Revisiting the Layout Decomposition Problem for Double Patterning Lithography*

Andrew B. Kahng[a,b], Chul-Hong Park[b], Xu Xu[c], and Hailong Yao[a]
[a]CSE and [b]ECE Departments, UC San Diego, La Jolla, CA
[c]Magma Design Automation, San Jose, CA
abk@cs.ucsd.edu, chpark@vlsicad.ucsd.edu, xuxu@magma-da.com, hailong@cs.ucsd.edu

ABSTRACT

In double patterning lithography (DPL) layout decomposition for 45nm 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*.[5, 11, 14] 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 a layout decomposition algorithm that incorporates integer linear programming (ILP), phase conflict detection (PCD), and node-deletion bipartization (NDB) methods. We evaluate our approach on both real-world and artificially generated testcases in 45nm technology. Experimental results show that our proposed layout decomposition method effectively decomposes 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. While we have previously reported other facets of our research on DPL pattern decomposition,[6] the present paper differs from that work in the following key respects: (1) instead of detecting conflict cycles and splitting nodes in conflict cycles to achieve graph bipartization,[6] we split all nodes of the conflict graph at all feasible dividing points and then formulate a problem of bipartization by ILP, PCD[8] and NDB[9] methods; and (2) instead of reporting unresolvable conflict cycles, we report the number of deleted conflict edges to more accurately capture the needed design changes in the experimental results.

1. INTRODUCTION

Conventional immersion lithography is unlikely to take the industry to 32nm node patterning, while *Double Patterning Lithography* (DPL) is a primary lithography candidate for that technology node. 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 (DOF). When using two litho and two etch steps, DPL increases manufacturing cost due to its complex process flows, and overlay control between the two patterning exposures becomes a critical issue.

Two primary approaches to DPL are the *LELE* (litho-etch-litho-etch) and *self-aligned* approaches. The first etch step in LELE[14] is necessary to transfer the pattern of the first resist layer into an underlying hardmask[15, 19] 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. In self-aligned DPL,[17, 20] 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 materials between spacers produce the patterns for the second layer.[18, 21] The major concern of DPL is overlay control, which leads to requirements for more accurate overlay metrology, more representative sampling, reduced model residuals, and improved overlay correction.[12] Regolli et al.[13] analyzed overlay margin in double exposure patterning. According to the ITRS,[4] DPL requires overlay control of between 9nm and 6nm, which is a major hurdle for production deployment.

A key issue in DPL from the design point of view is the decomposition of the layout for multiple exposure steps.[11] This recalls strong Alt-PSM (Alternating Phase Shift Mask) coloring issues and automatic phase

* Research at UCSD was supported in part by the Semiconductor Technology Academic Research Center (STARC).

conflict detection and resolution methods.[8] DPL layout decomposition must satisfy the requirement that two features are 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.[5, 22] In such cases, at least one feature must be *split* into two or more parts. The pattern splitting increases manufacturing cost and complexity due to (1) generation of excessive line-ends, which causes 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 (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, as well as other concerns such as forbidden-pitch and other design rule restrictions on each mask, as well as layout density balance across masks.

In this work, we address DPL layout decomposition using integer linear programming (ILP), phase conflict detection (PCD),[8] and node-deletion bipartization (NDB)[9] methods. A preprocessing step fractures layout features into small pieces according to vertex coordinates of neighboring features. From the fractured polygon pieces, we optimize polygon splitting with a process-aware cost function that avoids small jogging line-ends, maximizes overlap at dividing points of polygons, and preferentially makes splits at landing pads, junctions and long runs.[11] A layout partitioning heuristic helps achieve scalability for large layouts. We present an overall layout decomposition method which includes layout fracturing, graph construction, node splitting and graph updating, and ILP/PCD/NDB based graph bipartization and node coloring processes. Our contributions are as follows.

- After layout fracturing, we perform node splitting to further split mask features at all possible dividing points with maximum overlap length after decomposition, so that the pre-specified overlay margin can be attained. This allows use of ILP/PCD/NDB based graph bipartization and node coloring methods to determine optimal or near-optimal dividing points with minimum design changes, maximized overlap lengths, and minimum line-ends.
- Our ILP based node color assignment algorithm determines a coloring solution using the possible dividing points to maximize overlap length with the required overlap margin, while avoiding design rule violations and minimizing the number of line-ends. When there is no feasible color assignment solution, our ILP algorithm obtains a feasible solution with minimal design changes.
- Compared with the ILP based color assignment algorithm, our PCD based method obtains a fairly good color assignment solution with much faster runtime.
- Our NDB based graph bipartization and node coloring method also runs faster than the ILP based method. However, the solution quality, after considering the number of design changes needed, the minimum and average overlap lengths, and the number of line-ends, is worse than both ILP and PCD based methods.

While we have reported other facets of our research on DPL pattern decomposition,[6] this paper differs from that work in two key respects. (1) In this work, instead of detecting conflict cycles and splitting nodes in conflict cycles for conflict cycle removal (i.e., graph bipartization[6]), we split all the nodes at all feasible dividing points and formulate the problem of bipartizing the conflict graph by ILP, PCD and NDB based methods. (2) Instead of reporting unresolvable conflict cycles, we report the number of deleted conflict edges, which is a more direct metric of design changes, in our experimental results. 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 different color assignment methods. Section IV describes testcases, experimental setup and experimental results. Section V concludes with ongoing research directions.

