Layout Decomposition for Double Patterning Lithography*

Andrew B. Kahng[†‡], Chul-Hong Park[‡], Xu Xu[+], and Hailong Yao[†]
[†]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA
[+]Blaze DFM, Inc., Sunnyvale, CA
abk@cs.ucsd.edu, chpark@vlsicad.ucsd.edu, xuxu@blaze-dfm.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* [11, 9, 5]. However, there exist pattern configurations for which pattern features separated by less than the minimum color 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 includes graph construction, conflict cycle detection, and node splitting processes. We evaluate our technique 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.

1. INTRODUCTION

As Moore's law continues to drive performance and integration with smaller circuit features, lithography is pushed to new extremes. For 32nm node patterning, prospects for new lithography techniques such as extreme ultraviolet (EUV) and immersion ArF (IArF) are unclear. An EUV imaging system is composed of mirrors coated with multilayer structures designed to have high reflectivity at 13.5nm 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 $= 1.55 \sim 1.6$), and accompanying advances in high-index resists and optical materials. *Double Patterning Lithography* (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). DPL is likely to play an even more important role than previously anticipated, since the EUV adoption timeline has been delayed [6, 11]. However, DPL increases manufacturing cost in two fundamental ways: (1) 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 the *LELE* (litho-etch-litho-etch) and *self-aligned* approaches. In the LELE approach [11], the first etch step is necessary to transfer the pattern of the first resist layer into an underlying hardmask [12, 16] 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 [14, 17], the patterns for the first layer are transferred into the hardmask and then nitride spacers are formed on the sidewall 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 but leave only the material on the sidewalls. Then, film materials between spacers produce the patterns for the second layer [15, 18]. 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 [10]. 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 [9]. This recalls strong Alt-PSM (Alternating-Phase Shift Mask) coloring issues and automatic phase conflict detection and resolution methods [7]. 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 [5, 19]. 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.

Unlike previous works on Alt-PSM, which adopt graph bipartization algorithms, we formulate the optimization of DPL layout decomposition using integer linear programming (ILP). A pre-processing step fractures layout features into

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

ICCAD'08, November 10–13, 2008, San Jose, CA

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 [9]. A layout partitioning heuristic helps achieve scalability for large layouts. We achieve an overall layout decomposition method for DPL which includes graph construction, conflict cycle detection, and node splitting processes. Our contributions are as follows.

- Our *conflict cycle* detection algorithm efficiently finds patterns with unresolvable color assignment during the pattern splitting process. Because such patterns cannot be manufactured by DPL even with availability of layout decomposition and two mask exposures, this detection step enables fast feedback to the designer to modify the layout pattern.
- Our node splitting method determines mask features with maximum overlap length after decomposition, so that the pre-specified overlay margin can be observed.
- Our integer linear programming (ILP) based color assignment algorithm minimizes the number of line-ends and avoids design rule violations.
- Our layout partitioning technique improves scalability of the ILP-based coloring solution, whose runtime would otherwise increase unmanageably with layout size.

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.

2. DPL LAYOUT DECOMPOSITION FLOW

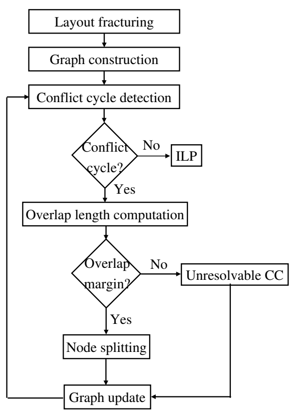


Figure 1: Overall DPL layout decomposition flow.

Figure 1 shows the overall flow for DPL layout decomposition. Given a layout, the polygonal layout features are first fractured into a set of non-overlapping rectangles using the minimum-sliver fracturing algorithm of [13]. 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; an edge exists between two nodes if the corresponding features are not touching each other, and 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 (i.e., thus adding new nodes and inducing new edges) so that the graph can be properly 2-colored. To this end, the key is the removal of *conflict cycles* (CCs), which are the odd-length - and hence not 2-colorable - cycles in the conflict graph. We use a breadth first search (BFS) based conflict cycle detection algorithm to find the conflict cycles in the graph.[1] When a conflict cycle is found, *node splitting* is applied to find the best layout feature to split, considering the maximization of overlap lengths. If the maximum possible overlap length is less than a given required overlap margin, then the layout has an *unresolvable conflict cycle* (uCC) which must be flagged to the designer for layout modification. Otherwise, the layout feature will be split into smaller features to remove the conflict cycle. (Note that the node splitting process is *required* to remove a conflict cycle while generating line-ends having overlap length greater than the required overlap margin; there is no other option to remove a conflict cycle other than costly layout modification.) The graph is then updated, and the conflict cycle detection and node splitting processes are iterated until no conflict cycle remains in the graph.

