Automated Layout and Phase Assignment Techniques for Dark Field Alternating PSM

Andrew B. Kahng, Huijuan Wang and Alex Zelikovsky

UCLA Department of Computer Science, Los Angeles, CA 90095-1596
{abk, huijuan, alexz } @cs.ucla.edu
http://vlsicad.cs.ucla.edu

ABSTRACT

We describe new, efficient algorithms for layout modification and phase assignment for dark field alternating-type phase-shifting masks in the single-exposure regime. We make the following contributions. First, we give optimal and fast algorithms to minimize the number of phase conflicts that must be removed to ensure 2-colorability of the conflict graph. These methods can potentially reduce runtime and/or improve solution quality, compared to previous approaches of Moniwa et al. and Ooi et al. Second, we suggest a new iterative 2-coloring and compaction approach that simultaneously optimizes layout and phase assignment. The approach iteratively performs the following steps: (i) compact the layout and find the conflict graph; (ii) find the minimum set of edges whose deletion makes the conflict graph bipartite; and (iii) add a new compaction constraint for each edge in this minimum set, such that the corresponding pair of features will no longer conflict. Third, we describe additional approaches to co-optimization of layout and phase assignment for alternating PSM. Preliminary computational experience appears promising.

Keywords: Layout verification, manufacturability, alternating phase-shift mask (PSM), compaction, physical design, yield enhancement

1. INTRODUCTION

Phase-shifting mask (PSM) technology enables the clear regions of a mask to transmit light with prescribed phase shift. Consider two adjacent clear regions with small separation, and respective phase shifts of 0 and 180 degrees. Light diffracted into the nominally dark region between the clear regions will interfere destructively; the improved image contrast leads to better resolution and depth of focus. All PSM variants employ this basic concept, which was proposed by Levenson et al.[6] in 1982 (see Figure 1). Our work, like that of Moniwa et al.[8,9] and Ooi et al.,[11,12] pertains to the *dark field*, *alternating* (or *Levenson-type*) phase-shifting mask technology with negative photoresist. In particular, we seek methods compatible with *single-exposure* alternating PSM.

As in previous work,[11] we use the positive constants $b < B$ to define a simplified relationship between printability and the distance between two clear regions. The distance between any two features cannot be smaller than b without violating the minimum spacing design rule. If the distance between two features is at least b but smaller than B, the features are in *phase conflict*.* Phase conflict can be resolved by assigning opposite phases to the conflicting features. In other words, B defines the minimum spacing rule when two features have the same phase. If the distance between two features is greater than B, there is no phase conflict and the features can be assigned arbitrary phases. Note that the values of b and B are layer-dependent. We also let $w \geq b$ denote the minimum allowed width of any feature on the layer of interest. Finally, we assume that all features are rectilinearly oriented (axis-parallel) polygons.

The Phase Assignment Problem: Given a layout, assign phases to all features such that no two conflicting features are assigned the same phase.

Research at UCLA was supported by a grant from Cadence Design Systems, Inc.

*More precisely, two features are in phase conflict if (i) there is no pair of points, one from each feature, whose separation is less than b; and (ii) there is some pair of points, one from each feature, whose separation is less than B.

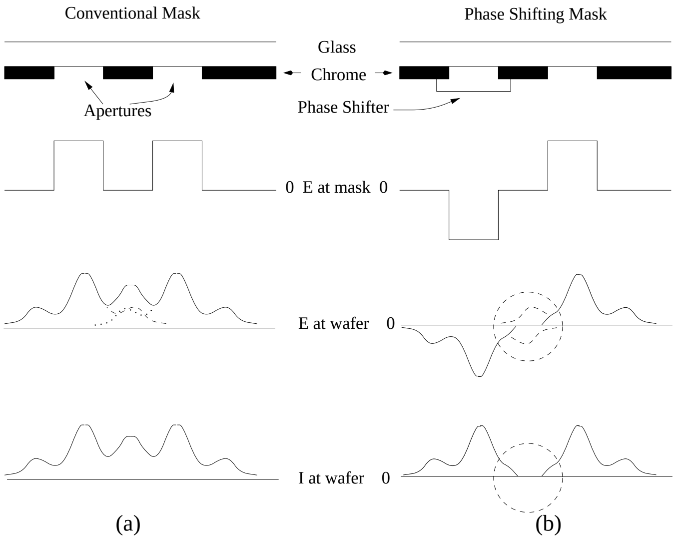

Figure 1. Comparison of diffraction optics of conventional and phase-shifting masks. E denotes electric field and I denotes intensity. With the conventional mask (a) light diffracted by two adjacent apertures constructively interferes, increasing the light intensity in the dark area of the wafer between the apertures. With the phase-shifting mask (b), the phase shifter reverses the sign of the electric field, and destructive interference minimizes light intensity at the wafer in the dark area between apertures.