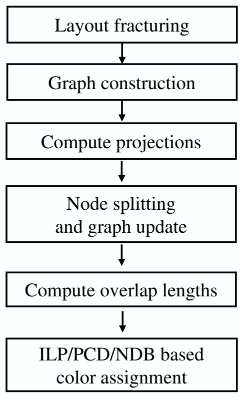

Figure 1: Overall DPL layout decomposition flow.

2. DPL LAYOUT DECOMPOSITION FLOW

Figure 1 shows our flow for DPL layout decomposition. Given a layout, the polygonal layout features are first fractured into a set of non-overlapping rectangles using a minimum-sliver fracturing algorithm.[16] The minimum-sliver fracturing minimizes the number of small rectangles and helps simplify downstream operations. Next, a *conflict graph* is constructed over the rectangular features according to the given minimum coloring spacing, t. 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 are touching each other, and a conflict edge exists between two nodes if the corresponding features are not touching each other, but the distance between the features is less than t.

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 find all possible dividing points on the layout feature nodes with maximized overlap length, we adopt the concept of *projections*. Details of node splitting are discussed in Section 3.3.

For each node in the graph, we compute its projections from other nodes that are adjacent and connected with conflict edges. Based on the projections, the dividing points with maximum overlap lengths for each layout feature can be computed. On one hand, not all layout features have feasible dividing points, i.e., the dividing points with the resulting overlap lengths greater than the required overlap margin. On the other hand, a given layout feature may have several dividing points where the feature can be split into several smaller features. The layout feature will be split whenever there is a feasible dividing point, and the conflict graph will be updated with the newly generated nodes and edges. In fact, the layout fracturing process can be regarded as a dividing point selection and node splitting process, where the projections are not computed and the required overlap margin is not guaranteed.

After node splitting and graph updating, ILP, PCD or NDB based graph bipartization and color assignment process is performed on the final conflict graph to find the coloring solution that minimizes the number of design changes[†], cuts[‡] and design rule violations, while maximizing the overlap lengths. Finally, a post-processing phase reports the number of design changes needed (i.e., the number of deleted conflict edges), the minimum, average, and standard deviation 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 any design rule violations in the final mask solution.

[†]By increasing the spacing between adjacent features, 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.

[‡]Where there are two touching nodes with different colors, there is a cut, which corresponds to two new line-ends.

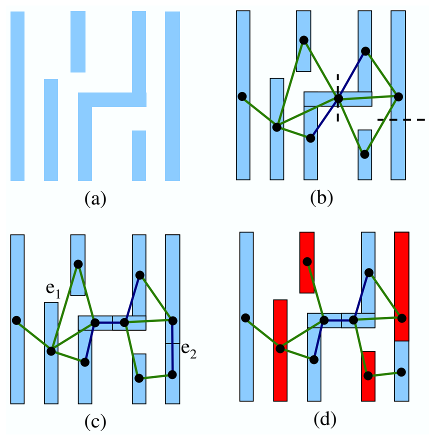

Figure 2. An example of the conflict graph with conflict edges (green) and touching edges (dark blue), and layout coloring according to our DPL layout decomposition flow: (a) input layout, (b) fractured layout and conflict graph before node splitting, (c) node splitting and graph updating, and (d) ILP, PCD, or NDB based graph bipartization and coloring.

Figure 2 illustrates the ILP, PCD and NDB based graph coloring according to our layout decomposition flow. Polygonal layout features in (a) are fractured into rectangles, over which the initial conflict graph is constructed. In the conflict graph, the conflict edges are green and the touching edges are dark blue. For all the nodes in the conflict graph, we compute their projections, according to which the feasible dividing points are computed as denoted by the dashed lines in (b). Then the node splitting process is carried out at the dividing points and the conflict graph is updated with newly generated nodes and updated edges as in (c). Finally, ILP, PCD, or NDB based color assignment is carried out to obtain the final coloring solution as given in (d), where some conflict and touching edges (e.g., conflict edge e_1 and touching edge e_2 in (c)) may be deleted to make the graph two-colorable.

3. DPL COLOR ASSIGNMENT PROBLEM

3.1. Problem Formulation

We now state the layout fracturing and DPL color assignment problem.[6]

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: (i) two non-touching fractured features corresponding to nodes n_i and n_j with $0 < d_{i,j} < t$ must be assigned different colors, and (ii) two touching features with $d_{i,j} = 0$, if assigned different colors, incur a cost $c_{i,j}$.

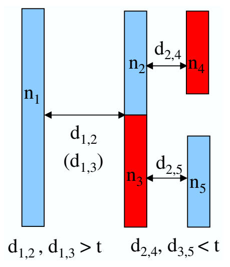

Figure 3. Example of color assignment problem: feature n_3 is assigned a different color from n_2 and n_5 because $d_{2,4} < t$ and $d_{3,5} < t$.