After the iterative conflict cycle detection and node splitting process, we perform ILP-based coloring on the final conflict cycle-free graph to find an optimal coloring solution, considering minimization of the number of cuts (or line-ends) and design rule violations. Post-processing functions include analysis of minimum 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.

Figure 2 illustrates DPL-based coloring according to our layout decomposition flow. Polygonal layout features in (a) are fractured into rectangles, over which the conflict graph is constructed and conflict cycles detected as shown in (b). To remove the conflict cycles in the conflict graph, the node splitting process is carried out along the dividing points denoted by the dashed lines in (b). After the node splitting process for conflict cycle removal, the conflict graph is updated with newly generated nodes and updated edges as in (c). Finally, ILP-based coloring is carried out to obtain the final coloring solution on the 2-colorable conflict graph (d).

3. DPL COLOR ASSIGNMENT PROBLEM

3.1 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: (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. (ii) Two touching features with $d_{i,j} = 0$, if assigned different colors, incur a cost $c_{i,j}$.

Figure 3 illustrates the color assignment problem. Feature n_2 (respectively, n_3) is assigned a different color from n_4

[1]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 ($\leq$ 10), whereas DFS-based detection is more efficient for conflict cycles with more edges ($>$ 10). Since most conflict cycles in the layouts we have studied have fewer than 7 edges, we adopt BFS-based conflict cycle detection.

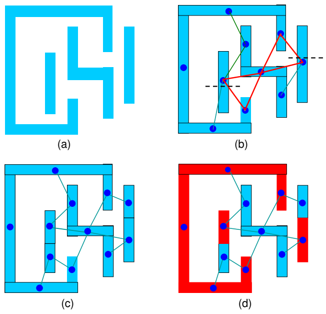


Figure 2: An example of the graph and layout coloring according to the DPL flow: (a) input layout, (b) fractured layout and conflict graph, (c) conflict cycle removal, and (d) ILP-based DPL coloring.

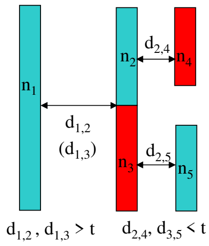


Figure 3: Example of color assignment problem: feature n_3 is assigned a different color from n_2 and n_5 because $d_{2,4} < t$ and $d_{3,5} < t$.

(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. We should maximize the overlap between the respective mask layouts of n_2 and n_3 in this case, as we now discuss.

Just as with 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 for which 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 – but this causes pinching under worst process conditions of defocus, exposure dose variation and misalignment. Thus, two line-ends at a dividing point (DP) must be sufficiently overlapped. Moreover, in Figure 4 below, the extended features (EF) that address the overlap requirement must still satisfy DPL design rules: the spacing between patterns at the dividing point must be greater than the minimum coloring spacing. Figure 4 shows how two dividing points lead to different minimum spacings after layout decomposition. In the figure, each layout decomposition can remove the conflict cycle. However, when extending patterns for overlay margin, the dividing point in Figure 4(a) causes violation of the minimum coloring spacing, t. The dividing point in Figure 4(b) maintains the minimum coloring spacing even with line-end extension for overlay margin.

Our layout decomposition solution includes conflict graph construction, conflict cycle detection, node splitting, and minimum cost color assignment. We give details of each step in the following subsections.

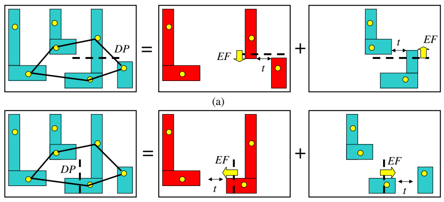


Figure 4: Two examples of dividing points. (a) Extended features at the dividing point cause coloring violation. (b) Extended features at the dividing point still satisfy the minimum coloring spacing rule.

