Layout Decomposition Approaches for Double Patterning Lithography

Andrew B. Kahng, *Fellow, IEEE,* Chul-Hong Park, *Member, IEEE,* Xu Xu, *Student Member, IEEE,* and Hailong Yao, *Member, IEEE,*

***Abstract*—In double patterning lithography (DPL) layout decomposition for 45 nm and below process nodes, two features must be assigned opposite colors (corresponding to different exposures) if their spacing is less than the *minimum coloring spacing*. However, there exist pattern configurations for which pattern features separated by less than the minimum coloring spacing cannot be assigned different colors. In such cases, DPL requires that a layout feature be split into two parts. We address this problem using two layout decomposition approaches based on a *conflict graph*. First, node splitting is performed at all feasible dividing points. Then, one approach detects conflict cycles in the graph which are unresolvable for DPL coloring, and determines the coloring solution for the remaining nodes using integer linear programming (ILP). The other approach, based on a different ILP problem formulation, deletes some edges in the graph to make it two-colorable, then finds the coloring solution in the new graph. We evaluate our methods on both real and artificial 45 nm testcases. Experimental results show that our proposed layout decomposition approaches effectively decompose given layouts to satisfy the key goals of minimized line-ends and maximized overlap margin. There are no design rule violations in the final decomposed layout.**

***Index Terms*—Double patterning lithography (DPL), integer linear programming (ILP), layout decomposition, node splitting.**

I. Introduction

As MOORE'S LAW continues to drive performance and integration with smaller circuit features, lithography is pushed to new extremes. For 32 nm node patterning, though immersion ArF (IArF) is already in use, the realization of extreme ultraviolet (EUV) lithography still faces significant technology obstacles. As a result, double patterning lithography (DPL) technology is attracting more and more attention.[1] An EUV imaging system is composed of mirrors coated with multilayer structures designed to have high reflectivity at 13.5 nm wavelength. There are significant technical hurdles to implementation of EUV lithography in terms of mask-blank fabrication, high output power source, resist material, etc. Challenges to production use of IArF include very high-refractive index fluids (to enable NA = 1.55–1.6), and accompanying advances in high-index resists and optical materials.

DPL involves the partitioning of dense circuit patterns into two separate exposures, whereby decreased pattern density in each exposure improves resolution and depth of focus. DPL is likely to play an even more important role than previously anticipated, since the EUV adoption timeline has been delayed [8], [15]. However, DPL increases manufacturing cost in two fundamental ways: 1) reduced fab throughput by complex process flows due to double exposure patterning, and 2) tight overlay control between the two patterning exposures.

There are distinct approaches to DPL, notably double patterning (DP), double exposure (DE) and spacer double patterning (SDP) [6]. In the DP approach [15], [29], the first etch step transfers the pattern of the first resist layer into an underlying hardmask [16], [20] which is not removed during the second exposure. Photoresist is re-coated on the surface of the first process for a second exposure. The second mask, having patterns separated from the first mask, is exposed and then the flow finishes up with the hardmask and resist of second exposure. Unlike the DP approach with two separate lithography/etch steps, the DE approach incorporates two lithographic exposures with only one etch step, where the image formed by the first exposure may interact with the image formed by the second exposure [7]. In the SDP approach [18], [21], the patterns for the first layer are transferred into the hardmask and then nitride spacers are formed on the sidewalls of the patterns. A spacer is formed by deposition or reaction of the film on the pattern, followed by etching to remove all the film material except for the material on the sidewalls. Then, film material between spacers produces the patterns for the second layer [19], [22]. In this paper, we focus on DP/DE

Manuscript received June 27, 2009; revised December 8, 2009. Date of current version May 21, 2010. Research at University of California at San Diego (UCSD), La Jolla, was supported in part by the Semiconductor Technology Academic Research Center. Preliminary versions of this paper appeared in [1] and [2]. Extensions beyond [1] and [2] include the minimal conflict cycle detection algorithm along with the proof of optimality, a new conflict cycle detection-based layout decomposition flow, the extension of the problem formulation to allow one feature to appear on both masks, and a new definition of the overlap length, along with new experimental results on both scaled and real layouts of different designs. This paper was recommended by Associate Editor C. J. Alpert.

A. B. Kahng is with the Departments of Computer Science and Engineering, and Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, CA 92093-0404 USA (e-mail: abk@ucsd.edu).

C.-H. Park is with Samsung Electronics Company, Seoul 137-857, Korea (e-mail: chul.h.park@samsung.com).

X. Xu is with Synopsys Inc., Mountain View, CA 94043 USA (e-mail: Xu.Xu@synopsys.com).

H. Yao is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: hailongyao@tsinghua.edu.cn).

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

Digital Object Identifier 10.1109/TCAD.2010.2048374

[1]It is our understanding (Chul-Hong Park) that at the leading edge of technology development for logic processes, DPL is currently (November 2009) viewed as necessary at 22 nm and 20 nm, and may be extended to the 15 nm node.

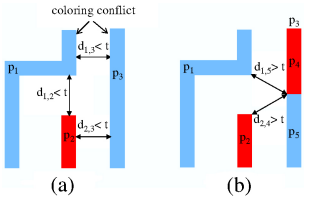


Fig. 1. Example of feature splitting to resolve the coloring conflict with minimum coloring spacing t: (a) Coloring conflict occurs between features p_1 and p_3, and (b) after splitting feature p_3 into p_4 and p_5, each pair of features within minimum coloring spacing t can be assigned opposite colors.

types of DP technology. The major concern of DPL is overlay control, which leads to requirements for more accurate overlay metrology, more representative sampling, reduction in model residuals, and improved overlay correction [13]. Regolli *et al.* [14] analyzed overlay margin in DE patterning. According to the ITRS [6], DPL requires overlay control of between 9 nm and 6 nm, a major hurdle for deployment.

A key issue in DPL from the design point of view is the decomposition of the layout for multiple exposure steps [12]. This recalls strong alternating phase-shift mask (AltPSM) coloring issues and automatic phase conflict detection and resolution methods [9]. DPL layout decomposition must satisfy the following requirement: two features must be assigned opposite *colors* (corresponding to mask exposures) if their spacing is less than the *minimum coloring spacing*. However, there exist pattern configurations for which features within this minimum coloring spacing cannot all be assigned different colors [7], [23]. In such cases, at least one feature must be *split* into two or more parts. Fig. 1 gives an example of using feature splitting to resolve the coloring conflict, where the minimum coloring spacing is t. In (a), because the distances between features p_1, p_2, and p_3 are all less than t, coloring conflict occurs between features p_1 and p_3, i.e., they can not be assigned opposite colors. In (b), after splitting feature p_3 into p_4 and p_5, each pair of features within minimum coloring spacing t can be assigned opposite colors. The feature splitting increases manufacturing cost and complexity due to 1) generation of additional line-ends, which cause yield loss due to overlay error in double-exposure, as well as line-end shortening under defocus, and 2) resulting requirements for tight overlay control, possibly beyond currently envisioned capabilities. Other risks include line edge (critical dimension, or CD) errors due to overlay error, and interference mismatch between different masks. Therefore, a key optimization goal is to reduce the total cost of layout decomposition, considering the above-mentioned aspects.

Yuan *et al.* [27] propose an algorithm to minimize the number of conflicts and stitches simultaneously based on a grid layout model and integer linear programming (ILP) [27]. Their proposed algorithm is based on a grid layout model and cannot process irregular layouts (i.e., with different line widths and line spacings). Moreover, overlay control is not considered in the algorithm, which is necessary for the stitches [12]. Cho *et al.* [28] propose a detailed routing algorithm for double patterning technology to improve layout decomposability and minimize the indecomposable wirelength and the number of stitches. Cork *et al.* [30] investigate the challenges involved in triple patterning of contact layouts. The time complexity of their three-coloring algorithm is $O(3^N)$, which is not practical for full-chip pattern splitting. Though pattern splitting and routing algorithms can solve the DP problem, they both introduce stitches. Lucas *et al.* [31] show that even with the same optical proximity correction (OPC) technology, different splitting points may have different error rates. Therefore, to produce safe stitches, it is important to choose safe splitting points (e.g., with corresponding overlap lengths that satisfy the given overlap margin) rather than risky splitting points on a given layout.

We formulate the optimization of DPL layout decomposition using ILP. A pre-processing step fractures polygonal layout features into rectangles according to vertex coordinates of neighboring features. The fractured rectangular features are further split by a *node splitting* process that resolves coloring conflicts and enlarges the solution space for DPL coloring. We then optimize the coloring of the rectangles with a process-aware cost function that avoids small jogging line-ends, and maximizes overlap at dividing points of polygons. The cost function may also be revised to make preferential splits at landing pads, junctions and long runs [12]. A layout partitioning heuristic helps achieve scalability for large layouts.

We present two different layout decomposition approaches. The first approach performs conflict cycle detection (CCD) to find and report coloring conflicts (i.e., design changes such as layout modifications are needed) then finds the coloring solution on the remaining features using an ILP formulation. The second approach [referred to as pure ILP (PILP)] directly computes the coloring solution on the rectangles after polygon splitting, along with the minimum number of necessary layout modifications, using a more sophisticated ILP formulation. Both of these methods are used to wholly color each feature, i.e., every feature appears on either one mask or the other. However, to enable enough overlap between the features on the two exposure masks, it may be desirable to assign one feature to both masks [24]. Our DPL coloring methods can be easily extended to allow one feature to have multiple colors (see Appendix). Our contributions are as follows.

1) Our *conflict cycle* detection algorithm efficiently finds patterns with unresolvable color assignment. Because such patterns cannot be manufactured by DPL even despite layout decomposition, this detection step enables fast feedback to the designer to modify the layout pattern.
2) Our node splitting method resolves the coloring conflicts considering the pre-specified overlay margin and enlarges the solution space for DPL coloring, from which the optimal coloring solution is obtained using our ILP formulation.
3) Our ILP-based color assignment algorithms (with both CCD and PILP approaches) optimize a weighted combination of the number of line-ends, the overlap lengths in the decomposed layout, and the number of design rule violations. Our PILP color assignment also minimizes the number of coloring conflicts (i.e., layout modifications are needed).
4) Our layout partitioning technique improves scalability of the ILP-based coloring solution, whose runtime would otherwise increase unmanageably with layout size.

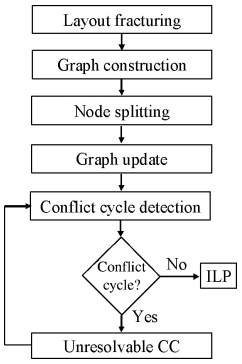


Fig. 2. DPL layout decomposition flow using the CCD approach.

The remainder of this paper is organized as follows. Section II gives the overall flow of our layout decomposition system. Section III formally states the DPL color assignment problem and gives details of the node splitting and color assignment techniques. Section IV describes testcases, experimental setup and experimental results. Section V concludes with ongoing research directions.