Figure 3 illustrates the color assignment problem. Feature n_2 (respectively, 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 (that is, risk) due to overlay error. Similar to Alt-PSM and SRAF (Sub-Resolution Assist Feature) techniques, a major barrier to widespread deployment of double patterning in random logic circuits is the lack of design compliance with layout decomposition and patterning requirements. There exist pattern configurations where features within the minimum coloring spacing cannot all be assigned different colors. In such cases, we must split at least one feature into two parts (e.g., n_2 and n_3 in Figure 3) – but this causes pinching under worst process conditions of defocus, exposure dose variation and misalignment. Thus, two line-ends at a dividing point must be sufficiently overlapped. Therefore, we should maximize the overlap between the respective mask layouts of n_2 and n_3.

We give details of the conflict graph construction, node splitting, and ILP/PCD/NDB graph bipartization and minimum cost color assignment steps in the following subsections.

3.2. Fracturing and Conflict Graph Construction

Given a layout L, a rectangular layout L_R is obtained by fracturing layout polygons into rectangles. We fracture the layout polygons into rectangles[16] so that distance computation and other feature operations (e.g., feature splitting) become easier. Our layout decomposition process begins with construction of a conflict graph based on the fractured layout. As illustrated in Figure 4, 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 non-touching 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 *touching edge* e^t.

If two non-touching features are connected by conflict edges in the graph, either they belong to different original polygonal layout features, or there do not exist any other features between them (i.e., no features entirely block the two non-touching features). In Figure 4, we see conflict edge set $E^C = \{e^c_{1,3}, e^c_{3,5}, e^c_{5,6}\}$ and touching edge set $E^T = \{e^t_{1,2}, e^t_{2,3}, e^t_{3,4}, e^t_{4,5}\}$. There is no conflict edge between n_2 and n_4 since node n_3 blocks these two nodes.

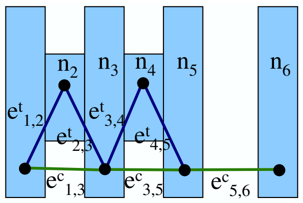

Figure 4. Example of conflict graph construction: every (rectangle) feature is represented by a node, and no feature entirely blocks two non-touching features that are connected by conflict edges in the graph.

3.3. Node Splitting

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. We perform node splitting with a similar flow as the rule-based node splitting.[6] The difference is that we perform node splitting for each node with feasible dividing points instead of only for those nodes in conflict cycles, i.e., cycles in the conflict graph which contain an odd number of conflict edges.[6] To compute the feasible dividing points, *projections* are calculated for each node from its adjacent nodes connected by conflict edges in the conflict graph. According to the projections of a given node, the overlap length for each feasible dividing point will be computed. For each dividing point with achievable overlap length greater than the required overlap margin, the node splitting process is carried out to split the node into two nodes at the dividing point. After node splitting at all feasible dividing points, the conflict graph will be updated. Then we apply an ILP, PCD or NDB based graph bipartization and coloring methods to decide which nodes to split (i.e., the corresponding pair of touching nodes are assigned different colors) in the final decomposition result having a maximized overlap lengths and minimized number of cuts. Note that the conflict graph may not be two-colorable after the node splitting process. In that case a minimized number of conflict edges will be deleted by either ILP, PCD or NDB based method. The deleted conflict edges will be reported and marked for layout optimization process to eliminate by design change. To reduce the cost of the design change, we preferentially delete conflict edges between features of different cell instances, so that by shifting the cell instances, the desired design changes can be obtained.

Not all layout configurations can be made two-colorable by the node splitting method. DPL layout decomposition fails when pattern features within the color spacing lower bound cannot be assigned different colors. Such a failure, where we require a layout change to delete some conflict edges, has two cases: (a) there is no dividing point to make the graph two-colorable among all of rectangles which have nonzero overlap length, and (b) the overlap length is less than the overlap margin, even if there is a dividing point to make the graph two-colorable. By construction, our node splitting process does not cause any violation of design rules or overlap margin for the newly generated nodes. On the other hand, since the layout features are split at all feasible dividing points, there is no solution other than conflict edge deletion to make the conflict graph two-colorable after the node splitting process. I.e., after the node splitting process, if the updated conflict graph is not two-colorable, the following ILP, PCD or NDB based method will delete some conflict edges, which corresponds to design changes, to make the graph two-colorable.

3.4. Layout Partitioning

In most placements, the conflict graph between cells is sparse, i.e., due to the required poly-to-cell boundary and whitespace between cells, there are not many edges between the cells. As a result, many *"islands"* can be found in the conflict graph. To improve both runtime and memory consumption, we also perform layout partitioning[6] to partition the conflict graph into connected components according to the connectivity information, with no edges or nodes of a given polygon occurring in multiple components. Each component has its separate conflict graph, and the coloring algorithms are carried out on each component in sequence. Because there are no edges between components, and no polygon has nodes in more than one component, the overall solution is obtained as the union of the solutions for all the small connected components.