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 into rectangles [13] 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 5, given a (post-fracturing) rectangular layout L_R, the layout graph $G = (V, E)$ 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 an edge e.

If two non-touching features are adjacent 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 5, we see edge set $\{E_{1,3}, E_{3,5}, E_{5,6}\}$. There is no edge between n_2 and n_4 since node n_3 blocks these two nodes.

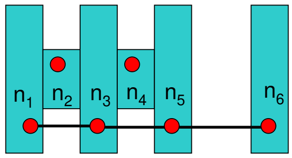


Figure 5: Example of conflict graph construction: every (rectangle) feature is represented by a node, and no feature entirely blocks two non-touching features that are adjacent in the graph.

3.3 Conflict Cycle Detection

To detect pattern configurations in which non-touching features cannot be assigned different colors, we find odd-length cycles in the conflict graph.

Definition 1: A **conflict cycle** is a cycle in the conflict graph which contains an odd number of *edges*.

Given a conflict graph as in Figure 2(b), we apply a breadth first search (BFS) technique, given in Algorithm 1 and 2, to detect conflict cycles. Time complexity of the conflict cycle detection algorithm is $O(V + E)$, where V and E are respectively the number of nodes and edges in the conflict graph G.

Conflict cycle detection and conflict cycle removal (cf. the node splitting process in Section 3.4) processes are carried out in an iterative manner. Each time a conflict cycle is detected, the conflict cycle removal process is invoked to remove the conflict cycle. Further rounds of detection and removal are performed until the graph G is conflict cycle-free. As described in the experimental results below, total runtime for the whole process, including the conflict cycle detection, conflict cycle removal and min-cost color assignment, is reasonable: less than 12 minutes for layouts of more than 545K features (2.1M rectangles after fracturing).[2]

Algorithm 1 Conflict cycle detection algorithm.

```
Input: Conflict graph G.
Output: Report the nodes in one conflict cycle if there are any conflict
    cycles.
Set distance d_i ← −∞ for each node n_i ∈ G;
Make a queue Q and enqueue node n_0 ∈ G into Q;
Set distance d_0 ← 0 for n_0;
while Q is not empty do
    Dequeue the first node n_j in Q;
    for all nodes n_k adjacent to n_j do
        if d_k ≥ 0 then
            if d_k = d_j then
                Report a conflict cycle as given in Algorithm 2;
                return;
            end if
        else
            Set d_k ← d_j + 1;
            Enqueue n_k into Q;
        end if
    end for
end while
```

Algorithm 2 Conflict cycle reporting algorithm.

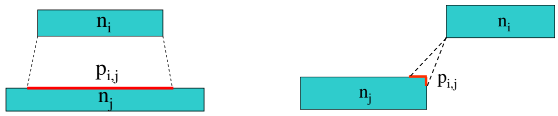
```
Input: Conflict graph G with marked distances and nodes n_j and n_k
    in the detected conflict cycle in Algorithm 1.
Output: Report the nodes in the detected conflict cycle in a double-
    linked list, where edges exist between adjacent nodes in the list.
Make a map F to store the father node for each node;
Set F(n_j) ← NULL, F(n_k) ← NULL;
Make a queue Q′ and enqueue nodes n_j and n_k into Q′;
while Q′ is not empty do
    Dequeue the first node n_r in Q′;
    for all nodes n_s adjacent to n_r do
        if d_s + 1 = d_r then
            if n_s is visited then
                Make a double-linked list L;
                Push n_s into L;
                Push n_r to the back of L;
                Set n_f ← F(n_r);
                while n_f ≠ NULL do
                    Push n_f to the back of L;
                    Set n_f ← F(n_f);
                end while
                Set n_f ← F(n_s);
                while n_f ≠ NULL do
                    Push n_f to the front of L;
                    Set n_f ← F(n_f);
                end while
                return L;
            end if
            Set F(n_s) ← n_r;
            Mark n_s as visited;
            Enqueue n_s into Q′;
        end if
    end for
end while
```

3.4 Node Splitting

Node splitting is applied to nodes in conflict cycles so that we may eventually obtain a graph without any conflict cycles. Each time a conflict cycle is detected, we compute the overlap lengths over all possible node splits that remove the conflict cycle (recall Figure 4). For each node with achievable overlap length greater than the required overlap margin, the node splitting process is carried out to split the node into two nodes and eliminate the conflict cycle. We apply an ILP-based coloring algorithm (Section 3.6 below) to decide which nodes to split in the final decomposition result having a minimized number of cuttings and maximized overlap lengths. The conflict graph is then updated with newly generated nodes and updated edges, and another iteration with BFS-based conflict cycle detection begins. The time complexity of node splitting is $O(C)$, where C is the number of nodes in the conflict cycle.

