Layout modification for library cell Alt-PSM composability
Ke Cao, Jiang Hu, Mosong Cheng
Dept of Electrical Engineering, Texas A&M University, College Station, TX 77843

ABSTRACT

In sub-wavelength lithography, light field Alt-PSM (Alternating Phase Shifting Mask) is an essential technology for poly layer printability. In a standard cell based design, the problem of obtaining Alt-PSM compliance for an individual cell layout has been solved well [3]. However, placing Alt-PSM compliant cells together can not guarantee Alt-PSM compliance of the entire chip/block layout due to phase interactions among adjacent cells. A simple solution to this Alt-PSM composability problem is to wrap blank area around each cell, which is very inefficient on chip area usage. In this paper, we formulate the composability problem as a graph model and propose a polynomial time optimal algorithm to achieve Alt-PSM composability with the least impact on cell layout.

1. INTRODUCTION

As VLSI technology feature size shrinks to sub-wavelength regime, Resolution Enhancement Techniques (RET), such as Phase-Shifting Masks (PSM) and Optical Proximity Correction (OPC), become essential to achieve lithography printability.

The concept of PSM is first proposed by Levenson et al. [1] in 1982 (See figure 1). In PSM, lights of opposite phases are projected to two sides of a feature so that the refractive lights of opposite phases cancel out each other at the region of feature and a sharper image is printed. The 180° light region is called phase shifter. It is common that two features are in phase conflict, i.e. the phase assignment of one feature depends on that of the other, like feature 1 and feature 4 in Figure 2, if we put the shifter at the right side of the feature 1, the shifter of the feature 4 has to be at the left side of it. For certain layout configurations, it is impossible to find a shifter assignment to satisfy Alt-PSM rule for all features, thus we say phase error exists in the layout. For the example in Figure 2, region F should be assigned 0° phase for the bottom feature 3 and 180° for the right feature 4. Simultaneous assignment of opposite phase to a same region is impossible, and a phase error occurs.

Several approaches have been proposed to solve the problem of phase error of Alt-PSM. Berman and Kahng[3] applied an optimal bipartization for phase error removal. They identified the features involved with phase errors and the...
minimum change necessary to fix the errors. Pierrat and Cote [4] proposed a mask conversion methodology to remove phase error. They split the shifter region whenever necessary to avoid phase error during the mask printing process. Although the first approach provides a good solution for cell/block level layout, it is not efficient to apply it directly to entire chip level layout in a cell based design. The latter approach has the advantage that no layout change is necessary, but it intensifies the mask complexity.

Figure 2. Example of Phase Error: same phase region at both side of feature 3 (region F). There is no phase assignment that can remove all the phase errors of these four features.

Figure 3. Concept of cell composability: (a). New phase error occurs when two cells are placed adjacent to each other. (b). No phase error is created for these two cells if cell 2 is processed to be composable.

In a cell based design, a specific cell may be utilized several times in different places of a chip. Applying Alt-PSM compliance algorithm such as [3] on all features on the entire chip simultaneously would make repeated corrections for a specific cell on each of its occurrence. A more efficient approach is to correct each cell layout for Alt-PSM compliance only once and utilize it as many times as we want without repeated corrections. Unfortunately, placing Alt-PSM compliant cells next to each other can not guarantee Alt-PSM compliance of entire layout, as interactions between features from adjacent cells may cause new phase errors, i.e., the cells may not be Alt-PSM composable. The
composability problem is illustrated in Figure 3. If the newly occurred phase errors are removed by employing a correction algorithm like [3] flatly over entire chip, the efforts of phase correction for each individual cell are nullified. Of course, the inter-cell phase errors can be avoided by simply enforcing extra blank space between cells. However, increasing space between every pair of adjacent cells may cause unnecessary area waste while increasing space for only those with inter cell phase errors still requests a global level Alt-PSM compliance check and correction. In order to avoid such dilemma, we propose to achieve cell Alt-PSM composability without uniformly wrapping blank area around each cell layout. With the Alt-PSM composability, cells can be placed next to each other without worrying any inter-cell phase errors.