3.5. ILP Based Min-Cost Color Assignment

After the node splitting process, the minimum-cost color assignment problem on the updated conflict graph is formulated as an integer linear programming problem. To formally state the ILP based problem formulation, we first introduce the following variables: (1) x_i: binary variable (0/1) for the color of rectangle r_i; (2) $y_{i,j}$: binary variable for touching edge $e^t_{i,j} \in E^T$. $y_{i,j} = 0$ when $x_i = x_j$, while $y_{i,j} = 1$ when $x_i \neq x_j$; and (3) $z_{i,j}$: binary variable 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 $x_i = x_j$. Then the ILP problem is formulated as

- **Objective:** minimize $\sum y_{i,j} \times (\alpha + \beta / OL_{i,j}) + \sum \gamma \times z_{i,j}$
- **Subject to:**

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

$$x_i = x_j \qquad \forall\ e^t_{i,j} \text{ and } l_{i,j} < FS_{min} \tag{2}$$

$$x_i = x_j \qquad \forall\ e^t_{i,j} \text{ and } OL_{i,j} < OM \tag{3}$$

$$\begin{cases} x_i - x_j \leq y_{i,j} \\ x_j - x_i \leq y_{i,j} \qquad \forall\ e^t_{i,j} \text{ and } OL_{i,j} \geq OM \end{cases} \tag{4}$$

$$\begin{cases} x_i + x_j - 1 \leq z_{i,j} \\ 1 - x_i - x_j \leq z_{i,j} \qquad \forall\ e^c_{i,j} \end{cases} \tag{5}$$

where $l_{i,j}$ is the length of the rectangle edge of r_i which is opposite to the touching edge between r_i and r_j, FS_{min} is minimum feature size, i.e., the threshold on the features, below which a design rule violation will occur, $OL_{i,j}$ is the overlap length between touching rectangles r_i and r_j, and OM is the required overlap margin.

Constraints (1) and (2) avoid design rule violations. In Constraint (1), 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 (2), 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 (3) enforces the required overlap margin. In Constraint (3), 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 (4) is related to the objective of maximizing the final overlap length. When the overlap length $OL_{i,j}$ between a pair of touching rectangles r_i and r_j is greater than the required overlap margin OM, a cost $\beta/OL_{i,j}$ is introduced in the objective function if r_i and r_j are assigned different colors ($x_i \neq x_j$), i.e., $y_{i,j} = 1$.

Constraint (5) is related to the objective of minimizing the number of design changes, i.e., the conflict edge removal process, which removes the conflict edges between rectangles by changing the design. A cost γ is introduced in the objective function 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., $z_{i,j} = 1$. α, β and γ are user-specified parameters for balancing the different objectives, e.g., a larger α corresponds to more emphasis on the minimization of the number of line-ends, a larger β relates to more improvement of overlap length, and a larger γ indicates that the number of design changes should be minimized. The results reported below are obtained with $\alpha = 1$, $\beta =$ the value of the maximum possible overlap length OL_{max}, and $\gamma = 1e6$[§].

3.6. PCD and NDB Based Min-Cost Color Assignment

Phase conflict detection based graph coloring method. For the phase conflict detection problem, a gadget based approach[8] is employed to produce an optimal edge-deletion bipartization solution for the planar graph. The algorithm consists of two main steps, as follows.

- *Planar graph embedding.* The phase conflict graph G is not necessarily an embedded planar graph, which is required by the optimal algorithm. Hence, G is converted to an embedded planar graph G' by greedily removing minimum weight conflict edges that cross other edges. These conflict edges are added to a potential set of conflicts P.
- *Optimal conflict removal for planar graph.* An optimal minimum-weight conflict cycle removal algorithm, *Bipartize*, is applied to G' for choosing the minimum set of conflicts that, when corrected, will produce a phase-assignable layout, i.e., a two-colorable layout. The list of edges deleted by the algorithm is added to D, a minimal set of conflicts whose removal ensures that G' is two-colorable.

In our phase conflict detection based problem formulation, the conflict edges in our conflict graph correspond to *feature* edges and our touching edges correspond to *conflict* edges in the *conflict cycle graph* in the gadget based approach.[8] Cost γ is assigned to each conflict edge to ensure that a minimum number of conflict edges will be deleted[¶]. For a touching edge between touching features, a cost is assigned considering two factors: the design rule violation, and the overlap length of the two features. When the size of the feature violates the design rule or the overlap length is less than the required overlap margin, a cost of 2γ is assigned to the touching edge. Otherwise, the cost on the touching edge is assigned as $\alpha + \beta/OL_{i,j}$, where $OL_{i,j}$ is the possible overlap length between the two features r_i and r_j. The values of the parameters α, β and γ are set as in Section 3.5. The phase conflict detection process outputs the deletions of conflict edges and touching edges that make the conflict graph two-colorable. A deleted conflict edge corresponds to a design change, whereas a deleted touching edge corresponds to a cutting on the layout feature, i.e., the two corresponding touching features generated by the layout fracturing or node splitting process will be assigned different colors.