Definition 2: 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 t.

Fact 1: In the conflict graph, node projections between each pair of nodes that are adjacent in the conflict graph are non-empty.



(a) n_i is aligned to n_j. (b) n_i is not aligned to n_j.

Figure 6: Node projection examples.

Figure 6 shows two examples of node projections. We now give a more precise description of node splitting.

Definition 3: A horizontal (vertical) **merged projection** $m_h(P)$ ($m_v(P)$) for a given projection P on node n is the union of P and all the projections on n that horizontally (vertically) overlap with P.

Definition 4: Node projections are **separable** if they are disjoint.

Definition 5: The **overlap length** of a newly generated node for the corresponding dividing point is the length that the node can be extended across the dividing point without introducing new edges in the conflict graph.

Rule-based node splitting: Given a conflict cycle and a node n_i in the cycle, if (i) the horizontal (vertical) merged projections $m_h(P_{j,i})$ ($m_v(P_{j,i})$) and $m_h(P_{k,i})$ ($m_v(P_{k,i})$) corresponding to adjacent nodes n_j and n_k are separable, (ii) the resulting overlap lengths of the horizontal (vertical) splitting are not less than the given overlap margin, and (iii) there are no design rule violations after splitting, then node n_i can be horizontally (vertically) split into two nodes to remove the conflict cycle. The dividing point may be chosen in between the merged projections such that no merged projections are cut, and no violations of overlap margin or design rules occur.

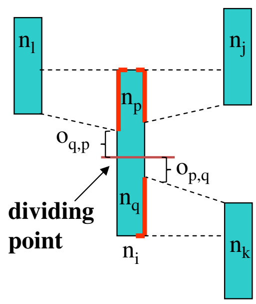


Figure 7: Example of rule-based node splitting: $o_{i,j}$, $o_{i,k}$ and $o_{i,l}$ are overlap lengths.

Figure 7 shows an example of rule-based node splitting. In the figure, assume there is a conflict cycle between nodes n_i, n_j and n_k, and the horizontal merged projections $m_h(P_{j,i}) = P_{j,i} \cup P_{l,i}$ and $m_h(P_{k,i}) = P_{k,i}$ on node n_i corresponding to nodes n_j and n_k are separable with overlap lengths not less

[2]This runtime includes all stages: layout partitioning, all rounds of conflict cycle detection and removal, ILP-based color assignment, etc. The total runtime of BFS-based conflict cycle detection (Algorithm 1 and 2) across all conflict cycle detection rounds is less than 0.42 seconds.

than the given overlap margin. Hence, node n_i can be split into two new nodes n_p and n_q at the dividing point, with corresponding overlap lengths of $o_{p,q}$ and $o_{q,p}$. The dividing point is in between the lower point of $P_{l,i}$ and the upper point of $P_{k,i}$[3]. Generally, the position of the dividing point is chosen so as to maximize the smaller overlap length. A more detailed illustration of overlap length is given in Figure 8, 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 larger than the given overlap margin to guarantee successful manufacturing. The overlap lengths of the touching features n_4 and n_5 are denoted as $o_{4,5}$ and $o_{5,4}$, respectively. 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 to touch the projection of feature n_7).

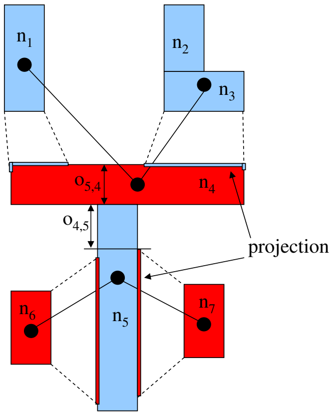


Figure 8: Example of overlap length calculation: $o_{4,5}$ and $o_{5,4}$ are the overlap lengths for n_4 and n_5, respectively.