1.1. The Conflict Graph

For a given layout of polygonal features, the *conflict graph* $G = (V, E)$ is constructed by defining a vertex for each feature, and introducing an edge between two vertices exactly when the corresponding features are in phase conflict. The conflict graph can be constructed in $O(n \log n)$ time, where n is the total number of segments in all polygon boundaries.†

In alternating PSM, we can remove all phase conflict by assigning opposite phases to each pair of adjacent vertices in the conflict graph G. This is equivalent to 2-coloring the vertices of G with phase 0 and phase 180. For this to be possible, G must be bipartite, i.e., have no odd cycles. Hence, if the conflict graph G is not bipartite, our goal is to *delete* enough edges such that no odd cycles exist in the remaining modified conflict graph. Edge deletion in the conflict graph is achieved by changing the placement of layout features so that they no longer conflict. Efficient algorithms that we give below for removing odd cycles depend on *planarity* of G, an assumption that we now justify.

Theorem 1. Assume that (i) the minimum feature size w satisfies $w \geq b$, (ii) $2b \geq B$ and (iii) four rectangles which are pairwise (with the possible exception of one pair) in conflict do not have diagonal conflicts (see Figure 2(a)). Then, the conflict graph G is planar.

Proof sketch. For each feature, locate a representative node arbitrarily within this feature. For any two features in conflict choose a pair of closest points on their boundaries and connect each of these points to the other and the the representative node of its respective feature (see Figure 2(b)). In other words, each conflict edge is subdivided into two *connections* inside features and an *outside segment* between features. This subdivision does not affect planarity. For each feature, the connections between its representative node and all points on the boundary can be routed

†A standard construction is as follows.

- Slice each feature (polygon) into rectangles by vertical cuts through all polygon vertices, maintaining a pointer from each rectangle to its containing polygon.
- Bloat each rectangle by distance $B/2$.
- Using sweepline and interval trees, detect conflicts between polygons by finding overlapping pairs of rectangles that belong to different polygons.

Alternatively, one may detect intersections of *bounding boxes* of polygons, then check whether the corresponding polygons actually intersect.

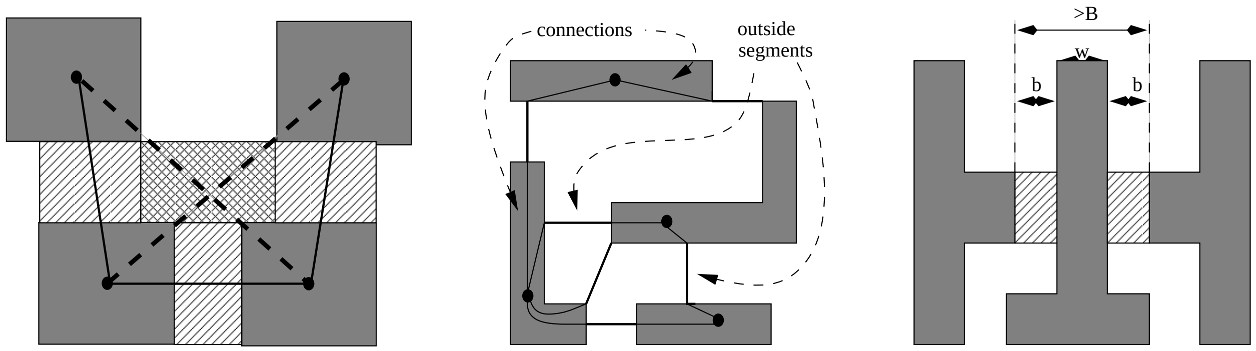

Figure 2. (a) We assume that there are no conflicts between diagonal pairs (dashed edges) in a set of four features if at least three other conflicts exist. (b) The subdivision of edges of the conflict graph. (c) There is no conflict between the left feature and the right feature because $B \leq 2b$.

without intersections. Any outside segment is shorter than B, therefore, no outside segment can go into a feature and then leave it because it would then have length at least $2b \geq B$ (see Figure 2(c)). Thus if two conflict edges intersect, their outside segments intersect outside of features.