Node-deletion bipartization based graph coloring method. For node-deletion bipartization, we revise our conflict graph and build a *conflict cycle graph*[8] as follows: (1) for each conflict edge $e_{i,j}$, create a new node n_k to substitute edge $e_{i,j}$ with two new edges $e_{i,k}$ and $e_{k,j}$, where $e_{i,k}$ is a *feature* edge and $e_{k,j}$ is a *conflict* edge, and (2) for each touching edge $e_{i,j}$, create a new node n_k to substitute edge $e_{i,j}$ with two new edges $e_{i,k}$ and

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

[¶]Again, $\gamma/2$ will be assigned to those edges between features of different cell instances.

$e_{k,j}$, where both $e_{i,k}$ and $e_{k,j}$ are *feature* edges. In the resulting conflict cycle graph, only the newly inserted nodes are deletable during the node deletion bipartization process, and the original nodes are fixed. The weights of the newly inserted nodes are set to be the same as those of the original conflict edges and touching edges, which are computed in the same way as in the phase conflict detection based method discussed above. After the conflict cycle graph is constructed, the node-deletion bipartization method[9] is executed; it outputs deleted nodes to generate a two-colorable graph, where the nodes can be mapped back to conflict edges and touching edges in the original conflict graph. As with the phase conflict detection based method, a deleted conflict edge corresponds to a design change, while a deleted touching edge corresponds to cutting of a layout feature.

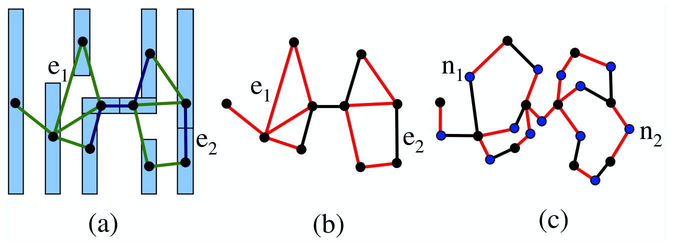

Figure 5. Example of the *conflict cycle graphs* constructed for PCD and NDB based graph coloring methods: (a) conflict graph with conflict edges (green) and touching edges (dark blue), (b) *conflict cycle graph* with *feature* edges (red) and *conflict* edges (black) for PCD based method, and (c) *conflict cycle graph* with new nodes (blue), *feature* edges (red) and *conflict* edges (black) for NDB based method.

Figure 5 shows an example of the constructed *conflict cycle graphs* from the conflict graph in Figure 2 for PCD and NDB based graph coloring methods. From the conflict graph in (a), the *conflict cycle graph* with *feature* edges (red) and *conflict* edges (black) for PCD based method is constructed in (b), and the *conflict cycle graph* with new nodes (blue), *feature* edges (red) and *conflict* edges (black) for NDB based method is constructed in (c). In the *conflict cycle graphs* of (b) and (c), the conflict cycles, i.e., the cycles with odd number of *feature* edges, need to be broken for graph bipartization, where possible solutions are the deletion of edges e_1 and e_2 in (b) and nodes n_1 and n_2 in (c). Therefore, to make the conflict graph in (a) two-colorable, the corresponding edges e_1 and e_2 in (a) need to be deleted to break all the conflict cycles.

After deleted conflict and touching edges are computed in our conflict graph using either the PCD or NDB based method, a heuristic breadth first search (BFS) based node coloring is performed on the two-colorable graph, and the number of deleted conflict edges, the number of cuts on features, and statistics (i.e., minimum, mean and standard deviation) of the final overlap lengths are extracted from the coloring solution.

4. EXPERIMENTAL RESULTS

Table 1. Parameters of the testcases: The minimum spacing between features and the minimum line width are 140nm and 100nm, which are scaled down by 0.4× to be 56nm and 40nm, respectively.

Design	#Cells	#Polygons	#Rectangles	Min. Spacing		Min. Width	
				Before	×0.4 Scaling	Before	×0.4 Scaling
AES	17304	90394	362380	140	56	100	40
TOP-B	60800	545000	2066800	140	56	100	40
TOP-C	305000	2725000	10334000	140	56	100	40
TOP-D	121600	1090000	4133600	140	56	100	40

Our layout decomposition system integrating ILP, PCD and NDB based node coloring methods is implemented in C++. We use one real-world design (*AES* from opencores.org) implemented with *Artisan* 90nm libraries using *Synopsys Design Compiler v2003.06-SP1.*[2] Because real-world synthesized netlists do not use all of the available standard-cell masters, we also run experiments with three artificial designs (*TOP-B*, *TOP-C* and *TOP-D*) with artificial netlists that instantiate more than 600 different types of cell masters from the same library. The testcases are placed with row utilizations of 70% and 90% using *Cadence First Encounter v3.3.*[3] Table 1 shows the parameters of the testcases. The minimum spacing between features in the 90nm library-based layout is 140nm, with minimum feature size of 100nm. To obtain experimental results that reflect

Table 2. Experimental results of ILP based layout decomposition system (four testcases, 70% and 90% utilization). t: minimum coloring spacing (nm). CEs: number of conflict edges between features. DEs: number of deleted conflict edges between features. $Cuts$: number of touching rectangle pairs with different colors. $(min., mean, \sigma)$: statistics of overlap lengths (nm) over all chosen splitting points. CPU: total runtime (s).

