7974, 2011, p. 79740L.

[22] Y. Ma, J. Sweis, H. Yoshida, Y. Wang, J. Kye, and H. J. Levinson, "Self-aligned double patterning (sadp) compliant design flow," in *Proc. SPIE*, vol. 8327, 2012, p. 832706.

[23] T. Taghavi *et al.*, "New placement prediction and mitigation techniques for local routing congestion," in *IEEE Intl. Conf. on Computer-Aided Design*, 2010, pp. 621–624.

[24] R. S. Ghaida, G. Torres, and P. Gupta, "Single-mask double-patterning lithography for reduced cost and improved overlay control," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 1, pp. 93–103, Feb 2011.

[25] Calibre standard verification rule format manual v2009. Http://www.mentor.com/.

[26] Boost C++ Libraries. Http://www.boost.org.

[27] OGDF: Open Graph Drawing Framework. Http://www.ogdf.net.

[28] Http://www.opencores.org/.

[29] FreePDK. Http://www.eda.ncsu.edu/wiki/FreePDK.

[30] Nangate open cell library v1.3. 2009. Http://www.si2.org/openeda.si2.org/projects/nangatelib.

Rani S. Ghaida (S'03) is a Principal Engineer at GlobalFoundries in the Design Enablement Division, Milpitas, CA. He earned his PhD in Electrical Engineering from UCLA, his M.S. in Computer Engineering from the University of New Mexico, and his B.S. in Computer Engineering from the Lebanese American University. Rani's research work has been primarily focused on the development of computational techniques and mathematical models for exploring, defining, optimizing, and enabling semiconductor technologies.

Kanak B. Agarwal (S'01–M'05–SM'12) received his MS and PhD degrees in Electrical Engineering from University of Michigan, Ann Arbor in 2003 and 2004, respectively. He is currently working as a researcher at IBM Research Lab in Austin, Texas where he is involved in research and development of next generation computer systems, circuits and tools, and semiconductor technology for IBM Server and Microelectronics business. He has published over 80 papers in refereed conferences and journals and holds over 30 patents. He has served on the technical program committee of several leading VLSI design and Electronic Design Automation conferences and has given invited talks and tutorials at major international forums on process variation and design for manufacturability. He is a Senior Member of IEEE.

Sani R. Nassif (S'80–M'86–SM'02–F'08) received his Bachelors degree from the American University of Beirut in 1980, and his Master's and PhD degrees from Carnegie-Mellon University in 1981 and 1985 respectively. He worked at Bell Laboratories until 1996, then joined the IBM Austin Research Laboratory where he is currently. He has authored numerous conference and journal publications, received five Best Paper awards (IEEE Trans. CAD, ICCAD, DAC, ISQED and ICCD), authored invited papers to ISSCC, IEDM, ISLPED, HOTCHIPS, and CICC, and given Keynote and Plenary presentations at Sasimi, ESSCIRC, BMAS, SISPAD, SEMICON, PATMOS and VLSI-SOC. He is an IEEE Fellow (2008), a member of the IBM Academy of Technology, a member of the ACM and the AAAS, and is an IBM Master Inventor with more than 50 patents.

Xin Yuan Xin Yuan received the B.S. and M.S. degrees in computer science and technology from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles in 2003. She joined IBM EDA in 2003 and has been working in the area of layout migration and layout optimization. Her research interests include VLSI physical design, design migration, and technology mapping for FPGAs.

Lars W. Liebmann Lars Liebmann received BS and MS degrees in Nuclear Engineering and a PhD in Engineering Physics from Rensselaer Polytechnic Institute, Troy, NY. He joined IBM in 1991 where his work on lithography friendly designs for layout intensive Resolution Enhancement Techniques (RET) lead to his pioneering work on restricted design rules (RDR) as a practical means of preserving profitable CMOS scaling. His current technical focus in IBM's Semiconductor Research and Development Center is design-technology co-optimization for sub-resolution patterning of leading-edge technology nodes. Dr. Liebmann holds over 60 patents, has published over 40 technical papers, and has received IBM's Corporate and Outstanding Technical Achievement awards. For his work on lithography friendly design, Dr. Liebmann was appointed Distinguished Engineer of IBM and Fellow of the SPIE - the International Society for Optical Engineering.

Puneet Gupta (S'02–M'08) is currently a faculty member of the Electrical Engineering Department at UCLA (http://nanocad.ee.ucla.edu). He received the B. Tech degree in Electrical Engineering from Indian Institute of Technology, Delhi in 2000 and Ph.D. in 2007 from University of California, San Diego. He co-founded Blaze DFM Inc. (acquired by Tela Inc.) in 2004 and served as its product architect till 2007. He has authored over 100 papers, 15 U.S. patents, and a book chapter. He is a recipient of NSF CAREER award, ACM/SIGDA Outstanding New Faculty Award and SRC Inventor Recognition Award. Dr. Gupta's research has focused on building high-value bridges across application-architecture-implementation-fabrication interfaces for lowered cost and power, increased yield and improved predictability of integrated circuits and systems.