II. DPL Layout Decomposition Flow

Before stating our DPL layout decomposition flow, we introduce key terminology. We use *feature* to represent either a layout polygon or a rectangle after the polygons are fractured. Polygons or rectangles can be *split* at the *dividing points*. After DPL color assignment, where there are two touching rectangles with different colors, there is a *cut*, which corresponds to two line-ends. In our conflict graph, each rectangle after polygon fracturing is represented as a node. In most cases rectangles and nodes can be used interchangeably, e.g., *node splitting* means the splitting of one or more rectangles into smaller ones.

A. CCD Approach

Fig. 2 shows the overall flow for DPL layout decomposition using the CCD approach.

1) *Layout fracturing:* Given a layout, the polygonal layout features are first fractured into a set of nonoverlapping rectangles using a minimum-sliver fracturing algorithm [17]. The minimum-sliver fracturing algorithm minimizes the number of slivers (=small rectangles) and helps simplify downstream operations.
2) *Graph construction:* A conflict graph is constructed over the rectangular features according to the given minimum coloring spacing, t (see Section III-B). Each node in the graph represents a rectangular feature. There are two types of edges in the graph: *touching* edges and *conflict* edges. A touching edge exists between two nodes if the corresponding features touch each other, and a conflict edge exists between two nodes if the corresponding features do not touch each other, but are separated by a distance less than t (see Fig. 9 for an example). When a conflict graph is two-colored, a touching edge implies that the corresponding nodes *may* be assigned different colors, but a conflict edge implies that the corresponding nodes *must* be assigned different colors. To make the conflict graph two-colorable, it may be necessary to remove some touching and conflict edges.
3) *Node splitting and graph update:* In the CCD approach, we cast DPL layout decomposition as a problem of modifying the conflict graph by decomposing selected layout feature nodes (thus, adding new nodes and inducing new edges) so that the graph can be properly two-colored. To this end, the key is the removal of conflict cycles (CCs), which are the odd-length—and hence not two-colorable—cycles of conflict edges in the conflict graph.[2] We perform *node splitting* to remove the conflict cycles. To find all possible dividing points on the layout feature nodes, we adopt the concept of *node projection*, details of which are discussed in Section III-C. For each node in the graph, we compute its projections from adjacent nodes connected with conflict edges. Based on the projections, feasible dividing points can be computed. Two important considerations are: 1) not all layout features have *feasible* dividing points, for which the resulting overlap lengths exceed the required overlap margin, and 2) a given layout feature may have several dividing points at which the feature can be split into several smaller features. When the feasible dividing points are computed, the layout features will be split at those dividing points, and the conflict graph will be updated with the newly generated nodes and edges. In fact, the layout fracturing may also be regarded as dividing point selection and node splitting, where the projections are not computed and the required overlap margin is not guaranteed. Note that node splitting is *required* to remove a conflict cycle while generating line-ends with overlap length greater than the required overlap margin. Besides node splitting, the only means of removing a conflict cycle is through costly layout change.
4) *CCD:* Most conflict cycles will be removed by node splitting, such that nodes on the cycle become two-colorable at the cost of more line-ends. However, for some conflict cycles there is no feasible dividing point for any node on the cycle. Such a conflict cycle is detected and reported as an unresolvable conflict cycle (uCC) which must be flagged to the designer for design change. We use a breadth-first search (BFS)-based conflict cycle detection algorithm to find such conflict cycles in the conflict graph.[3]
5) *ILP-based DPL color assignment:* After all conflict cycles have been detected and reported, the remaining graph after removing those conflict cycles becomes two-colorable. We perform ILP-based coloring on the conflict cycle-free graph to find an optimal coloring solution, optimizing the weighted combination of the overlap lengths, the number of cuts and design rule violations. Post-processing functions include analysis

[2]Conflict cycles do not contain touching edges because such coloring conflicts can be removed by flipping the colors of the nodes incident to touching edges.

[3]Depth first search (DFS)-based cycle detection may also be used. The BFS-based conflict cycle detection is more efficient for conflict cycles with fewer edges (e.g., ≤ 10), whereas DFS-based detection is more efficient for conflict cycles with more edges (e.g., > 10). Since most conflict cycles in the layouts we have studied have fewer than seven edges, we adopt BFS-based CCD.

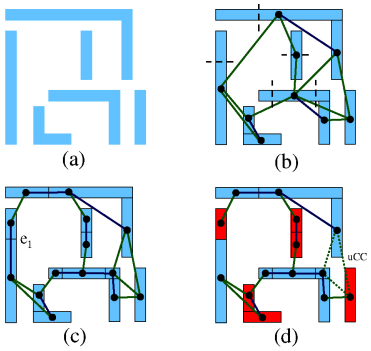


Fig. 3. Example of the conflict graph with conflict edges (green) and touching edges (dark blue), and layout coloring according to the DPL flow using the CCD approach along with an unresolvable conflict cycle (green dashed line). (a) Input layout. (b) Fractured layout and conflict graph with feasible dividing points on layout features (black dashed line). (c) Updated conflict graph after node splitting. (d) ILP-based DPL coloring.

of the overlap lengths for all pairs of touching features (= adjacent split parts of an original layout feature, which have been assigned different mask colors), and design rule checking in the final mask solution.

Fig. 3 illustrates layout decomposition and coloring according to the CCD approach. Polygonal layout features in (a) are fractured into rectangles, over which the conflict graph is constructed with conflict edges (green) and touching edges (dark blue) as shown in (b). To remove the conflict cycles in the conflict graph, node splitting is carried out along the computed dividing points denoted by the dark dashed lines in (b). After node splitting, the conflict graph is updated with newly generated nodes and updated edges shown in (c). Unresolvable conflict cycles which cannot be removed by the node splitting process are reported and marked (green dashed lines) as shown in (d). When unresolvable conflict cycles are detected by CCD, their edges are marked and deleted to make the remaining graph two-colorable. Finally, ILP-based coloring is used to obtain a final coloring solution for the two-colorable conflict graph in (d), where the weighted cost of the overlap lengths, number of line-ends and design rule violations is minimized. Here, touching edge e_1 is deleted as a result of a cut specified in the coloring solution.

B. PILP Approach

Fig. 4 shows our PILP flow for DPL layout decomposition. The flow is similar to CCD except 1) there is no CCD, and 2) a more sophisticated ILP formulation is used to find a coloring solution with minimized coloring conflicts. More precisely, given a layout, the steps of "layout fracturing," "graph construction," "node splitting," and "graph update" are the same as in the CCD approach, and yield an updated conflict graph on the split layout feature nodes. Although most coloring conflicts (conflict cycles in the CCD approach) are resolved during node splitting, the updated graph is not guaranteed to be two-colorable. In other words, some conflict edges may require deletion for a feasible coloring solution to exist. Our ILP-based color assignment is performed on the conflict graph to find the coloring solution that minimizes a weighted sum of

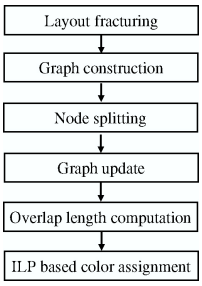


Fig. 4. DPL layout decomposition flow using the PILP approach.

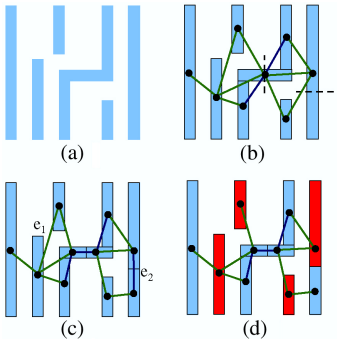


Fig. 5. Example of the conflict graph with conflict edges (green) and touching edges (dark blue), and layout coloring according to our PILP DPL layout decomposition flow. (a) Input layout. (b) Fractured layout and conflict graph before node splitting. (c) Node splitting and graph updating. (d) ILP-based graph coloring.

the number of coloring conflicts,[4] cuts, design rule violations and overlap lengths. Besides the ILP-based coloring algorithm, phase conflict detection [9] and node-deletion bipartization [10] methods could conceivably be adopted with better runtime efficiency but possibly degraded solution quality [2].

Finally, a post-processing phase reports the number of coloring conflicts (i.e., the number of deleted conflict edges), the minimum, average, and standard deviation of the overlap lengths for all pairs of touching features with different colors, and any design rule violations in the final mask solution.

Fig. 5 illustrates the PILP layout decomposition flow. Polygonal layout features in (a) are fractured into rectangles, over which the conflict graph is constructed. In the conflict graph, the conflict edges are green and the touching edges are dark blue. We compute projections for all nodes in the conflict graph, according to which feasible dividing points are computed as indicated by the dashed lines in (b). Then node splitting is carried out at the feasible dividing points and the conflict graph is updated with newly generated nodes and updated edges as in (c). Finally, ILP-based color assignment obtains the final coloring solution given in (d), where conflict edge e_1 and touching edge e_2 in (c) are deleted to make the graph two-colorable.

[4]By increasing the spacing between adjacent features to be greater than the minimum coloring spacing, the conflict edge between the corresponding nodes can be deleted. In our implementation, we try to preserve the conflict edges between features of the same cell instance, and preferentially delete conflict edges between features of different cell instances. This is because a spacing perturbation between cell instances is much easier to accomplish than a spacing perturbation within a cell instance.

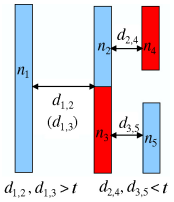


Fig. 6. Example of color assignment problem: feature n_2 (resp. n_3) is assigned a different color from n_4 (resp. n_5) because $d_{2,4} < t$ (resp. $d_{3,5} < t$).

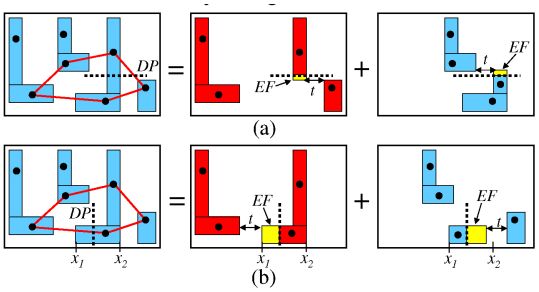


Fig. 7. Two examples of dividing points. (a) Extended features at the dividing point cause coloring violation. (b) Extended features at the dividing point still satisfy the minimum coloring spacing rule.

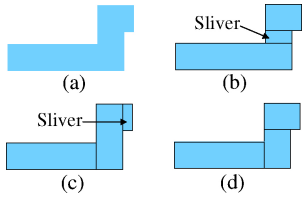


Fig. 8. Example of minimum-sliver fracturing. (a) Polygonal layout feature. (b) Horizontal fracturing. (c) Vertical fracturing. (d) Minimum-sliver fracturing.