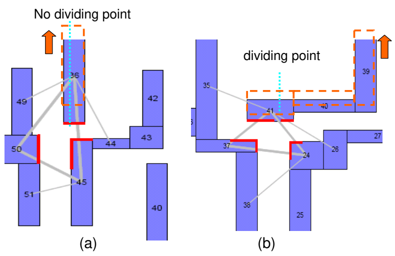


Figure 9: Example of unresolvable conflict cycle (uCC): (a) uCC with zero overlap length and (b) uCC with non-zero overlap length (less than the overlap margin).

Of course, not all conflict cycles can be eliminated 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, which we call an *unresolvable conflict cycle* (uCC), consists of two cases: (a) there is no dividing point to remove the conflict cycle 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 remove the conflict cycle. Figure 9 illustrates these two types of uCC. If we divide the rectangle in the center as shown in Figure 9(a), the size of the rectangle violates the minimum design rule (CD). Removal of the conflict cycle may be achieved by layout perturbation which increases the spacing to neighboring patterns to be $> t$ (as shown in orange color). In Figure 9(b), the overlap length at the dividing point is less than the required overlap margin. A fix by layout perturbation is similarly available. We observe that the space required to increase overlap length is less than that required to remove the conflict cycle, i.e., the pattern can be split after a smaller perturbation.

In summary, node splitting handles conflict cycles in two ways: (i) splitting a node in the conflict cycle, or (ii) reporting an unresolvable conflict cycle for layout optimization to eliminate. Whenever a conflict cycle is detected in the conflict graph, it is eliminated using one of these ways. By construction, the rule-based node splitting does not cause any violation of design rules or overlap margin for the newly generated nodes. On the other hand, if any feature split in the conflict cycle will result in a violation, then an unresolvable conflict cycle is reported. Hence, the iterative conflict cycle detection and node splitting process will terminate without any conflict cycle in the graph, and we have:

Fact 2: The iterative conflict cycle detection and node splitting method obtains a conflict graph which is 2-colorable.

3.5 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. At the same time, the runtime of ILP-based coloring algorithm increases dramatically when the number of nodes in the graph is large and the runtime for solving the coloring problem in the whole conflict graph is not endurable. Therefore, we merge the nodes into small 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 the ILP-based coloring algorithm is carried out on each cluster in sequence. Because there are no edges between clusters, and no polygon has nodes in more than one cluster, the solution is obtained as the union of solutions for all the small clusters. Our layout partitioning algorithm is given in Algorithm 3. 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.

Algorithm 3 Layout partitioning algorithm.

Input: Conflict graph G and the mapping information from nodes to polygons.
Output: A set of clusters of nodes where no edges or nodes of the same polygon exist 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' corresponds to the polygons in which rectangles of u and v are respectively located;
4. **end for**
5. **while** there is an unvisited node $n'_i \in G'$ **do**
6. Make a new cluster c'_i containing n'_i;
7. Perform breadth first search from n'_i and add all visited nodes into c'_i;
8. **end while**
9. **for all** clusters c'_i **do**
10. Make cluster c_i on the nodes of the polygons in c'_i;
11. **end for**

[3] After merging the projections $P_{j,i}$ and $P_{l,i}$ horizontally, the lower point of the merged projection is the same as that of projection $P_{l,i}$. Since there is no projections that horizontally overlap with projection $P_{k,i}$, the horizontal merged projection of $P_{k,i}$ is equal to $P_{k,i}$

3.6 Min-Cost Color Assignment Problem Formulation

Finally, we have:

Min-Cost Color Assignment Problem

Given: A list of rectangles R which is color assignable, and maximum distance between two features, t, at which the color assignment is constrained.

Find: A color assignment of rectangles to minimize the total cost.

Subject to: For any two non-touching rectangles with $0 < d(i,j) \le t_{ij}$, assign different colors.