Design	t	CEs	DEs	Cuts	min.	mean	σ	CPU
AES (70%)	58	19117	0	29	56	301.45	117.20	17.0
	59	20888	0	38	56	157.89	100.26	17.4
	60	21209	0	38	56	156.92	100.56	17.3
	61	85733	1	133	18	68.77	78.72	22.3
TOP-B (70%)	58	402045	0	10701	13	150.49	75.63	404.0
	59	454272	300	13919	9	149.92	75.99	473.0
	60	461568	800	14075	11	153.04	73.88	440.0
	61	827517	9400	49863	11	123.57	92.24	747.9
TOP-C (70%)	58	2010351	0	53460	13	150.20	75.10	5252.8
	59	2271470	1500	69762	9	149.54	75.91	6412.4
	60	2307901	4000	69597	11	152.78	72.84	5336.2
	61	4138598	46820	244734	11	122.69	90.97	7625.6
TOP-D (70%)	58	804031	0	21389	13	150.24	75.26	1018.4
	59	908416	600	27845	9	149.69	75.97	1181.3
	60	923032	1600	27878	11	153.07	73.27	1183.6
	61	1655312	18812	98836	11	122.50	91.44	1763.9
AES (90%)	58	18972	0	29	56	297.48	114.45	17.0
	59	20721	0	35	56	149.63	74.97	17.6
	60	21043	0	35	56	148.66	75.12	17.2
	61	85636	1	129	18	63.67	65.15	22.4
TOP-B (90%)	58	402099	0	10696	13	150.39	75.48	354.7
	59	454343	300	13941	9	149.73	76.31	423.0
	60	461661	800	14027	11	152.83	73.62	448.1
	61	827675	9408	49634	11	122.94	91.62	735.6
TOP-C (90%)	58	2010400	0	53470	13	150.25	75.33	5597.0
	59	2271458	1500	69669	9	149.55	76.05	5873.4
	60	2307944	4000	69490	11	152.65	72.99	6629.0
	61	4138551	47001	243439	8	122.59	91.07	7026.7
TOP-D (90%)	58	803893	0	21456	13	150.85	73.93	1044.8
	59	907959	600	27898	9	149.83	74.98	1130.3
	60	922546	1600	27908	11	153.10	71.98	1228.0
	61	1655530	18887	98944	11	122.75	91.15	1748.6

future designs with smaller feature sizes, we scale down the GDS layout by a factor of 0.4, which results in 56nm minimum spacing and 40nm minimum feature size.

We sweep the color spacing lower bound t as well as placement utilization, and evaluate solution quality according to various metrics, including number of deleted conflict edges (design changes), overlap length and number of cuts. Table 2, Table 3 and Table 4 show the experimental results of our layout decomposition system based on our ILP, PCD and NDB based methods, respectively. In the tables, "t" gives the minimum coloring spacing; "CEs" gives the number of conflict edges in the conflict graph; "DEs" 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 standard deviation of the overlap lengths, and "CPU" gives the total runtime in seconds.

We cannot directly compare our new results with the previous work[6] in all metrics because different methods and objectives are used in the two works. E.g., in previous work, the number of unresolvable conflict cycles (uCCs) is used for recording the layout pattern configurations which are not two-colorable; whereas in this work, the number of deleted conflict edges (preferentially on edges between different cell instances) are used for recording the necessary design changes. Deleted conflict edges more accurately reflect the complexity of design changes than do uCCs. The two ILP based formulations have similar runtimes; in the new ILP problem formulation of this work, though there is no conflict cycle detection process, the number of constraints is larger than with the previous ILP based problem formulation.[6] Our heuristic PCD and NDB based methods obtain

Table 3. Experimental results of phase conflict detection based layout decomposition system (four testcases, 70% and 90% utilization). t: minimum coloring spacing (nm). CEs: number of conflict edges between features. DEs: number of deleted conflict edges between features. $Cuts$: number of touching rectangle pairs with different colors. $(min., mean, \sigma)$: statistics of overlap lengths (nm) over all chosen splitting points. CPU: total runtime (s).

Design	t	CEs	DEs	Cuts	min.	mean	σ	CPU
AES (70%)	58	19117	0	29	56	300.72	118.10	7.7
	59	20888	0	38	56	157.89	100.26	7.7
	60	21209	0	38	56	156.92	100.56	7.8
	61	85733	1	133	18	68.60	78.81	10.2
TOP-B (70%)	58	402045	250	10403	13	153.53	73.29	64.5
	59	454272	550	13435	9	152.32	73.81	70.1
	60	461568	801	13616	38	153.21	71.88	68.3
	61	827517	9845	46980	11	121.58	88.51	86.9
TOP-C (70%)	58	2010351	1232	51991	13	152.97	72.70	585.0
	59	2271470	2732	67064	9	151.68	73.45	603.0
	60	2307901	4006	68208	38	152.82	71.25	658.1
	61	4138598	48411	235968	11	121.69	88.19	589.3
TOP-D (70%)	58	804031	499	20774	13	153.34	73.26	159.6
	59	908416	1099	26782	9	151.71	73.56	168.5
	60	923032	1602	27224	38	152.83	71.53	159.8
	61	1655312	19584	94018	11	121.39	88.31	185.9
AES (90%)	58	18972	0	29	56	295.07	117.51	7.6
	59	20721	0	35	56	149.40	75.03	7.7
	60	21043	0	35	56	148.43	75.18	7.8
	61	85636	1	130	18	63.64	65.02	10.1
TOP-B (90%)	58	402099	254	10390	13	153.41	73.36	65.3
	59	454343	554	13400	9	151.81	73.68	67.9
	60	461661	801	13596	38	152.83	71.73	67.3
	61	827675	9812	47024	11	121.57	88.49	87.3
TOP-C (90%)	58	2010400	1244	51988	13	153.12	73.01	631.7
	59	2271458	2744	67024	9	151.76	73.64	607.7
	60	2307944	4000	68217	38	152.92	71.47	608.8
	61	4138551	48545	235745	11	121.59	88.21	562.2
TOP-D (90%)	58	803893	509	20810	13	153.84	71.69	160.7
	59	907959	1109	26798	9	151.76	72.36	177.0
	60	922546	1601	27250	38	152.84	70.23	168.1
	61	1655530	19653	93947	11	121.31	88.21	190.7