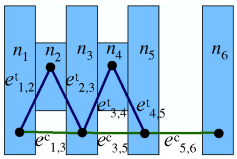


Fig. 9. Example of conflict graph construction: every (rectangular) feature is represented by a node, each pair of touching features are connected by a touching edge, each pair of nontouching features within minimum coloring spacing t are connected by a conflict edge, and no touching feature entirely blocks two nontouching features connected by conflict edges.

III. DPL Color Assignment Problem

A. Problem Formulation

Fracturing and DPL Color Assignment Problem

Given: Layout L, and maximum distance between two features (i.e., polygons), t, at which the color assignment is constrained.

Find: A fracturing of L and a color assignment of fractured features to minimize total cost.

Subject to: 1) Two nontouching fractured features corresponding to nodes n_i and n_j with $0 < d_{i,j} \leq t$ must be assigned different colors, and 2) two touching features with $d_{i,j} = 0$, if assigned different colors, incur a cost $c_{i,j}$.

Fig. 6 illustrates the color assignment problem. Feature n_2 (resp. n_3) is assigned a different color from n_4 (resp. n_5), because $d_{2,4} < t$ (resp. $d_{3,5} < t$). Since $d_{1,2} > t$ and $d_{1,3} > t$, there is no need for the pairs of features n_1 and n_2, and n_1 and n_3, to be assigned different colors. Note that when two touching fractured features, e.g., n_2 and n_3 in the figure, are assigned different colors, the two features raise the manufacturing cost (i.e., risk) due to overlay error. We should maximize the overlap between the respective mask layouts of n_2 and n_3 in this case, as we now discuss.

When a feature is split into two parts, the two line-ends at a dividing point (DP) must be sufficiently overlapped. In Fig. 7, the extended features (EF) that address the overlap requirement must still satisfy DPL design rules: the spacing between features at the dividing point must be greater than the minimum coloring spacing. Fig. 7 shows how two dividing points lead to different minimum spacings after layout decomposition. In the figure, each layout decomposition is two-colorable. However, when extending features for overlay margin, the dividing point in Fig. 7(a) causes violation of the minimum coloring spacing, t. The dividing point in Fig. 7(b) maintains the minimum coloring spacing even after line-end extension for overlay margin.

B. Fracturing and Conflict Graph Construction

We use a minimum-sliver fracturing algorithm [17] to fracture the layout polygons into a set of nonoverlapping rectangles so that distance computation and other feature operations (e.g., feature splitting) become easier. The minimum-sliver fracturing algorithm minimizes the number of small rectangles and thereby helps avoid minimum design rule violations. Fig. 8 shows an example of minimum-sliver fracturing. Polygonal layout feature in (a) is fractured in horizontal direction in (b), in vertical direction in (c), and using minimum-sliver fracturing in (d). Both horizontal and vertical fracturing methods generate a sliver as shown in the figure. Such a sliver, when its adjacent touching features are assigned different colors, may violate a minimum design rule even after line-end extension for overlay margin. Though ILP-based coloring (Section III-G) helps avoid such minimum design rule violations, it would be helpful to avoid as many slivers as possible at the beginning.

Our layout decomposition begins with construction of a conflict graph based on the fractured layout. As illustrated in Fig. 9, given a (post-fracturing) rectangular layout L_R, the conflict graph $G = (V, E^C \cup E^T)$ is constructed by: 1) representing each feature (i.e., rectangle) by a node n; 2) for any two nontouching features within distance t, connecting the two corresponding nodes with a *conflict edge* e^c; and 3) for any two touching features, connecting the two corresponding nodes with a *touching edge* e^t. A further condition for 2) is that two nontouching features connected by a conflict edge in the graph must either belong to different polygonal layout features, or belong to the same polygonal layout feature with no other touching feature between them (i.e., no touching features entirely block the two nontouching features). In Fig. 9, $E^C = \{e^c_{1,3}, e^c_{3,5}, e^c_{5,6}\}$ are conflict edges and $E^T = \{e^t_{1,2}, e^t_{2,3}, e^t_{3,4}, e^t_{4,5}\}$ are touching edges. There is no conflict edge between n_2 and n_4 since node n_3 blocks these two nodes.

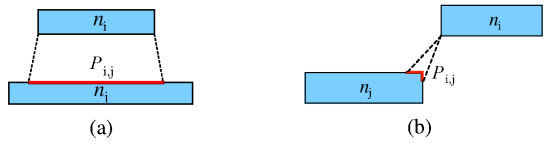


Fig. 10. Node projection examples.

C. Node Splitting

Now we describe our node splitting technique used in both the CCD and PILP approaches.

Definition 1: The *node projection* $P_{i,j}$ from node n_i to node n_j is a set of points on n_j that have distance to node n_i less than minimum coloring spacing t.

Fig. 10 shows two examples of node projections from node n_i to n_j. Node projections can have complicated shapes including circular sectors, rectangles, etc.; the figure shows only the projections on the surface of node n_j. In our implementation, the projections are approximated as rectangles to reduce the computational complexity.

Fact 1: In the conflict graph, node projections between each pair of nodes connected with a conflict edge are nonempty.

Definition 2: Node projections are *separable* in horizontal (vertical) direction if there exists at least one horizontal (vertical) splitting line that separates the projections into two parts without cutting on any projections.

Definition 3: The *overlap length* between two touching nodes is the maximum length that the two nodes can be overlapped by extending them in opposite directions toward each other without introducing new edges in the conflict graph.

In the conflict graph, wherever there is a pair of touching nodes, there is a dividing point, i.e., the touching nodes were generated by splitting the original feature at the dividing point. *Rule-based node splitting:* Given a node $n_i \in G$, if 1) the projections on n_i are separable in horizontal (resp. vertical) direction, 2) the resulting overlap length of the horizontal (resp. vertical) splitting is not less than the given overlap margin, and 3) there are no design rule violations after splitting, then node n_i can be horizontally (resp. vertically) split into two nodes. The dividing point (i.e., splitting line) may be chosen anywhere in between the projections such that no projections are cut, and no violations of the required overlap margin or design rules occur. According to Definition 3, the overlap length is not affected by the position of the dividing point in the overlap area. In this paper, we prefer to set the dividing point at the center of the overlap area such that each touching node can be extended by half of the overlap length. In design rule checking, we consider the extended layout features on each exposure mask rather than the features immediately after node splitting. While the size of a node after splitting may violate a minimum design rule, the node size after extension for overlay margin may be acceptable. During node splitting and design rule checking, we compute the sizes of extended nodes based on the projections and overlap length, which is more accurate and appropriate with DP technique.

Fig. 11 shows an example of rule-based node splitting. In the figure, node projections $P_{j,i}$, $P_{l,i}$, and $P_{k,i}$ on node n_i corresponding to nodes n_j, n_l, and n_k are separable with the overlap length not less than the given overlap margin. Hence,

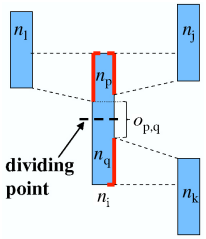


Fig. 11. Example of rule-based node splitting: $o_{p,q}$ is the overlap length and the dividing point is at the center of the overlap area.

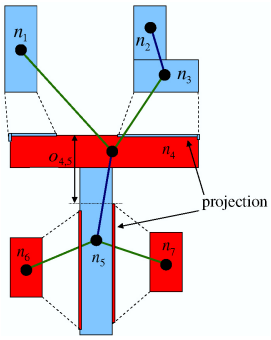


Fig. 12. Example of overlap length calculation: $o_{4,5}$ is the overlap length between nodes n_4 and n_5.

node n_i can be split into two new nodes n_p and n_q at the dividing point, with the corresponding overlap length $o_{p,q}$. The dividing point is set at the center of the overlap area between the lower point of $P_{l,i}$ and the upper point of $P_{k,i}$. A more detailed illustration of overlap length is given in Fig. 12, where the two touching features n_4 and n_5 are assigned different colors, and thus the overlap between n_4 and n_5 is required to be greater than the given overlap margin to guarantee successful manufacturing. The overlap length of the touching features n_4 and n_5 is denoted as $o_{4,5}$. When computing overlap length, two features of the same color cannot be extended such that the distance between them is less than the minimum coloring spacing t (e.g., in the figure, n_4 cannot be extended downward to touch the projection of feature n_7).

Our node splitting is applied to all the nodes with feasible dividing points in the conflict graph, so that we may eventually obtain a graph which becomes two-colorable after removing the minimum number of conflict edges or reporting the minimum number coloring conflicts (i.e., the conflict cycles). We perform the rule-based node splitting for each node with feasible dividing points. To compute the feasible dividing points, *projections* are calculated for each node from its adjacent nodes connected with conflict edges in the conflict graph. According to the projections of a given node, the overlap length for each possible dividing point is computed. For each dividing point with achievable overlap length greater than the required overlap margin, node splitting is used to split the node into two nodes at the dividing point. After node splitting at all feasible dividing points, the conflict graph is updated.

D. CCD

After node splitting, there may still be coloring conflicts in the updated conflict graph due to some special layout pattern configurations. To detect such coloring conflicts, in our CCD approach we find odd-length cycles of conflict edges in the

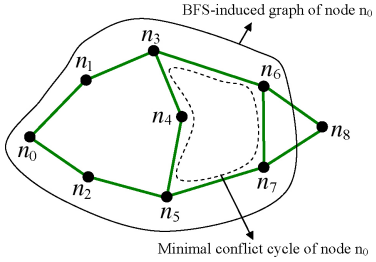


Fig. 13. Example of minimal conflict cycle of node n_0.

conflict graph. In the following part of this subsection, unless otherwise specified, all cycles and paths are for conflict edges rather than touching edges.

Definition 4: Given a conflict graph G, a *conflict cycle* is an odd-length (odd number of conflict edges) cycle in G.

Definition 5: Given a conflict cycle C in conflict graph G, assume for any two nodes $u, v \in C$, path $\mathcal{P}_{u,v}$ in C is of odd length and path $\mathcal{P}'_{u,v}$ in C is of even length, then C is a *minimal conflict cycle* if there is no odd-length path in G between u and v shorter than $\mathcal{P}_{u,v}$ and no even-length path between u and v in G shorter than $\mathcal{P}'_{u,v}$.

Definition 6: Given a conflict graph G, a *BFS-induced graph of node n* is a subgraph of G, which has been visited by the BFS search from node n when the first conflict cycle is detected.

Definition 7: A *minimal conflict cycle of node n* is a minimal conflict cycle in the BFS-induced graph of node n.

Details of our BFS-based CCD algorithm along with proof of optimality is given in the Appendix. Fig. 13 shows an example. Starting from node n_0, BFS search is performed. When reaching the first conflict cycle $\{n_0, n_2, n_5, n_7, n_6, n_3, n_1\}$, the BFS-induced graph of node n_0 is $\{n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7\}$, and the minimal conflict cycle of n_0 in the BFS-induced graph is $\{n_3, n_4, n_5, n_7, n_6\}$. From the example, from different starting nodes, different minimal conflict cycles can be found, e.g., starting from node n_8, the minimal conflict cycle of n_8 will be found as $\{n_6, n_7, n_8\}$.