Suppose that (a, b) and (c, d) are two intersecting outside segments. Assume first that two points, say a and b, belong to the same feature. Because (a, b) and (c, d) intersect, $|a, b| + |c, d| > |a, d| + |b, c|$. Therefore, either (a, b) is longer than (a, d) or (c, d) is longer than (c, b), which contradicts the definition of the outside segment as a closest pair. If all four points belong to different features, then it can be shown that three pairwise distances other than (a, b) and (c, d) between these four points are smaller than B, which contradicts assumption (iii). □

1.2. Outline of the Paper

The remainder of this paper is organized as follows. In Section 2, we review previous approaches to edge deletion in conflict graphs, and then suggest several improvements as well as new approaches. In Section 3 we describe an optimal algorithm for odd cycle elimination; we also describe a fast algorithm based on the *Voronoi graph* concept. Section 4 describes the implementation of two proposed methods. We conclude with computational experience and directions for future work.

2. APPROACHES TO REMOVING ODD CYCLES FROM THE CONFLICT GRAPH

2.1. Previous Work

Moniwa et al.[8] and Ooi et al.[11] first posed the phase assignment problem and suggested methods of detecting cases when there is no valid phase assignment, i.e., when the conflict graph contains odd cycles. Subsequently, Ooi et al.[12] and Moniwa et al.[9] suggested interactive methods which fully exploit information in the mask layout. The heuristic of Moniwa et al.[9] first constructs the conflict graph G, then creates a list of all odd cycles in G using an enumerative approach. The heuristic then iteratively finds and deletes the edge that is in the greatest number of minimum-length odd cycles. Deletion is accomplished by increasing the lower bound on separation between the corresponding features, and then applying a compactor to perturb the shape or position of these features (see Figure 3). This approach may be feasible for the cell layout editing context, but likely does not scale to large instances since the number of odd cycles can be exponential in the size of the layout. Moreover, the heuristic does not necessarily delete the minimum number of edges, nor will it necessarily select edges whose deletion will have minimum impact on the layout.

Ooi et al.[12] also suggested a compaction-based method which (i) produces a symbolic layout from the mask layout; (ii) performs phase assignment in the symbolic layout; and (iii) compacts the symbolic layout using minimum spacing design rules consistent with the phase assignment. The advantage of the compaction-based method is that it is fully automated and guarantees to remove all odd cycles from the conflict graph. On the other hand, the phase assignment step is relatively oblivious to details of the mask layout; the ensuing compaction step may not minimize distortion of the original layout.

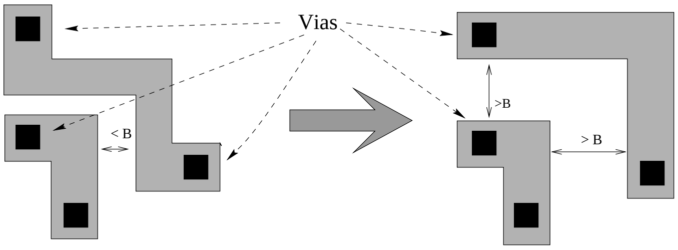

Figure 3. Changing a shape without changing positions of vias.

2.2. Proposed Methods

We propose the following approaches to layout modification and phase assignment for alternating PSM.

Iterative coloring and compaction. Our first new approach generalizes the approaches of Ooi et al.[12] and Moniwa et al.[9] Initially, we constrain the layout only by the minimum-spacing design rule, i.e., no two features can be less than distance b apart. We then *iteratively* apply the following three steps until the conflict graph G becomes bipartite:

(i) compact the layout and find the conflict graph G;

(ii) find the minimum set of edges whose deletion makes the conflict graph G 2-colorable; and

(iii) add a new constraint for each edge in this minimum set, such that the pair of features connected by this edge must be separated by distance at least B;

"One-shot" phase assignment. This is an improvement over the method in Ooi et al.[12] It consists of the following steps: (i) find the conflict graph G; (ii) find the minimum set of edges whose deletion makes the conflict graph G 2-colorable; (iii) assign phases such that only the conflict edges in the minimum set connect features of the same phase; and (iv) compact the layout with "PSM design rules", i.e., minimum separation B between features that are assigned the same phase, and minimum separation b between features that are assigned different phases.