The paper is organized as follows: section 2 presents the problem formulation, section 3 gives the algorithm, section 4 provides results for the algorithm and section 5 provides some discussion and conclusion.

2. PROBLEM FORMULATION

Since any phase error can be removed by applying the algorithm of [3], we can assume that every cell layout has no phase error inside. If we can assign the same phase to the boundary regions of a cell without creating phase errors in the cell, we would achieve cell Alt-PSM composability. For example, if we can assign 0° phase to all four sides of every cell layout, any two cells can be placed adjacent to each other with 0° phase along their boundary and no phase error may occur. Depending on the design methodology, we can also make the cells to have the same phase assignment on left and right boundary, or upper and lower boundary, or any desired boundary combinations. Thus, the problem of obtaining cell composability is formulated as:

Given a cell layout without phase errors, determine if this cell is Alt-PSM composable. If not, determine the minimum change of the layout in order to make it composable.

As in previous work of Alt-PSM [6], the relation of $b \leq B$ holds, where $b$ is the minimum spacing between two features according to design rules, and $B$ is the minimum spacing for features not to have phase conflict. It is also commonly true that $B < 2b$.

The basic approach to correct a phase error in a layout is to increase the spacing between two features with phase conflict such that their distance is greater or equal to $B$. The phase conflict graph is constructed according to [5] (figure 4). In the graph, each feature is a feature node, the shifter of the feature is a shifter node, and if two features are in phase conflict, there will be a conflict node representing this conflict. There exists an edge between a feature node and its associated shifter node, the shifter node and the conflict node. Also an edge between a feature node and the conflict node is added if there is no shifter node in that side of the feature. In case one feature is in phase conflict with multiple features, each phase conflict will be represented by a different conflict node and a different shifter node if the conflict occurs at the shifter side of the feature. The work of [5] has proved that this graph is planar. The layout is phase error free if the phase conflict graph is a bi-partite graph [5], i.e., there is no odd cycle in the graph.

We define critical boundaries to be the boundaries of a cell that need the same phase for composability. Critical boundaries can be specified by designers. In a conflict graph, for every feature node whose location is within a certain distance $B$ to any critical boundary of the cell, we define another type of node - boundary node, it is either the feature node if the shifter of the feature node is pointing into the cell, or the shifter node of the feature otherwise. For example, assuming the phase conflict graph looks like figure 5, if critical boundaries are the four boundaries of the cell, node $N_1$-$N_7$ are boundary nodes. We further define the odd path in a graph as a path composed by odd number of edges. For example, the path between $N_2$ and $N_1$ in figure 5 is an odd path.

Obviously, only boundary nodes need to be considered for potential phase error when two cells are placed next to each other. Also, in the phase conflict graph, we can not assign the same phase along the critical boundaries if and only if the there exists an odd path between any boundary nodes. Note that if there are multiple paths between two feature nodes, either all of them are odd paths or none of them is an odd path, since there is no odd cycle in a conflict graph for a cell layout without phase error inside. Thus, the problem of cell composability is further reduced to:

Given a phase conflict graph of a cell layout without phase errors, determine if odd path between any two boundary nodes exists. If so, determine a minimum change of layout that can remove such odd paths.
3. ALGORITHM

For every edge in the phase conflict graph, we define its weight as follows: For an edge that has a conflict node incident on it, if we traverse this edge in both directions, we will reach two feature nodes in phase conflict, we define the weight of the edge is the silicon area increment if we have to increase the spacing between those two features to remove the phase conflict. For all other edges, the weight is infinite as removing them does not have any actual meaning in silicon implementation.

Since the original cell layout does not have any phase errors, the corresponding phase conflict graph is bi-partite, which means we can color the graph with two colors. Then, the nodes can be partitioned into two sets L and R, with all the edges between L and R. For any node x in L and node y in R, if they are connected in the conflict graph, the path from x to y is an odd path. For any two nodes x and y in the same set, the path from x to y can not be an odd path. Hence, the composability problem becomes to find the minimum edge weight cut to separate boundary nodes that belong to

Figure 4. Phase Conflict Graph for four features.