When a conflict cycle is detected, we must report it for removal by design change (e.g., by increasing the spacing between feature nodes in the cycle, so as to remove one or more conflict edges) since node splitting has already been performed. CCD and reporting is carried out in an iterative manner: after each conflict cycle is detected, conflict cycle reporting is invoked to mark the conflict cycle and remove it from the conflict graph by deleting all the conflict edges in the cycle. Further rounds of detection and reporting are performed until no conflict cycles exist in the graph, i.e., the graph is two-colorable.

The runtime of CCD and reporting depends on the number of conflict cycles and the density of the conflict graph. As described in the experimental results in Table III, total runtime for the whole process, including the CCD and reporting, and min-cost color assignment, is reasonable: around 2.2 h for layouts of 45 nm design ART-C45 with more than 3.2M polygons (more than 10M rectangles after fracturing).[5]

[5]This runtime includes all stages: layout partitioning, all rounds of conflict cycle detection and reporting, ILP-based color assignment, etc. The total runtime of BFS-based CCD across all CCD rounds is less than 13 s for t = 91 nm.

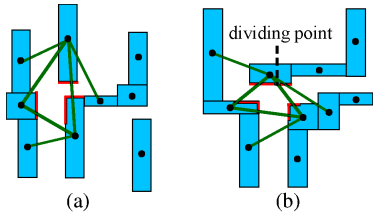


Fig. 14. Example of unresolvable conflict cycle (uCC). (a) uCC with no dividing point. (b) uCC with nonzero overlap length that is less than the required overlap margin.

E. Unresolvable Conflict Cycle (uCC)

We now discuss why not all conflict cycles can be eliminated by the node splitting method. DPL layout decomposition fails when layout features within the coloring spacing lower bound cannot be assigned different colors. Such a failure, which we call an uCC, consists of two cases: 1) among all the feature nodes there is no dividing point with nonzero overlap length to remove the conflict cycle, and 2) even if there is a possible dividing point to remove the conflict cycle, the overlap length is less than the required overlap margin. Fig. 14 illustrates these two types of uCCs, where conflict edges are drawn between the features. In Fig. 14(a), the projections are not separable on any of the three nodes in the conflict cycle denoted by bold conflict edges. Removal of the conflict cycle may be achieved by design change, e.g., by increasing the spacing between two neighboring features to be $> t$. In Fig. 14(b), the overlap length at the dividing point is less than the required overlap margin. Again, a fix by design change is needed.

After CCD and reporting, we apply ILP-based coloring (Section III-G1) to decide which nodes to split in the final decomposition result. The ILP optimizes a weighted objective of number of cuts, design rule violations, and overlap lengths.

In our PILP approach, the conflict graph may not be two-colorable after the node splitting. We apply a more sophisticated ILP-based coloring method (Section III-G2) to compute a minimized number of conflict edges for deletion. The deleted conflict edges will be reported and marked for design changes. To reduce the cost of the design change, we preferentially delete conflict edges between features of different cell instances, so that the desired design changes may be obtained by shifting the cell instances. Besides minimizing the number of coloring conflicts, the ILP also determines which nodes to split (i.e., the corresponding pair of touching nodes are assigned different colors) in the final decomposed layout to optimize a weighted objective of number of cuts, design rule violations, and overlap lengths.

F. Layout Partitioning

In most placements, the conflict graph between cells is sparse: due to the required poly-to-cell boundary distance, as well as whitespace between cells, there are not many edges between the cells. As a result, many "*islands*" (connected components) can be found in the conflict graph. At the same time, runtimes of our ILP-based coloring algorithms increase dramatically when the number of nodes and edges in the graph is large. Therefore, we merge the nodes into small

Algorithm 1 Layout Partitioning Algorithm

Input: Conflict graph G and the mapping information from nodes to polygons.
Output: A set of clusters of nodes such that no edge or node pair from any single layout polygon exists in between any pair of clusters.
1: Make a new graph G' on polygons with each node n' representing a polygon;
2: **for all** $e = (u, v) \in G$ **do**
3: Set edge $e' \leftarrow (u', v') \in G'$, where u' and v' correspond to the polygons in which rectangles of u and v are, respectively, located;
4: **while** there is an unvisited node $n'_i \in G'$ **do**
5: Make a new cluster c'_i containing n'_i;
6: Perform breadth-first search from n'_i and add all visited nodes into c'_i;
7: **for all** clusters c'_i **do**
8: Make cluster c_i on the nodes of the polygons in c'_i;

clusters according to the connectivity information, with no edges or nodes of a given polygon occurring in multiple clusters. Each cluster has its separate conflict graph, and ILP-based coloring may be performed on each cluster in sequence. Because there are no edges between clusters, and no polygon has nodes in more than one cluster, the final solution is simply the union of solutions for all the clusters, with no degradation of solution quality.

Our layout partitioning algorithm is given in Algorithm 1. The time complexity of Steps 2–4 is $O(E)$, the complexity of Steps 5–8 is $O(V' + E')$, and the complexity of Steps 9–11 is $O(C' + V)$, where V is the total number of nodes, E is the total number of edges between nodes, V' is the total number of polygons, E' is the total number of edges between polygons, and C' is the total number of clusters on polygonal layout features. From the above analysis, our layout partitioning algorithm runs in linear time in the size of the conflict graph. For dense graphs which can be induced by a large minimum coloring spacing t, cluster sizes after layout partitioning may still be large. Graph partitioning methods [25] may be adopted to partition large clusters into smaller ones to further reduce the runtime of ILP-based coloring, but this is at the cost of introducing suboptimality into the coloring solutions. In our implementation, we preferentially delete touching edges to partition the clusters to avoid incurring unnecessary design changes. Furthermore, to avoid violating the required overlap margin, only when the overlap length between the corresponding touching nodes is greater than the overlap margin can a touching edge be deleted.

G. Min-Cost Color Assignment Problem Formulation

1) *CCD Approach:* Min-cost color assignment is formulated as the following ILP:

Objective: Minimize $\sum c_{i,j} \times y_{i,j}$

Subject to

$$x_i + x_j = 1 \qquad \forall\, e^c_{i,j} \tag{1}$$

$$x_i - x_j \le y_{i,j} \qquad \forall\, e^t_{i,j} \tag{2}$$

$$x_j - x_i \le y_{i,j} \qquad \forall\, e^t_{i,j} \tag{3}$$

where x_i and x_j are binary variables (0/1) for the colors of rectangles r_i and r_j, and $y_{i,j}$ is a binary variable for touching edge $e^t_{i,j} \in E^T$, i.e., for any pair of touching rectangles r_i and r_j. Constraint (1) specifies that nontouching rectangles r_i and r_j within distance t should be assigned different colors. Constraints (2) and (3) are used for evaluating the cost when

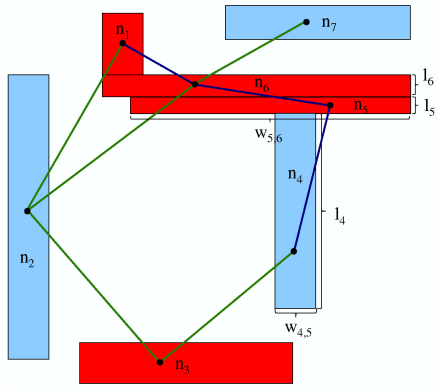


Fig. 15. Example of cost function: $l_5 < l_6 < FS_{\min}$, $c_{4,5} = \alpha/l_5 + \beta + \gamma/o_{4,5}$, $c_{5,6} = \alpha \cdot FS_{\min}/(l_5 \cdot l_6) + \beta + \gamma/o_{5,6}$, $o_{4,5} > o_{5,6}$, $c_{5,6} > c_{4,5}$.

touching rectangles r_i and r_j are assigned different colors. The cost for touching rectangles is defined as

$$c_{i,j} = \alpha \cdot f(w_{i,j})/(f(l_i) \cdot f(l_j)) + \beta + \gamma/o_{i,j} \tag{4}$$

where $w_{i,j}$ is the width of the rectangle edge between rectangles r_i and r_j, l_i and l_j are the lengths of the rectangle edges of r_i and r_j which are opposite to the touching edge between r_i and r_j,[6] $o_{i,j}$ $(= o_{j,i})$ is the overlap length between nodes n_i and n_j, and α, β and γ are user-defined parameters to weight the different optimization objectives. Function f is defined as

$$f(x) = \begin{cases} FS_{\min} & \forall x \ge FS_{\min} \\ x & \forall x < FS_{\min} \end{cases} \tag{5}$$

where $FS_{\min}$ is minimum feature size, i.e., the threshold on the features, below which a design rule violation will occur. Our ILP problem formulation seeks to minimize the weighted cost over the design rule violations, the number of cuts on the layout polygons, and the overlap lengths between touching features of different colors.

a) *Minimizing Design Rule Violations:* During layout fracturing, small rectangles may be generated due to certain polygonal layout features (e.g., n_5 and n_6 in Fig. 15). According to (4), higher costs will be assigned to smaller-size pairs of touching rectangles. Thus, the ILP aims to minimize design rule violations in the final layout.

b) *Minimizing the Number of Cuts:* The cuts on the layout features will introduce more line-ends, which is undesirable due to line-end shortening effects and overlay errors. In (4), setting the second term (β) to be greater than the first term, gives cut minimization higher priority relative to design rule violations.

c) *Maximizing the Overlap Length:* Maximum overlap length satisfying the required overlap margin is also desirable. In (4), a larger γ value corresponds to more emphasis on overlap length maximization.

Fig. 15 illustrates the cost function computation, where 1) $l_5 < l_6 < FS_{\min}$; 2) $c_{4,5} = \alpha \cdot f(w_{4,5})/(f(l_4) \cdot f(l_5)) + \beta + \gamma/o_{4,5} = \alpha/l_5 + \beta + \gamma/o_{4,5}$; 3) $c_{5,6} = \alpha \cdot f(w_{5,6})/(f(l_5) \cdot f(l_6)) + \beta + \gamma/o_{5,6}$

[6] As mentioned in Section III-C, touching line ends are extended across each other to ensure the required overlap margin. So, the sizes of touching features are computed after their extension based on the precomputed projections.

$= \alpha \cdot FS_{\min}/(l_5 \cdot l_6) + \beta + \gamma/o_{5,6}$; 4) $o_{4,5} > o_{5,6}$;[7] and 5) $c_{5,6} > c_{4,5}$. From this computation, we get $c_{5,6} > c_{4,5}$. Given such costs on the touching rectangles, the ILP solver can output the color assignment results as illustrated in Fig. 15. Since the length of rectangle n_5, l_5, is less than the minimum feature size $FS_{\min}$ (even after the extension for overlap), the design rule will be violated if n_5 is assigned a different color than n_6. In this way, the cost function supports the minimization of design rule violations.