Splitting. Our third proposed method "splits" sufficiently long features into several parts; this is equivalent to splitting one vertex into several parts in the conflict graph. Introducing *partial shifters* between these parts allows gradual transitions between 0-phase and 180-phase with no dark edge being formed. Such approaches are used for bright field PSM with positive photoresists. In the context of dark field with negative photoresists, splitting allows us to assign different phases to neighboring parts within the same feature.[‡] Notice that if we assign the same phase to neighboring parts of a given feature, we save the cost of the partial shifters between the parts, since no transition is needed. The splitting technique requires a more complicated mask, but has two advantages: (i) it can only decrease the number of odd cycles, and (ii) it does not perturb the layout geometry. Figure 4 shows how splitting can eliminate odd cycles. Splitting can be implemented via an optional preprocessing of G that splits vertices corresponding to long features into several disconnected vertices.

Spacing. Our next suggestion is to widen the layout at selected coordinates, by inserting vertical or horizontal gaps along the whole layout (see Figure 5). Inserting a gap has the effect of deleting all edges of the conflict graph G that cross the gap. We observe that if one has already identified a set of edge deletions that will remove all odd cycles from G, we can find the minimum number of spacings that will delete these edges. The entire layout can be represented as a 0-1 matrix in which 1's are placed in all the conflict areas corresponding to desired edge deletions. Each column or row in the matrix corresponds to a gap to be introduced. We can find the minimum number of columns and rows that cover all 1's by applying an $O(n^{1.5})$ algorithm for minimum matching in bipartite graphs.

‡Partial shifter technology that allows 60-, 120- or 180-degree phase shifting has been reported by AMD.[10] The technology is "free" in the sense that 180-degree phase shifters are in any case built out of several layers of 60-degree phase shifters. Such a mask technology reduces defects because it imposes a "voting" regime: a defect that causes a phase edge or otherwise affects the printed image must occur at the same location in two out of the three 60-degree layers.

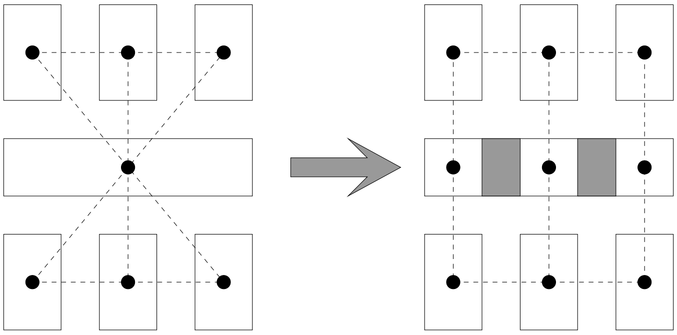

Figure 4. Splitting a long feature may result in odd cycle elimination. The phase is shifted between 0 and 180 degrees in the shaded areas, using partial shifter techniques.

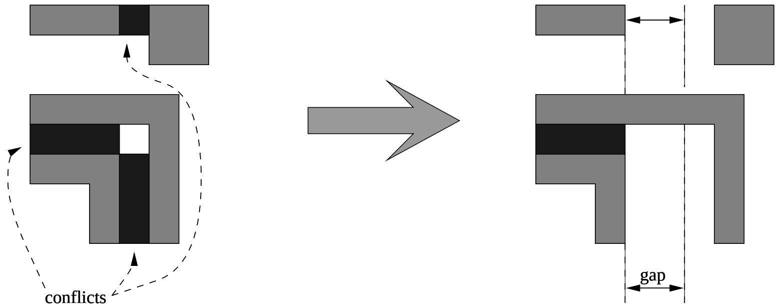

Figure 5. Resolving multiple conflicts between features by introducing a gap.

Layer assignment. Finally, we suggest the use of layer assignment to remove phase conflicts, as shown in Figure 6. This is a key degree of freedom that is natural for a routing tool, but unnatural for current mask optimization tools that view a given layer's geometries as immutable. With layer assignment, a problematic *wire* feature can, quite literally, be replaced by two vias and then transferred to another layer. Of course, such an approach does not apply to device layers.