For any two touching rectangles with $d(i,j) = 0$, if they are assigned different colors, there is a corresponding cost $c_{i,j}$. $c_{i,j}$ is the cost for assigning nodes n_i and n_j different colors. We cast this as an integer linear program (ILP):

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

Subject to:

$$x_i + x_j = 1 \quad (1)$$

$$x_i - x_j \le y_{i,j} \quad (2)$$

$$x_j - x_i \le y_{i,j} \quad (3)$$

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 any pair of touching rectangles r_i and r_j. Constraint (1) specifies that non-touching 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 touching rectangles r_i and r_j are assigned different colors. The cost for touching rectangles is defined as follows.

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

where $w_{i,j}$ is width of the rectangle edge between rectangle r_i and r_j, l_i and l_j are lengths of the rectangle edges of r_i and r_j which are opposite to the touching edge, $o_{i,j}$ and $o_{j,i}$ are the minimum possible overlap lengths of nodes n_i and n_j, respectively,[4] and α, β and γ are user-defined parameters for scaling the three items for different optimization objectives. Function f is defined as follows.

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

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 design rule violations, the number of cuts on the layout polygons and maximize the overlap lengths between touching features of different colors.

Minimizing design rule violations: During the layout fracturing process, small rectangles may be generated due to specific polygonal layout features (e.g., n_5 and n_6 in Figure 10). According to Equation (4), it is easy to understand that higher costs will be assigned to pairs of touching rectangles of smaller sizes. By minimizing the total cost, the ILP-based problem formulation aims to minimize the design rule violations in the final layout.

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. Therefore, the number of cuts should be minimized to improve the quality of the decomposed layouts. In Equation (4), by setting the second term (β) to be greater than the first term, cut minimization can be given higher priority relative to consideration of design rules.

Maximizing the overlap length: Maximum overlap length satisfying the required overlap margin is also desirable. In Equation (4), a larger γ value corresponds to more emphysis on the overlap length optimization.

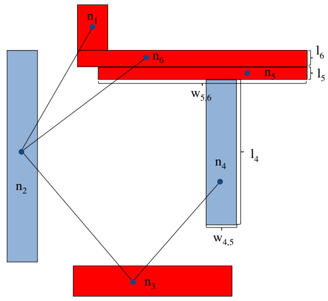


Figure 10: Example of cost function: $l_5 < l_6 < FS_{min}$, $c_{4,5} = \alpha/l_5 + \beta + \gamma/min(o_{4,5}, o_{5,4})$, $c_{5,6} = \alpha \cdot FS_{min}/(l_5 \cdot l_6) + \beta + \gamma/min(o_{5,6}, o_{6,5})$, $min(o_{4,5}, o_{5,4}) > min(o_{5,6}, o_{6,5})$, $c_{5,6} > c_{4,5}$.

Figure 10 illustrates the cost function computation, where: (1) $l_5 < l_6 < FS_{min}$; (2) $c_{4,5} = \alpha \cdot f(w_{4,5})/(f(l_4)+f(l_5))+\beta+\gamma/min(o_{4,5}, o_{5,4}) = \alpha/l_5 + \beta + \gamma/min(o_{4,5}, o_{5,4})$; (3) $c_{5,6} = \alpha \cdot f(w_{5,6})/(f(l_5)+f(l_6))+\beta+\gamma/min(o_{5,6}, o_{6,5}) = \alpha \cdot FS_{min}/(l_5 \cdot l_6)+\beta+\gamma/min(o_{5,6}, o_{6,5})$; (4) $min(o_{4,5}, o_{5,4}) > min(o_{5,6}, o_{6,5})$;[5] 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 Figure 10. Since the length of rectangle n_5, l_5, is less than the minimum feature size FS_{min}, the design rule will be violated if n_5 is assigned a different color from that of n_6. In this way, the cost function supports the minimization of design rule violations.

4. EXPERIMENTAL RESULTS

We empirically test our approach on one real-world design and three artificial designs. We evaluate our DPL solutions with respect to (1) solution quality, (2) scalability, and (3) correctness.

4.1 Experimental Setup

Table 1: Testcase parameters. Minimum spacing (140nm) and minimum line width (100nm) are scaled by 0.4x to 56nm and 40nm, respectively.