2) *PILP Approach:* Min-cost color assignment is formulated using a different ILP. In addition to the variables x_i and $y_{i,j}$, we introduce a new binary variable $z_{i,j}$ for conflict edge $e^c_{i,j} \in E^C$. $z_{i,j} = 0$ when $x_i \neq x_j$, while $z_{i,j} = 1$ when $x_i = x_j$. We then have:

Objective: Minimize $\sum y_{i,j} \times (\beta + \gamma/o_{i,j}) + \sum \lambda \times z_{i,j}$

Subject to

$$\begin{cases} 2x_i - x_j - x_k \leq 1 \\ x_j + x_k - 2x_i \leq 1 \end{cases} \quad \forall\ e^t_{i,j}, e^t_{i,k} \text{ and } l_i < FS_{\min} \tag{6}$$

$$x_i = x_j \quad \forall\ e^t_{i,j} \text{ and } l_i < FS_{\min} \tag{7}$$

$$x_i = x_j \quad \forall\ e^t_{i,j} \text{ and } o_{i,j} < OM \tag{8}$$

$$\begin{cases} x_i - x_j \leq y_{i,j} \\ x_j - x_i \leq y_{i,j} \end{cases} \quad \forall\ e^t_{i,j} \text{ and } o_{i,j} \geq OM \tag{9}$$

$$\begin{cases} x_i + x_j - 1 \leq z_{i,j} \\ 1 - x_i - x_j \leq z_{i,j} \end{cases} \quad \forall\ e^c_{i,j} \tag{10}$$

where l_i, $FS_{\min}$, $o_{i,j}$ are defined as above, and OM is the required overlap margin.

Constraints (6) and (7) avoid design rule violations. In Constraint (6), if the size of rectangle r_i is less than the minimum feature size and r_i touches rectangles r_j and r_k on two sides, then r_i should be assigned the same color as one or both of r_j and r_k. In Constraint (7), if the size of rectangle r_i is less than the minimum feature size and r_i only touches rectangle r_j, then r_i should be assigned the same color as r_j to avoid a design rule violation.

Constraint (8) enforces the required overlap margin: if the overlap length between touching rectangles r_i and r_j is less than the required overlap margin, then r_i and r_j should be assigned the same color. Constraint (9) is related to the objective of maximizing the final overlap length. When the overlap length $o_{i,j}$ between a pair of touching rectangles r_i and r_j is greater than the required overlap margin OM, a cost $\gamma/o_{i,j}$ is incurred when r_i and r_j are assigned different colors ($x_i \neq x_j$), i.e., $y_{i,j} = 1$.

Constraint (10) is related to the objective of minimizing the number of coloring conflicts, i.e., the conflict edge removal process which removes the conflict edges between rectangles by changing the design. A cost λ is incurred for a conflict edge $e^c_{i,j}$ if r_i and r_j are assigned the same color ($x_i = x_j$), i.e., $z_{i,j} = 1$. β, γ and λ are user-specified parameters for balancing the different objectives, e.g., a larger β corresponds to more emphasis on the minimization of the number of line-ends, a larger γ relates to more improvement of overlap length, and a larger λ indicates that the number of coloring conflicts should be minimized. The results reported below are obtained with $\beta = 1$, $\gamma = OM$, and $\lambda = 1e2$.[8]

[7] In Fig. 15, the dividing point between n_4 and n_5 (resp. n_5 and n_6) is obtained by layout fracturing, so it is not at the center of the overlap area between n_4 and n_5 (resp. n_5 and n_6). However, the overlap length between the touching nodes can still be obtained by the computed projections from adjacent nodes, and the overlap length between n_4 and n_5 is larger than that between n_5 and n_6.

TABLE I
TESTCASE PARAMETERS

Design	#Cells	#Polygons	#Rects
AES90	17 304	90 394	362 380
ART-A90	50 288	309 232	1 162 560
ART-B90	100 127	615 703	2 314 740
ART-C90	300 381	1 847 109	6 944 220
ART-D90	500 186	3 075 754	11 563 320
AES45	26 026	65 201	176 131
ART-A45	100 098	641 673	2 187 963
ART-B45	300 026	1 923 301	6 558 031
ART-C45	500 088	3 205 788	10 931 028

Testcases of 90 nm designs are suffixed by "90" and 45 nm designs are suffixed by "45." For 90 nm testcases, minimum spacing (140 nm) and minimum line width (100 nm) on poly layer are scaled by 0.5× to 70 nm and 50 nm, respectively. For 45 nm testcases, minimum spacing and minimum line width on poly layer are 70 nm and 50 nm, respectively.

IV. EXPERIMENTAL RESULTS

We empirically test our CCD and PILP approaches on both real and artificial designs. We evaluate our DPL solutions with respect to: 1) solution quality, 2) scalability, and 3) correctness. Note that our previous work [2] has explored application of alternating-aperture phase-shift mask (AAPSM) conflict resolution methods—specifically, edge deletion bipartization (EDB) and node deletion bipartization (NDB)—to the DPL layout decomposition problem. However, the AAPSM and DPL layout decomposition problems seek different solutions (respectively, feature spacing versus feature splitting), and methods for the former require a planar conflict graph. Reference [2] shows that, due to the suboptimality introduced by the planarization step in EDB, or the heuristic method in NDB, the previous AAPSM methods are inferior (in the DPL pattern splitting context) to the PILP and CCD methods that we present here.

A. Experimental Setup

Our layout decomposition system is implemented in C++. We use one real design (*AES*) implemented using both *Artisan* 90 nm library and Nangate 45 nm open cell library [26] by *Synopsys Design Compiler v2003.06-SP1* [4]. Because real synthesized netlists do not use all of the available standard-cell masters, we also run experiments with artificial designs using the same 90 nm and 45 nm libraries. Our 90 nm artificial designs instantiate more than 600 different types of cell masters, and our 45 nm designs instantiate more than 120 different types of cell masters. The testcases are placed with row utilizations of 70% and 90% using *Cadence SoC Encounter* (v07.10) [5].[9] Table I shows key parameters of the testcases,

[8] To preferentially delete conflict edges between features of different cell instances, the edge deletion cost for those edges is set to $\lambda/2$.

[9] The cell masters in the artificial designs are randomly chosen from the library such that each type of cell master corresponds to approximately the same number of cell instances. Since no netlist is available, the artificial designs are placed into rows in random order, with only the utilization (density) constraint.

TABLE II
EXPERIMENTAL RESULTS OF LAYOUT DECOMPOSITION SYSTEM USING CCD APPROACH ON THE POLY LAYER OF SCALED 90 NM DESIGNS (70% AND 90% UTILIZATION)

Design	t	CEs	DTEs	uCCs	Cuts	Min.	Mean	σ	CPU
AES90	76	85 729	0	0	130	15	131.03	112.94	30.7
(70%)	77	87 241	0	1	128	35	124.36	114.40	30.9
ART-A90	76	363 357	0	0	8578	25	254.60	167.71	196.4
(70%)	77	367 053	0	0	8679	20	255.05	169.65	199.2
ART-B90	76	723 143	0	0	17 079	25	258.08	169.11	548.0
(70%)	77	730 474	0	0	17 291	20	257.86	170.57	589.1
ART-C90	76	2 170 312	0	0	51 214	25	255.97	168.25	4657.3
(70%)	77	2 192 325	0	0	51 864	20	256.26	169.95	4882.4
ART-D90	76	3 612 466	0	0	85 304	25	256.35	168.42	10 509.7
(70%)	77	3 649 073	0	0	86 394	20	256.72	169.97	10 993.1
AES90	76	85 632	0	0	130	15	137.16	114.15	31.1
(90%)	77	87 149	0	1	128	35	130.67	115.92	31.2
ART-A90	76	363 336	0	0	8567	25	255.69	168.58	195.1
(90%)	77	367 047	0	0	8684	20	255.83	170.16	198.2
ART-B90	76	723 619	0	0	17 056	25	255.80	168.14	548.6
(90%)	77	730 978	0	0	17 256	20	255.24	169.60	536.6
ART-C90	76	2 170 098	0	0	51 227	25	256.53	168.18	5483.6
(90%)	77	2 192 132	0	0	51 819	20	256.43	169.77	5403.0
ART-D90	76	3 613 640	0	0	85 296	25	256.15	168.41	14 921.1
(90%)	77	3 650 513	0	0	86 388	20	256.52	170.01	11 055.8

TABLE III
EXPERIMENTAL RESULTS OF LAYOUT DECOMPOSITION SYSTEM USING CCD APPROACH ON THE POLY LAYER OF 45 NM DESIGNS (70% AND 90% UTILIZATION)

Design	t	CEs	DTEs	uCCs	Cuts	Min.	Mean	σ	CPU
AES45	90	45 750	8	0	3409	38	360.64	261.44	20.2
(70%)	91	45 764	11	0	3410	34	359.21	262.19	20.2
ART-A45	90	628 733	188	0	25 521	8	355.57	251.43	378.6
(70%)	91	636 038	195	0	25 560	9	353.85	251.99	378.3
ART-B45	90	1 884 320	477	0	76 550	10	355.44	252.88	2316.8
(70%)	91	1 906 267	510	0	76 609	9	353.66	253.11	2391.2
ART-C45	90	3 142 308	791	0	127 935	8	358.18	255.35	7895.8
(70%)	91	3 178 851	898	0	128 100	8	356.75	255.86	7991.6
AES45	90	46 098	14	0	3489	40	367.14	258.92	20.1
(90%)	91	46 109	12	0	3491	36	364.89	258.67	20.2
ART-A45	90	637 186	204	0	26 629	13	364.86	259.47	391.2
(90%)	91	644 574	260	0	26 698	9	363.12	260.02	395.6
ART-B45	90	1 912 100	962	0	79 836	10	364.08	258.08	2355.2
(90%)	91	1 934 210	980	0	79 929	8	362.66	258.46	2388.0
ART-C45	90	3 181 602	1477	0	132 303	8	362.74	257.20	8205.0
(90%)	91	3 218 341	1537	0	132 563	8	361.34	257.72	8513.1

where testcases of 90 nm designs are suffixed by "90" and 45 nm designs are suffixed by "45." In the 45 nm layouts, minimum spacing between features is 70 nm and minimum feature size is 50 nm. In the 90 nm layouts, minimum spacing between features is 140 nm, and minimum feature size is 100 nm. To match the 45 nm layouts with smaller feature sizes, we scale the 90 nm layouts by a factor of $0.5\times$, which results in 70 nm minimum spacing and 50 nm minimum feature size.