feasible solutions with much less runtime than the previous methods.

From the experimental results, the ILP based method deletes a smaller number of conflict edges to make the conflict graph two-colorable, i.e., a smaller number of design changes are needed using the ILP based layout decomposition method. Besides, since the overlap length is given higher priority by setting α and β to be 1 and the maximum possible overlap length OL_{max}, the minimum and mean overlap lengths obtained by the ILP based method are better than that of the other two methods when the number of deleted conflict edges are the same. Since the PCD and NDB based methods either delete more conflict edges or obtain smaller minimum/mean overlap lengths, the results confirm that the ILP based method can always obtain better solution quality than the other two methods. However, the PCD based method runs much faster than the ILP based method. The NDB based method obtains the worst solution quality with runtime between the PCD and ILP based methods.

We have verified DPL layouts generated by our layout decomposition system using *Mentor Calibre DRCv2.6.9-11*.[1] 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 to $2\times t + min.\ line\ width$; (2) the minimum linewidth 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 the two mask layouts correspond to overlaps at node splitting points, and must be larger than the prescribed overlap margin (e.g., 8nm). The DRCs are performed on layouts having extended features at the splitting points. Per the three design rule checks, we have confirmed that there is no design rule violation in all testcases.

Table 4. Experimental results of node-deletion bipartization-based layout decomposition system (four testcases, 70% and 90% utilization). t: minimum coloring spacing (nm). CEs: number of conflict edges between features. DEs: number of deleted conflict edges between features. $Cuts$: number of touching rectangle pairs with different colors. $(min., mean, \sigma)$: statistics of overlap lengths (nm) over all chosen splitting points. CPU: total runtime (s).

Design	t	CEs	DEs	Cuts	min.	mean	σ	CPU
AES (70%)	58	19117	0	29	43	101.55	38.25	15.9
	59	20888	0	33	41	93.82	40.04	16.1
	60	21209	0	33	40	92.82	40.04	16.1
	61	85733	4	121	18	49.88	38.28	23.0
TOP-B (70%)	58	402045	649	9054	11	102.55	64.22	143.9
	59	454272	1646	10709	9	104.87	62.83	154.3
	60	461568	1899	10927	9	106.27	63.18	158.9
	61	827517	14969	36624	8	87.79	71.27	222.3
TOP-C (70%)	58	2010351	3218	45283	11	103.79	64.91	1164.9
	59	2271470	8227	53522	9	106.13	63.57	1130.8
	60	2307901	9502	54632	9	107.73	63.79	1188.9
	61	4138598	80940	194955	8	86.54	72.48	1263.7
TOP-D (70%)	58	804031	1298	18105	11	102.91	64.48	342.5
	59	908416	3305	21388	9	104.87	62.89	354.7
	60	923032	3810	21841	9	106.26	63.19	350.9
	61	1655312	29845	73272	8	87.83	71.25	482.3
AES (90%)	58	18972	0	29	43	112.03	41.35	15.9
	59	20721	0	33	41	124.45	49.81	16.0
	60	21043	0	33	40	123.45	49.81	16.1
	61	85636	6	118	18	57.19	49.85	23.0
TOP-B (90%)	58	402099	654	9044	11	102.46	64.43	145.1
	59	454343	1645	10712	9	104.39	62.85	152.2
	60	461661	1894	10947	9	105.78	63.19	163.1
	61	827675	15000	36593	8	87.65	71.20	217.8
TOP-C (90%)	58	2010400	3229	45297	11	104.25	64.67	1109.1
	59	2271458	8248	53498	9	106.71	63.46	1181.0
	60	2307944	9504	54546	9	108.31	63.79	1164.8
	61	4138551	81080	194774	8	86.42	72.34	1272.0
TOP-D (90%)	58	803893	1367	17965	11	102.91	64.82	326.3
	59	907959	3358	21295	9	104.38	62.95	343.5
	60	922546	3850	21805	9	105.81	63.19	346.1
	61	1655530	29934	73161	8	87.40	71.16	469.0

5. CONCLUSIONS AND ONGOING WORK

We have proposed three layout decomposition approaches based on ILP, phase conflict detection and node-deletion bipartization methods to address design needs for double exposure patterning at 45nm and below. Our approaches practically and effectively improve overlap length and hence lithography yield. Experimental results with real-world and artificial testcases show that the overlap lengths in the final layout are not less than the pre-specified overlap margin (e.g., 8nm margin in 45nm node) with all necessary design changes reported, confirming the effectiveness of our approaches.