Design	#Cells	#Polygons	#Rects
AES	17304	90394	362380
TOP-A	30400	275650	1043950
TOP-B	60800	545000	2066800
TOP-C	305000	2725000	10334000

Our layout decomposition system is implemented in C++. We use one real-world design (*AES*) implemented using *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-A*, *TOP-B* and

[4]The minimum possible overlap length of two touching features is computed based on all the projections from their non-touching adjacent features. By shifting the dividing points between two touching features, the minimum overlap length can be improved, especially for those touching features split during the layout fracturing process.

[5]Note that by setting the dividing point between n_4 and n_5, the minimum overlap length between n_4 and n_5 is much larger than that between n_5 and n_6, and hence the overlap length cost between n_5 and n_6 is larger than that between n_4 and n_5.

TOP-C) 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 key parameters of the testcases. In the original 90nm layouts, minimum spacing between features is 140nm, and minimum feature size is 100nm. To reflect future nodes with smaller feature sizes, we scale GDS layout of all testcases by a factor of 0.4×, which results in 56nm minimum spacing and 40nm minimum feature size.

4.2 Experimental Results

Solution quality. We sweep the color spacing lower bound as well as placement utilization, and evaluate solution quality according to various metrics, including number of conflict cycles, number of unresolvable conflict cycles, and minimum, average and standard deviation of overlap lengths. Table 2 shows the experimental results of our layout decomposition system. In Table 2, t is the minimum coloring spacing, CCs is the number of detected conflict cycles, $uCCs$ is the number of unresolvable conflict cycles, and $Cuts$ is the number of touching rectangle pairs with different colors. (Note that we do not consider the cuts on unresolvable conflict cycles. Thus, as the number of unresolvable conflict cycles increases, the reported total number of cuts may decrease.) The minimum, average and the standard deviation of the overlap length values for all the cuts in the final decomposed layout are also reported. We furthermore verify that there are no design rule violations in the final decomposed layout.

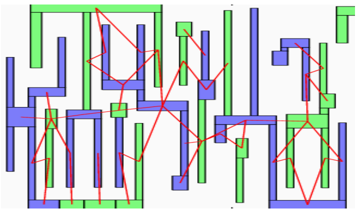


Figure 11: Example of DPL layout decomposition in the poly layer.

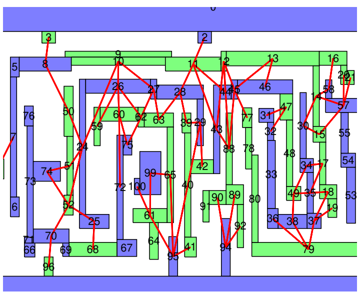


Figure 12: Example of DPL layout decomposition in the M1 layer.

From the experimental results, we see that the CC and uCC values increase as the minimum coloring spacing t increases. Also, *AES* has a smaller number of conflict cycles and unresolvable conflict cycles relative to *TOP-A*, *TOP-B* and *TOP-C*. This is because *AES* uses fewer types of cell masters, and these types of cell masters have fewer inherent unresolvable conflict cycles, while *TOP* uses many different cell masters, including some which contain more conflict and unresolved conflict cycles. Tracking the CC and uCC metrics 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 utilizations). Figures 11 and 12 show small examples of our layout decomposition solutions, where all layout is correctly decomposed with respect to the pre-specified overlap margin.

In our experiments, the overlap margin is set to 8nm, i.e., when a conflict cycle cannot be removed by node splitting with overlap length greater than 8nm, an unresolvable conflict cycle is reported. As noted above, the number of uCCs increases with t. We observe that it is costly to make layout perturbations in a post-layout processing loop, and that such layout modifications can be avoided by decreasing the value of t (e.g., $t = 60$ for AES and $t = 58$ for TOP) via smaller k_1 in lithography. From the columns under "min." and "mean", we can see that all the overlap lengths, i.e., the minimum and the average overlap lengths, in the final mask decomposition are greater than the pre-specified overlap margin (8nm margin in 45nm node [4]), which confirms the effectiveness of our layout decomposition system.

Scalability and runtime. Figure 13 compares runtime of ILP with layout partitioning, versus runtime of ILP without layout partitioning. Unsurprisingly, ILP runtime without layout partitioning increases very rapidly. On the other hand, the layout partitioning delivers much faster runtimes - e.g., over 100× speedup for the *AES* testcase compared to naive use of ILP.