B. Experimental Results

1) *Solution Quality:* We sweep the coloring spacing lower bound as well as placement utilization to test our DPL color assignment approaches. For the CCD approach, we evaluate the solution quality according to number of unresolvable conflict cycles, and minimum, average and standard deviation of overlap lengths. Tables II and III show the experimental results of our layout decomposition system using the CCD approach, where "t" is the minimum coloring spacing, "CEs" is the number of conflict edges between features before node splitting, "DTEs" is the number of deleted touching edges between features in graph partitioning to reduce the ILP problem size (see Section III-F), "uCCs" is the number of unresolvable conflict cycles,[10] "Cuts" is the number of touching rectangle pairs with different colors, and "CPU" gives the total runtime of our layout decomposition algorithm in seconds. The minimum, average and the standard deviation of the overlap length values for all the cuts in the final decomposed layout are also reported, i.e., "$(min., mean, \sigma)$." Since node splitting is already performed before CCD, all the detected conflict cycles are unresolvable, i.e., the number of detected conflict cycles is equal to the number of uCCs.

From Tables II and III, the number of uCCs will increase as the minimum coloring spacing t increases. Note that in the results we do not count the cuts related to features within unresolvable conflict cycles. Thus, as the number of unresolvable conflict cycles increases, the reported total number of cuts may decrease. Tracking the uCC metric across 70% and 90% placement utilizations, we can infer that unresolvable conflict cycles mainly exist within each cell instance rather than between cell instances, i.e., there is only a small impact from the different placement utilizations. Higher placement utilization may not result in more unresolvable conflict cycles because unresolvable conflict cycles have a local property and depend more on relative positions between neighboring features. Most conflict cycles between cell instances can be removed using our node splitting and ILP node coloring methods. Figs. 16 and 17 show small examples of our layout decomposition solutions, where all layout is correctly decomposed with respect to the pre-specified overlap margin.

From the columns under "min." and "mean," we can see that all overlap lengths in the final mask decomposition are greater than the pre-specified overlap margin (8 nm overlap margin in 45 nm node [6]), which confirms the effectiveness of our layout decomposition system. In most cases, the minimum overlap length decreases as t increases when there are no coloring conflicts (i.e., when the number of uCCs or DCEs is 0). However, for ART-A45 with 70% placement utilization, the minimum overlap length increases as t increases. We attribute this to different deleted touching edges (DTEs) which introduce suboptimality to the coloring solution. (For example, when the overlap length related to a deleted touching edge is 8 nm and the corresponding touching nodes, which are partitioned into different clusters, are assigned different colors, the minimum overlap length in the final solution will be 8 nm even ILP obtains a greater overlap length in each cluster.)

For the PILP approach, we evaluate solution quality according to number of deleted conflict edges, overlap length and number of cuts. Tables IV and V show the experimental results, where "t" gives the minimum coloring spacing, "CEs" gives the number of conflict edges before node splitting in the conflict graph, "DTEs" is the number of deleted touching edges between features in graph partitioning to reduce the ILP problem size (see Section III-F),[11] "DCEs" gives the number of deleted conflict edges, "Cuts" gives the number of touching feature pairs with different colors, the columns under "min," "mean," and "σ," respectively, give minimum, average and

[10]Because node splitting is already performed before CCD, the number of uCCs gives the iterations of the CCD process, i.e., for each iteration of CCD, one uCC will be reported (unless there is no conflict cycle in the remaining conflict graph).

[11]The number of deleted touching edges is very small compared with the total number of partitions generated during layout decomposition. For example, for ART-C45 (90%) with $t = 91$ nm in Table V, the number of deleted touching edges (DTEs) is 1537. By contrast, the total number of partitions is 92 615 (the ratio is 1.66%).

TABLE IV
EXPERIMENTAL RESULTS OF OUR LAYOUT DECOMPOSITION SYSTEM USING PILP APPROACH ON THE POLY LAYER OF SCALED 90 NM DESIGNS (70% AND 90% UTILIZATION)

Design	t	CEs	DTEs	DCEs	Cuts	Min.	Mean	σ	CPU
AES90	76	85 729	0	0	128	15	131.23	113.84	32.6
(70%)	77	87 241	0	1	126	35	124.51	115.30	33.3
ART-A90	76	363 357	0	0	8572	25	258.31	169.21	239.0
(70%)	77	367 053	0	0	8680	20	258.35	170.62	243.9
ART-B90	76	723 143	0	0	17 076	25	261.31	170.32	624.6
(70%)	77	730 474	0	0	17 292	20	262.52	171.58	679.8
ART-C90	76	2 170 312	0	0	51 217	25	259.62	169.79	5678.4
(70%)	77	2 192 325	0	0	51 872	20	260.71	171.54	6022.3
ART-D90	76	3 612 466	0	0	85 311	25	260.05	170.09	11 985.4
(70%)	77	3 649 073	0	0	86 403	20	260.56	171.25	14 672.3
AES90	76	85 632	0	0	128	15	137.46	115.04	33.3
(90%)	77	87 149	0	1	126	35	130.92	116.83	33.4
ART-A90	76	363 336	0	0	8568	25	259.21	169.93	243.6
(90%)	77	367 047	0	0	8685	20	260.38	171.01	241.1
ART-B90	76	723 619	0	0	17 054	25	259.70	169.99	646.1
(90%)	77	730 978	0	0	17 258	20	259.94	171.22	630.9
ART-C90	76	2 170 098	0	0	51 216	25	260.20	169.85	6233.8
(90%)	77	2 192 132	0	0	51 824	20	260.74	171.36	4972.4
ART-D90	76	3 613 640	0	0	85 316	25	259.88	170.17	12 171.7
(90%)	77	3 650 513	0	0	86 394	20	260.41	171.40	11 823.5

TABLE V
EXPERIMENTAL RESULTS OF OUR LAYOUT DECOMPOSITION SYSTEM USING PILP APPROACH ON THE POLY LAYER OF 45 NM DESIGNS (70% AND 90% UTILIZATION)

Design	t	CEs	DTEs	DCEs	Cuts	Min.	Mean	σ	CPU
AES45	90	45 750	8	0	2973	40	387.15	264.93	40.3
(70%)	91	45 764	11	0	2977	36	384.90	266.93	40.2
ART-A45	90	628 733	188	0	24 290	8	361.07	253.92	564.6
(70%)	91	636 038	195	0	24 331	19	360.02	254.70	578.6
ART-B45	90	1 884 320	477	0	72 828	10	360.87	255.57	2887.4
(70%)	91	1 906 267	510	0	72 904	9	359.48	255.60	3219.2
ART-C45	90	3 142 308	791	0	121 916	8	364.40	258.20	8291.2
(70%)	91	3 178 851	898	0	122 125	9	362.66	257.80	8612.0
AES45	90	46 098	14	0	3071	40	397.33	259.73	41.0
(90%)	91	46 109	12	0	3078	36	393.49	259.30	40.9
ART-A45	90	637 186	204	0	25 432	13	370.20	262.31	612.1
(90%)	91	644 574	260	0	25 493	9	369.48	262.74	596.4
ART-B45	90	1 912 100	962	0	76 292	10	370.46	261.05	2892.2
(90%)	91	1 934 210	980	0	76 415	8	368.90	261.04	2912.4
ART-C45	90	3 181 602	1477	0	126 238	8	369.20	260.20	8129.2
(90%)	91	3 218 341	1537	0	126 482	8	367.38	259.88	8209.6

TABLE VI
EXPERIMENTAL RESULTS OF LAYOUT DECOMPOSITION SYSTEM USING CCD APPROACH ON METAL 1 AND METAL 2 LAYERS OF AES45 WITH 90% PLACEMENT UTILIZATION

Design	t	CEs	DTEs	uCCs	Cuts	Min.	Mean	σ	CPU
AES45-M1	54	2351	0	0	202	22	165.24	136.63	12.2
	55	2385	0	0	202	16	165.07	136.79	12.3
	56	4225	0	2	369	10	214.08	166.37	12.7
	57	4285	0	6	363	10	219.18	173.81	12.7
AES45-M2	54	47 384	0	269	2069	14	318.19	569.12	5.4
	55	47 425	0	270	2069	8	314.24	568.21	5.5
	56	47 585	0	286	2071	10	310.30	567.20	5.5
	57	47 591	0	837	1447	20	424.20	633.44	5.5

standard deviation of the overlap lengths, and "CPU" gives the total runtime in seconds. From the experimental results, our PILP method deletes some conflict edges to make the conflict graph two-colorable. The minimum overlap length is again always larger than the pre-specified overlap margin 8 nm. Comparing the results from the CCD and PILP approaches, the number of reported unresolvable conflict cycles (uCCs) is equal to that of deleted conflict edges (DCEs). In the CCD approach, when a conflict cycle is detected and reported as a uCC, all the conflict edges of the cycle will be removed to make the graph two-colorable. Then, another round of CCD starts, until no conflict cycles exist. That is, for each detected uCC, more than one edge will be deleted, and the deleted edges may further remove other conflict cycles. Hence, the reported number of uCCs may be less than or equal to the number of deleted conflict edges.

Tables VI and VII show the experimental results of our CCD and PILP approaches on Metal 1 and Metal 2 layers of

TABLE VII
EXPERIMENTAL RESULTS OF OUR LAYOUT DECOMPOSITION SYSTEM USING PILP APPROACH ON METAL 1 AND METAL 2 LAYERS OF AES45 WITH 90% PLACEMENT UTILIZATION

Design	t	CEs	DTEs	DCEs	Cuts	Min.	Mean	σ	CPU
AES45-M1	54	2351	0	0	200	22	162.96	129.68	22.4
	55	2385	0	0	200	16	163.21	129.70	22.5
	56	4225	0	2	366	10	216.94	165.04	23.0
	57	4285	0	6	360	10	221.12	172.63	22.7
AES45-M2	54	47 384	0	270	2079	14	312.48	555.25	20.7
	55	47 425	0	271	2079	8	310.59	562.40	20.6
	56	47 585	0	287	2082	10	305.57	555.49	21.6
	57	47 591	0	840	1456	20	416.76	621.58	23.5

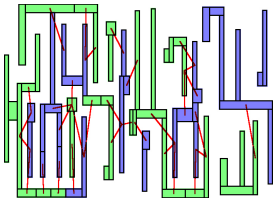


Fig. 16. Example of DPL layout decomposition in the poly layer.

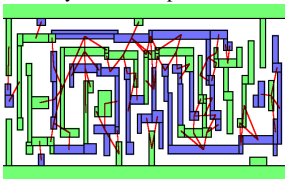


Fig. 17. Example of DPL layout decomposition in the M1 layer.