Figure 5. Cell composability problem is equivalent to removing odd paths between “boundary nodes” in a conflict graph.

For every edge in the phase conflict graph, we define its weight as follows: For an edge that has a conflict node incident on it, if we traverse this edge in both directions, we will reach two feature nodes in phase conflict, we define the weight of the edge is the silicon area increment if we have to increase the spacing between those two features to remove the phase conflict. For all other edges, the weight is infinite as removing them does not have any actual meaning in silicon implementation.
different sets. In Figure 6, the solid dots indicate boundary nodes and the empty nodes indicate all other nodes. We use the network flow model to separate the set of boundary nodes in subset L and those in subset R by introducing a source node S and a sink node T. We also add edges between S and all the boundary nodes in subset L and edges between T and all the boundary nodes in subset R, with all the added edges having infinite edge weights. We then search for a minimum weight cut of S and T, the result will give us the minimum cut of the boundary nodes in L and R, which is the same as the minimum change of the layout for composability.

The optimality of this algorithm is obvious. If we cut all edges with finite weight, we can separate S from T, which implies that the minimum cut of S and T has to be finite, and each edge that has been cut has a conflict node incident on it since the weight of the edge is infinite otherwise. Because all the cut that we make are necessary and the weight of the cut is minimized, this algorithm is optimal for the cell. Note that the cut could also include an edge that has no boundary node incident on it, but it has to have a conflict node incident on it.

We analyze the complexity of this algorithm as follows: the generation of the conflict graph has the complexity of O(n log n) as demonstrated in [5], with n being the number of features. The partitioning of the nodes into L and R sets has the complexity of O(n) as each feature needs to be processed only once. We use the Minimum Capacity Cut algorithm [7] as the S-T minimum cut algorithm, the capacity of each edge is the weight of the edge. This minimum cut algorithm takes time O(n^3). Therefore, the complexity of our algorithm is O(n^3).

4. EXPERIMENT RESULTS

We ran the algorithm on two test cells, a XOR gate and a flip-flop. Both test cells are Alt-PSM compliant before applying our algorithm. Figure 7 shows the layout of the XOR gate and it corresponding phase conflict graph and the change that our algorithm makes to the cell. Table 1 shows the area increment comparison of our algorithm and the method of applying blank area around the cells. We used the minimum poly spacing b = 0.42 unit length and phase conflict spacing B = 0.78 unit length.
Table 1. Area increment of cell composability process

<table>
<thead>
<tr>
<th>Cell</th>
<th>Number of gates</th>
<th>Area before processing(Unit²)</th>
<th>Area increment with composable processing(Unit²)</th>
<th>Area increment with blank insertion (Unit²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>12</td>
<td>84.82</td>
<td>6.89</td>
<td>20.26</td>
</tr>
<tr>
<td>Flip-flop</td>
<td>22</td>
<td>203.43</td>
<td>3.67</td>
<td>13.57</td>
</tr>
</tbody>
</table>

The comparison shows area increment savings of our approach. For the library cells that will be placed into the design for multiple times, the total area saving for Alt-PSM designs will be significant.

5. CONCLUSION

We proposed an optimal algorithm with polynomial complexity for cell layout composability of Alt-PSM and this algorithm provides a significant silicon area saving over the previous method. The global Alt-PSM compliance of poly layer is guaranteed when cells are placed next to each other after being processed with our method. Future work includes hierarchical algorithm for Alt-PSM for metal layers with composable cells, fast algorithm to achieve Alt-PSM compliant and composable simultaneously.

6. ACKNOWLEDGEMENT

The authors would like to thank Prof. Weiping Shi for helpful discussion of the algorithm, Dr. Robert Maziasz, Puneet Dhawan and Xiang Lu for the help on preparing the test cells.

REFERENCE

Figure 7. Composability process. (a). Original XOR design. (b). Poly topology of original design. (c). Phase conflict graph of original design. (d). Phase conflict graph after composability process. (e). Poly topology for the composable cell. (f). Composable layout cell of the XOR gate.