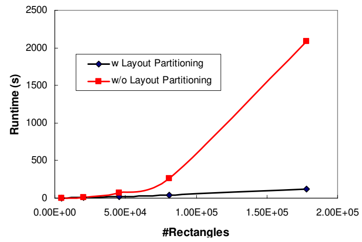


Figure 13: ILP runtime: typical vs. with layout partitioning.

Verification of DPL. 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 up 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 DRC is 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.

5. CONCLUSIONS AND ONGOING WORK

We have proposed a novel layout decomposition approach to address design needs for double exposure patterning at 45nm and below. Our approach practically and effectively improves overlap length and hence lithography yield; it has been implemented using a conflict graph infrastructure and ILP solver. 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), and that all unresolvable conflict cycles are reported, confirming the effectiveness of our approach.

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 are introducing 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).
- We are also investigating the possibility of integrating all key optimization objectives, including the number of unresolvable conflict cycles, the number of line-ends, the overlap lengths between touching rectangles, etc., into a unified ILP problem formulation.
- Besides the ILP-based approach, we are also investigating combinatorial frameworks, including node- and edge-deletion based graph bipartizations, that offer potentially attractive runtime-quality tradeoffs.

Table 2: Experimental results of layout decomposition system (four testcases, 70% and 90% utilization). t: minimum coloring spacing (nm). CCs: number of conflict cycles detected. $uCCs$: number of unresolvable conflict cycles. $Cuts$: 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	CCs	uCCs	Cuts	min.	mean	σ	CPU
AES (70%)	58	0	0	29	44	237.7	149.2	17.6
	59	0	0	33	56	111.5	58.6	19.1
	60	0	0	33	56	110.6	58.6	18.0
	61	53	1	132	18	55.6	48.7	19.9
TOP-A (70%)	58	326	0	6261	11	145.2	81.1	136.7
	59	503	150	8013	9	140.4	77.8	145.8
	60	503	400	7884	22	143.1	75.9	149.7
	61	7704	3230	25818	11	95.8	75.9	224.4
TOP-B (70%)	58	654	0	12502	11	145.0	81.0	488.5
	59	1004	300	15997	9	134.0	77.7	494.4
	60	1004	800	15752	22	142.9	75.7	450.1
	61	15175	6356	51040	11	96.3	76.0	716.9
TOP-C (70%)	58	4040	0	63212	13	139.1	77.5	4611
	59	5536	1500	80472	9	128.7	69.9	4566
	60	5536	4000	78720	22	131.2	68.0	4535
	61	78696	31677	243853	11	94.6	75.6	5097
AES (90%)	58	0	0	29	44	184.8	146.7	17.8
	59	0	0	33	56	130.3	53.9	17.7
	60	0	0	33	56	129.4	53.8	17.8
	61	57	1	132	18	59.7	52.7	20.6
TOP-A (90%)	58	325	0	6291	11	145.1	80.9	135.1
	59	503	150	8025	9	140.9	77.7	143.7
	60	503	400	7903	22	143.5	75.5	155.0
	61	7709	3243	25825	11	95.7	75.8	227.5
TOP-B (90%)	58	648	0	12502	11	145.2	81.1	444.7
	59	999	300	15990	9	134.0	77.8	441.9
	60	999	800	15755	22	142.9	75.9	500.2
	61	15170	6356	51054	11	96.3	76.0	601.4
TOP-C (90%)	58	3994	0	63219	13	139.0	77.6	4328
	59	5501	1500	80440	9	128.8	70.2	4562
	60	5501	4000	78709	22	131.3	68.3	4655
	61	78754	31743	243695	11	94.5	75.7	4623

6. 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] 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.

[7] C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Zelikovsky, "Bright-Field AAPSM Conflict Detection and Correction", *Proc. DATE*, 2005, pp. 908-913.

[8] C. Chiang, A. B. Kahng, S. Sinha and X. Xu, "Fast and Efficient Phase Conflict Detection and Correction in Standard-Cell Layouts", *Proc. ICCAD*, 2005, pp. 149-156.

[9] 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.

[10] 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.

[11] J. Finders, M. Dusa and S. Hsu, "Double Patterning Lithography: The Bridge Between Low k1 ArF and EUV", *Microlithography World*, Feb. 2008.

[12] http://en.wikipedia.org/wiki/Hardmask.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.