AES45 with 90% placement utilization (t = 54–57 nm). Again, the minimum overlap length is larger than the pre-specified overlap margin 8 nm. Comparing the results from the CCD and PILP approaches, the number of uCCs is less than the number of DCEs. We have discussed the reason for that above. From the results, Metal 2 layer is more difficult than Metal 1, i.e., for the same t value more uCCs are detected on Metal 2 layer. That is because the router (i.e., we use *Cadence SoC Encounter*) prefers to choose higher metal layers than Metal 1 layer to avoid local congestion. As a result, Metal 2 layer is likely to be more congested than Metal 1 layer, which results in more uCCs on Metal 2.

We do not directly compare the CCD and PILP approaches with respect to all metrics, as different methods and objective functions in problem formulations are used. For example, with the CCD approach the number of unresolvable conflict cycles (uCCs) captures layout pattern configurations which are not two-colorable; in the PILP approach the number of deleted conflict edges captures the minimum number of coloring conflicts. The PILP approach has larger runtime than the CCD approach because in the PILP approach, though there is no conflict cycle detection (which runs very fast), the number of constraints is larger than that in the CCD approach.

Note that different t values are used for different designs (i.e., 76–77 nm for the scaled 90 nm designs, 90–91 nm for the 45 nm designs, and 54–57 nm for Metal 1 and Metal 2 layers). By sweeping t values, we want to find example t values where the number of deleted conflict edges or uCCs is 0 (i.e., no layout changes are needed). If t values of 90–91 nm are used for the scaled 90 nm designs, there would be many deleted conflict edges or uCCs. The results show that the 45 nm designs can have larger t values with no layout changes. We attribute this to the more regular cell master patterns in the

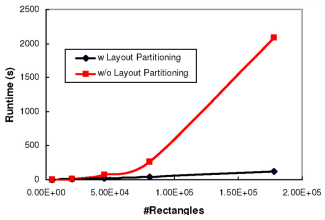


Fig. 18. ILP runtime: typical versus with layout partitioning.

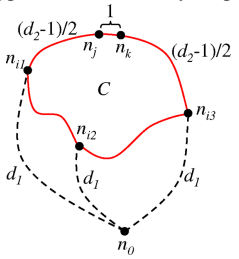


Fig. 19. Example of the BFS-induced graph and minimal conflict cycle C of node n_0: nodes n_{i1}, n_{i2}, and n_{i3} have shortest distance d_1 to n_0; only one path (i.e., the path between nodes n_{i1} and n_{i3} of length d_2) can be of odd length and its length is greater than the other two paths in C, because otherwise the graph is not a BFS-induced graph of n_0; adjacent nodes n_j and n_k are of maximum distance $D = (d_2 - 1)/2 + d_1$ from n_0.

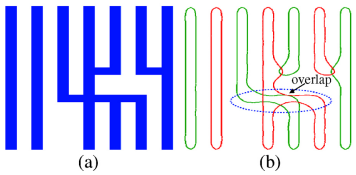


Fig. 20. Overlap between two mask exposures: (a) Original layout, and (b) superimposed post-etch contours where overlap is more litho-friendly. Source [24].

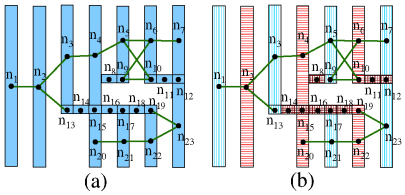


Fig. 21. Coloring solution with one feature having multiple exposures. (a) Conflict graph with conflict edges. (b) Each node without a conflict edge can have two colors.

Nangate 45 nm open cell library. However, metal layers have the smallest t values, which proves the difficulty of layout decomposition on metal layers.

2) *Scalability and Runtime:* Fig. 18 compares runtime of the CCD ILP with versus without layout partitioning.[12] Unsurprisingly, ILP runtime without layout partitioning increases very rapidly. On the other hand, the layout partitioning delivers much faster runtimes—for example, over 100× speedup for the $AES90$ testcase compared to naive use of ILP.

3) *Verification of DPL:* We have verified DPL layouts generated by our layout decomposition system with both ILP approaches using *Mentor Calibre DRCv2.6.9-11* [3]. Specifically, we set up three key design rule checks (DRCs) as follows: 1) the minimum spacing check rule in the DPL mask layouts is increased up to t; 2) the minimum line width checks for DPL are the same as those in single-exposure lithography; and 3) overlap length checks are performed by use of the AND boolean shape operation, i.e., intersections of features in

[12]Here, layout partitioning refers to both layout partitioning and graph partitioning methods to reduce the ILP problem size as discussed in Section III-F.

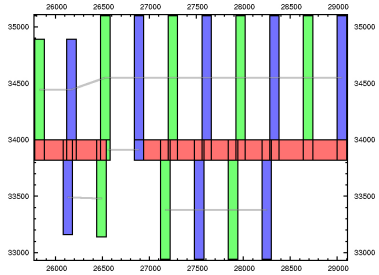


Fig. 22. Coloring solution for a real layout with one feature having multiple exposures: blue and green features appear on each mask and red features appear on both masks.

the two mask layouts correspond to overlaps at node splitting points, and must be larger than the prescribed overlap margin (i.e., 8 nm). The DRC is performed on layouts having extended features at the dividing points. Per the three design rule checks, no design rule violation exists in any of our testcases.

V. Conclusion and Ongoing Work

We have proposed different layout decomposition approaches to address design needs for DE patterning at 45 nm and below. Our approaches practically and effectively improve overlap length and hence lithography yield. Experimental results with real and artificial testcases show that the overlap lengths in the final layout are not less than the pre-specified overlap margin (e.g., 8 nm margin in 45 nm node) with all coloring conflicts reported, confirming the effectiveness of our approaches.

Our ongoing research is in the following directions.

1) The two mask exposures in DPL can result in distinct CD populations with different statistical distributions, which may increase guardbanding compared to the guardband of a single-exposure process. We are investigating optimal timing/power model guardbanding under the bimodal CD distribution in DPL.
2) We seek variability-awareness in the DPL layout decomposition cost function. Examples are 1) minimizing the difference between the pitch distributions of two masks, and 2) minimizing the number of distinct DPL layout solutions across all instances of a given master cell (to reduce variability between the instances).
3) We are also working on the DPL layout decomposition problem with balanced mask layout density and forbidden pitch interval constraints.

Appendix A
Minimal Conflict Cycle Detection

Given a connected conflict graph as in Fig. 13, conflict cycles are detected from a randomly chosen starting node (e.g., n_0) using Algorithm A with *flag* for cycle reporting set to be *false*. From Lines 9–15 in Algorithm A, the algorithms are called in the following order: 1) Algorithm A (*flag* = *false*), 2) Algorithm A (*flag* = *true*), and 3) Algorithm 3. Time complexity of the CCD algorithm is $O(V + E)$, where V and E are, respectively, the number of nodes and edges in the conflict graph G.

Theorem 1: The CCD algorithm given in Algorithms A and 3 can detect the minimal conflict cycle of starting node n_0.

Algorithm 2 CCD Algorithm

Input: Conflict graph G, starting node n_0 and *flag* for cycle reporting.
Output: Report one conflict cycle if there exist any conflict cycles.

```
1: Set distance d_i ← −∞ for each node n_i ∈ G;
2: Make a queue Q and enqueue node n_0 ∈ G into Q;
3: Set distance d_0 ← 0 for n_0;
4: while Q is not empty do
5:     Dequeue the first node n_j in Q;
6:     for all nodes n_k adjacent to n_j do
7:         if d_k ≥ 0 then
8:             if d_k = d_j then
9:                 if flag = true then
10:                    Call Algorithm 3 to report a conflict cycle;
11:                    return;
12:                else
13:                    Call Algorithm A on the BFS-induced graph of n_0 with starting
       node n_j and flag = true;
14:                return;
15:        else
16:            Set d_k ← d_j + 1;
17:            Enqueue n_k into Q;
```

Proof: Given a conflict graph G and a starting node n_0, assume without loss of generality that cycle $C = (n_1, n_2, \ldots, n_j, n_k, \ldots, n_l)$ of length l is a minimal conflict cycle of n_0, i.e., by BFS search from n_0, all nodes on C can be first visited before those of any other minimal conflict cycle. We first show that two adjacent nodes on C have largest distance (call this distance D) to n_0.

1) Assume there is exactly one node n_{near} on C of shortest distance d_1 to n_0. Since C is a minimal cycle of odd length l, there are two adjacent nodes on C with maximum distance $(l-1)/2$ from n_{near}, i.e., there are two nodes of maximum distance $(l-1)/2 + d_1$ from n_0.
2) Assume there are m ($m > 1$) nodes on C having shortest distance d_1 to n_0, which partition C into m paths. Fig. 19 shows an example where three nodes n_{i1}, n_{i2}, and n_{i3} partition C into three paths. Among all the m paths, there must be only one path of odd length and its length must be greater than all the other even-length paths, because otherwise the BFS-induced graph of n_0 cannot cover all the paths in C (i.e., C is not the minimal conflict cycle of n_0). Assume the length of this odd-length path is d_2, there must be two adjacent nodes on the path of maximum distance $D = (d_2 - 1)/2 + d_1$ from n_0 (e.g., nodes n_j and n_k in Fig. 19).

Next, let the two adjacent nodes be n_j and n_k. By BFS search from n_0, one conflict cycle can be found when reaching n_j and n_k, where the length of the conflict cycle is $2D + 1$. And all nodes in the minimal conflict cycle are visited during this BFS search.

Without loss of generality, we assume the second round of the BFS search in the BFS-induced graph of n_0 starts from node n_j. Assume the farthest nodes on C from node n_j are n_1 and n_2, respectively. Then there are two paths of length $(l-1)/2$ from n_j to n_1 and n_2. Since C is a minimal conflict cycle of n_0, there is no path of length less than $(l-1)/2$ from n_j to n_1 and n_2. Thus, after $(l+1)/2$ steps of BFS search, all nodes in the minimal conflict cycle have been visited. The conflict cycle reporting (Algorithm 3), following links back, collects all nodes the minimal conflict cycle.

Algorithm 3 Conflict Cycle Reporting Algorithm

Input: Conflict graph G with marked distances and nodes n_j and n_k in the detected conflict cycle in Algorithm A.
Output: Report the nodes in the detected conflict cycle in a doubly-linked list, where edges of the cycle are between adjacent nodes in the list.

```
1: Make a map F to store the parent node for each node;
2: Set F(n_j) ← NULL, F(n_k) ← NULL;
3: Make a queue Q' and enqueue nodes n_j and n_k into Q';
4: while Q' is not empty do
5:     Dequeue the first node n_r in Q';
6:     for all nodes n_s adjacent to n_r do
7:         if d_s + 1 = d_r then
8:             if n_s is visited then
9:                 Make a doubly-linked list L;
10:                Push n_s into L;
11:                Push n_r to the back of L;
12:                Set n_f ← F(n_r);
13:                while n_f ≠ NULL do
14:                    Push n_f to the back of L;
15:                    Set n_f ← F(n_f);
16:                Set n_f ← F(n_s);
17:                while n_f ≠ NULL do
18:                    Push n_f to the front of L;
19:                    Set n_f ← F(n_f);
20:                return L;
21:            Set F(n_s) ← n_r;
22:            Mark n_s as visited;
23:            Enqueue n_s into Q';
24:            break;
```

Similarly, it can be proved that when n_0 is on a minimal conflict cycle, one round of BFS search is sufficient to find this minimal conflict cycle. ■

APPENDIX B
MULTIPLE MASK EXPOSURES FOR ONE FEATURE

Our CCD and PILP approaches are easily extended to find coloring solutions for features to be exposed on multiple masks, i.e., one feature can have two colors. It may be desirable for some cases that overlap between the two mask exposures be enabled to improve the litho performance [24]. Fig. 20 shows an example layout where overlap between the two mask exposures are more litho-friendly. Our DPL color assignment approaches can be modified to allow one node to have two colors (proof omitted due to space limit).