Our ongoing research is in the following directions.

- 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.
- We seek variability-awareness in the DPL layout decomposition cost function. Examples are (i) minimizing the difference between the pitch distributions of two masks, and (ii) minimizing the number of distinct DPL layout solutions across all instances of a given master cell (to reduce variability between the instances).
- Besides, we are also working on the DPL layout decomposition problem with balanced mask layout density and forbidden pitch intervals.

REFERENCES

1. Calibre User's Manual. http://www.mentor.com/.
2. Design Compiler User's Manual. http://www.synopsys.com/.
3. SOC Encounter User's Manual. http://www.cadence.com/.
4. International Technology Roadmap for Semiconductors. http://public.itrs.net/.
5. G. E. Bailey et al., "Double Pattern EDA Solutions for 32nm HP and Beyond", *Proc. SPIE Conf. on Design for Manufacturability Through Design-Process Integration*, 2007, pp. 65211K-1 - 65211K-12.
6. A. B. Kahng, C.-H. Park, X. Xu and H. Yao, "Layout Decomposition for Double Patterning Lithography", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 2008.
7. G. Capetti et al., "Sub k1 = 0.25 Lithography with Double Patterning Technique for 45nm Technology Node Flash Memory Devices at 193nm", *Proc. SPIE Conf. on Optical Microlithography*, 2007, pp. 65202K-1 - 65202K-12.
8. C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Zelikovsky, "Fast and Efficient Bright-Field AAPSM Conflict Detection and Correction", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 26(1) (2007), pp. 115-126.
9. A. B. Kahng, S. Vaya and A. Zelikovsky, "New Graph Bipartizations for Double-Exposure, Bright Field Alternating Phase-Shift Mask Layout", *Proc. Asia and South Pacific Design Automation Conference*, 2001, pp. 133-138.
10. C. Chiang, A. B. Kahng, S. Sinha and X. Xu, "Fast and Efficient Phase Conflict Detection and Correction in Standard-Cell Layouts", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 2005, pp. 149-156.
11. M. Drapeau, V. Wiaux, E. Hendrickx, S. Verhaegen and T. Machida, "Double Patterning Design Split Implementation and Validation for the 32nm Node", *Proc. SPIE Conf. on Design for Manufacturability Through Design-Process Integration*, 2007, 652109-1 - 652109-15.
12. M. Dusa et al., "Pitch Doubling Through Dual-Patterning Lithography Challenges in Integration and Litho Budgets", *Proc. SPIE Conf. on Optical Microlithography*, 2007, pp. 65200G-1 - 65200G-10.
13. P. Rigolli et al., "Double Patterning Overlay Budget for 45nm Technology Node Single And Double Mask Approach", *Journal of Vacuum Science and Technology B: Microelectronics and Nanometer Structures*, 25(6) (2007), pp. 2461-2465.
14. J. Finders, M. Dusa and S. Hsu, "Double Patterning Lithography: The Bridge Between Low k1 ArF and EUV", *Microlithography World*, Feb. 2008.
15. http://en.wikipedia.org/wiki/Hardmask.
16. A. B. Kahng, X. Xu and A. Zelikovsky, "Fast Yield-Driven Fracture for Variable Shaped-Beam Mask Writing", *Proc. SPIE Conf. on Photomask and Next-Generation Lithography Mask Technology*, 2006, pp. 62832R-1 - 62832R-9.
17. S.-M. Kim et al., "Issues and Challenges of Double Patterning Lithography in DRAM", *Proc. SPIE Conf. on Optical Microlithography*, 2006, pp. 65200H-1 - 65200H-7.
18. C. Lim et al., "Positive and Negative Tone Double Patterning Lithography for 50nm Flash Memory", *Proc. SPIE Conf. on Optical Microlithography*, 2006, pp. 615410-1 - 615410-8.
19. C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*, Wiley, 2007.
20. M. Maenhoudt, J. Versluijs, H. Struyf, J. Van Olmen, and M. Van Hove, "Double Patterning Scheme for Sub-0.25 k1 Single Damascene Structures at NA=0.75, λ=193nm", *Proc. SPIE Conf. on Optical Microlithography*, 2005, pp. 1508-1518.
21. W.-Y. Jung et al., "Patterning With Spacer for Expanding the Resolution Limit of Current Lithography Tool", *Proc. SPIE Conf. on Design and Process Integration for Microelectronic Manufacturing*, vol. 6125 pp. 61561J-1 - 61561J-9, 2006.
22. J. Rubinstein and A. R. Neureuther, "Post-Decomposition Assessment of Double Patterning Layout", *Proc. SPIE Conf. on Optical Microlithography*, 2008, pp. 69240O-1 - 69240O-12.