Theorem 2: A node n in the conflict graph G can be assigned two colors without violating the minimum coloring spacing t iff n is not connected with conflict edges. ■

According to Theorem 2, any node not connected to conflict edges can be assigned two colors. To enable maximum flexibility for the overlaps, we need to fracture the layout polygons into rectangles in both horizontal and vertical directions. Then the conflict graph will be constructed over the rectangles according to the minimum coloring spacing t. We will compute the coloring solution using either the CCD or PILP approach. When the coloring solution is obtained, we perform a postprocessing step to assign two colors to all those nodes without conflict edges. Since the objective is to maximize the overlap between features on the two exposures, only those polygons with differently colored features will be processed for the both-mask color assignment, i.e., assign two colors to one feature, and the polygons with all their features assigned the same color will not be considered. For a polygon with different colors, we start from two touching features with different colors and perform BFS search according to touching edges. Wherever there are two touching features with different

colors,[13] each feature will be assigned two colors if it is not connected with a conflict edge. The BFS search in a polygon continues until all the features in the polygon are visited.

Fig. 21 gives an example of the coloring solutions for the layout in Fig. 20. In the conflict graph shown in Fig. 21(a), there are eight nodes without conflict edges, i.e., n_8, n_{11}, n_{12}, n_{14}, n_{15}, n_{16}, n_{17}, and n_{18}, which are assigned two colors and will appear on both masks. The final coloring solution is given in Fig. 21(b). Fig. 22 shows the coloring solution for part of a real layout where blue and green features appear on each mask and red features appear on both masks.

REFERENCES

[1] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 465–472.

[2] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Revisiting the layout decomposition problem for double patterning lithography," in *Proc. SPIE Photomask Technol.*, vol. 7122. 2008, p. 71220N.

[3] *Calibre User's Manual* [Online]. Available: http://www.mentor.com/

[4] *Design Compiler User's Manual* [Online]. Available: http://www.synopsys.com/

[5] *SoC Encounter User's Manual* [Online]. Available: http://www.cadence.com/

[6] *International Technology Roadmap for Semiconductors, Lithography Chapter*, 2007 [Online]. Available: http://public.itrs.net/

[7] 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 32 nm HP and beyond," in *Proc. SPIE Conf. Design Manufac. Through Design-Process Integr.*, 2007, p. 65211K.

[8] G. Capetti, P. Cantù, E. Galassini, A. V. Pret, C. Turco, A. Vaccaro, P. Rigolli, F. D'Angelo, and G. Cotti, "Sub-k1 = 0.25 Lithography with double patterning technique for 45 nm technology node flash memory devices at λ = 193 nm," in *Proc. SPIE Conf. Optical Microlithogr.*, 2007, p. 65202K.

[9] C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Zelikovsky, "Bright-field AAPSM conflict detection and correction," in *Proc. Design, Automat. Test Eur. (DATE)*, 2005, pp. 908–913.

[10] A. B. Kahng, S. Vaya, and A. Zelikovsky, "New graph bipartizations for double-exposure, bright field alternating phase-shift mask layout," in *Proc. Asia South Pacific Design Automat. Conf.*, 2001, pp. 133–138.

[11] C. Chiang, A. B. Kahng, S. Sinha, and X. Xu, "Fast and efficient phase conflict detection and correction in standard-cell layouts," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, 2005, pp. 149–156.

[12] M. Drapeau, V. Wiaux, E. Hendrickx, S. Verhaegen, and T. Machida, "Double patterning design split implementation and validation for the 32 nm node," in *Proc. SPIE Conf. Design Manufac. Through Design-Process Integr.*, 2007, p. 652109.

[13] M. Dusa, J. Quaedackers, O. F. A. Larsen, J. Meessen, E. van der Heijden, G. Dicker, O. Wismans, P. de Haas, K. van Ingen Schenau, J. Finders, B. Vleeming, G. Storms, P. Jaenen, S. Cheng, and M. Maenhoudt, "Pitch doubling through dual-patterning lithography challenges in integration and litho budgets," in *Proc. SPIE Conf. Optical Microlithogr.*, 2007, p. 65200G.

[14] P. Rigolli, C. Turco, U. Iessi, G. Capetti, P. Canestrari, and A. Fradilli, "Double patterning overlay budget for 45 nm technology node single and double mask approach," *J. Vacuum Sci. Technol. B: Microelectron. Nanometer Struct.*, vol. 25, no. 6, pp. 2461–2465, 2007.

[15] J. Finders, M. Dusa, and S. Hsu, "Double patterning lithography: The bridge between low k1 ArF and EUV," *Microlithography World*, Feb. 2008.

[16] *Hardmask* [Online]. Available: http://en.wikipedia.org/wiki/Hardmask/

[17] A. B. Kahng, X. Xu, and A. Zelikovsky, "Fast yield-driven fracture for variable shaped-beam mask writing," in *Proc. SPIE Conf. Photomask Next-Generation Lithogr. Mask Technol.*, 2006, p. 62832R.

[18] S.-M. Kim, S.-Y. Koo, J.-S. Choi, Y.-S. Hwang, J.-W. Park, E.-K. Kang, C.-M. Lim, S.-C. Moon, and J.-W. Kim, "Issues and challenges of double patterning lithography in DRAM," in *Proc. SPIE Conf. Optical Microlithogr.*, 2006, p. 65200H.

[19] C.-M. Lim, S.-M. Kim, Y.-S. Hwang, J.-S. Choi, K.-D. Ban, S.-Y. Cho, J.-K. Jung, E.-K. Kang, H.-Y. Lim, H.-S. Kim, and S.-C. Moon, "Positive and negative tone double patterning lithography for 50 nm flash memory," in *Proc. SPIE Conf. Optical Microlithogr.*, 2006, p. 615410.

[20] C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*. New York: Wiley, 2007.

[21] M. Maenhoudt, J. Versluijs, H. Struyf, J. V. Olmen, and M. V. Hove, "Double patterning scheme for sub-0.25 k1 single damascene structures at NA=0.75, λ = 193 nm," in *Proc. SPIE Conf. Optical Microlithogr.*, 2005, pp. 1508–1518.

[22] W.-Y. Jung, C.-D. Kim, J.-D. Eom, S.-Y. Cho, S.-M. Jeon, J.-H. Kim, J.-I. Moon, B.-S. Lee, and S.-K. Park, "Patterning with spacer for expanding the resolution limit of current lithography tool," in *Proc. SPIE Conf. Design Process Integr. Microelectron. Manufact.*, 2006, p. 61561J.

[23] J. Rubinstein and A. R. Neureuther, "Post-decomposition assessment of double patterning layout," in *Proc. SPIE Conf. Optical Microlithogr.*, 2008, p. 69240O.

[24] A. Sezginer and M. Dusa, "DPT panel discussion," in *Proc. SPIE Conf. Adv. Lithogr.*, 2007.

[25] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph bipartitioning," in *Proc. Asia South Pacific Design Automat. Conf.*, 2000, pp. 661–666.

[26] *NANGATE* [Online]. Available: http://www.nangate.com/

[27] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *Proc. Int. Symp. Phys. Design*, 2009, pp. 107–114.

[28] M. Cho, Y. Ban, and D. Z. Pan, "Double patterning technology friendly detailed routing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 506–511.

[29] H. Haffner, J. Meiring, Z. Baum, and S. Halle, "Paving the way to a full chip gate level double patterning application," in *Proc. SPIE Conf. Photomask Technol.*, 2009, p. 67302C.

[30] C. Cork, J. C. Madre, and L. Barnes, "Comparison of triple patterning decomposition algorithms using aperiodic tiling patterns," in *Proc. 15th SPIE Photomask Next-Generat. Lithogr. Mask Technol.*, 2008, p. 702839.

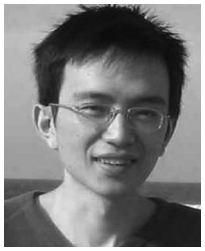
[31] K. Lucas, C. Cork, A. Miloslavsky, G. Luk-Pat, L. Barnes, J. Hapli, J. Lewellen, G. Rollins, V. Wiaux, and S. Verhaegen, "Interactions of double patterning technology with wafer processing, OPC and design flows," in *Proc. 21st SPIE Optical Microlithogr.*, 2008, p. 692403.

Andrew B. Kahng (SM'01–F'10), photograph and biography not available at the time of publication.

Chul-Hong Park (S'03–M'10) received the B.S. and M.S. degrees in mathematics from Kyung Hee University, Seoul, Korea, in 1992 and 1994, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California at San Diego, La Jolla, in 2008.

He is currently a Principal Engineer with the Semiconductor Research and Development Center, Samsung Electronics Company, Seoul, Korea. He currently leads the Integration of Design and Technology Team which has developed computational design rule and computational lithography solutions. He has published over 30 journal and conference papers, and is the holder of seven patents. His current research interests include nanometer physical design, design for manufacturing, integration of transistor/layout/circuit, and design/lithography for emerging technologies.

Dr. Park received the 1998 Honorable Outstanding Researcher Award and two Best Paper Awards in Samsung Electronics. He also received the 2009 Semiconductor Research Corporation Inventor Recognition Award.



Xu Xu (S'01) received the B.S. degree from Special Class for Gifted Young, University of Science and Technology of China, Hefei, China, in 1998, and the Ph.D. degree in computer science from University of California at San Diego, La Jolla, in 2006.

He was with Ammocore Technology, Inc., Santa Clara, CA, in 2002, Blaze DFM Inc., Sunnyvale, CA, from 2006 to 2008, and Magma Design Automation, San Jose, CA, in 2008. He is currently a Senior Software Engineer with Synopsys, Inc., Mountain View, CA. He has published over 30 papers on very large scale integration physical design, DNA array design, and integrated circuit manufacturing cost minimization. His current research interests include design for manufacturing and detailed routing.

Hailong Yao (S'07–M'09), photograph and biography not available at the time of publication.

[13] If one feature is assigned two colors and the touching feature is assigned one color, they are also regarded as being assigned different colors.