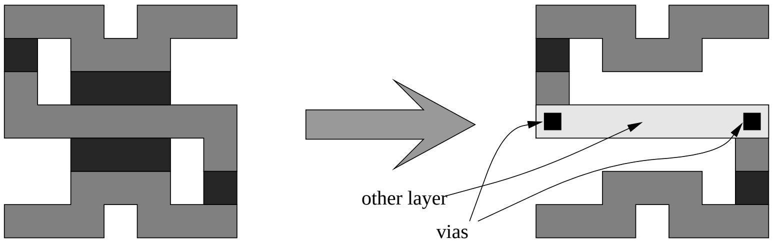

Figure 6. Resolving conflicts by moving the middle wire to another layer.

2.3. Edge Weights in The Conflict Graph

In general, when an optimal phase assignment is found, each conflict edge that separates two same-phase features induces a minimum spacing requirement of B between the features. Such a requirement is passed to compaction in the form of a spacing constraint. To fully exploit compaction technology and achieve optimal algorithms for phase assignment with minimum layout distortion, we may assign different weights to different conflict edges. With the methods for optimal odd cycle removal that we develop below, even if the conflict edges have different weights it is still possible to find the minimum-weight set of conflict edges whose deletion makes the conflict graph bipartite. During compaction we may detect spacing constraints (between feature pairs) that are on *critical paths*, i.e., constraints that when increased will directly increase the size of the layout. We propose to assign larger weight to conflict edges corresponding to such constraints; lesser weight should be assigned to conflict edges that do not lie on critical paths (see Figure 7). Finally, we observe that several methods to delete edges and eliminate odd cycles can be combined. Then, the weight of a given conflict edge should reflect the minimum possible cost of breaking that edge using any of the available methods.

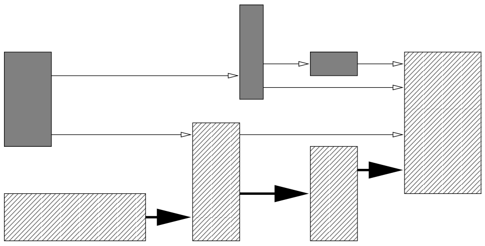

Figure 7. Critical path between leftmost and rightmost features consists of thick edges. Thin edges on non-critical paths may be broken for free.

3. OPTIMAL AND FAST ALGORITHMS FOR ODD CYCLE ELIMINATION IN CONFLICT GRAPHS

We now present optimal and fast algorithms for finding the minimum-cost set of conflict edges whose deletion eliminates all odd cycles.

The Minimum Distortion Problem: Given a planar graph $G = (V, E)$ with weighted (multiple) edges, find the minimum weight edge set M such that the graph $(V, E - M)$ contains no odd cycles.

3.1. Optimal Algorithm

Hadlock[2] and Orlova et al.[13] proposed the first exact algorithm for odd cycle elimination.

Theorem 2.[2,13] The Minimum Distortion Problem can be solved optimally in polynomial time.

Proof sketch. To eliminate all odd cycles it is sufficient to eliminate odd faces of the planar graph G (see Figure 8). Consider a graph D that is the geometric dual of G: all faces of G are vertices of D (note that there is a vertex of D that corresponds to the "outer face" of G), and each edge of D intersects with exactly one edge of G. The odd faces of G correspond to odd-degree vertices of D. Any edge elimination in G corresponds to edge contraction in D; in particular, we may remove a pair of odd faces from G by contraction edges of a (minimum-weight) path between the corresponding odd-degree vertices in D. Hence, the minimum distortion set corresponds to minimum weight matching of odd-degree vertices in D, which can be found in polynomial time. □

3.2. Fast Algorithm

The main drawback of the Hadlock and Orlova et al. algorithm is that finding all pairwise distances between odd-degree vertices in D is too time- and memory-consuming. We now propose a new fast approach based on the Hadlock and Orlova et al. algorithm, but modified using the Voronoi graph paradigm of Mehlhorn.[7] It consists of the following steps:

1. Build a graph G' which is the geometric dual of G. Each vertex in G' corresponds to a face in G. Each edge e' between two vertices of G' corresponds to (i.e., "crosses") the edge e that separates the two corresponding faces in G. The weight of an edge e' in G' is equal to the weight of the corresponding edge e in G.

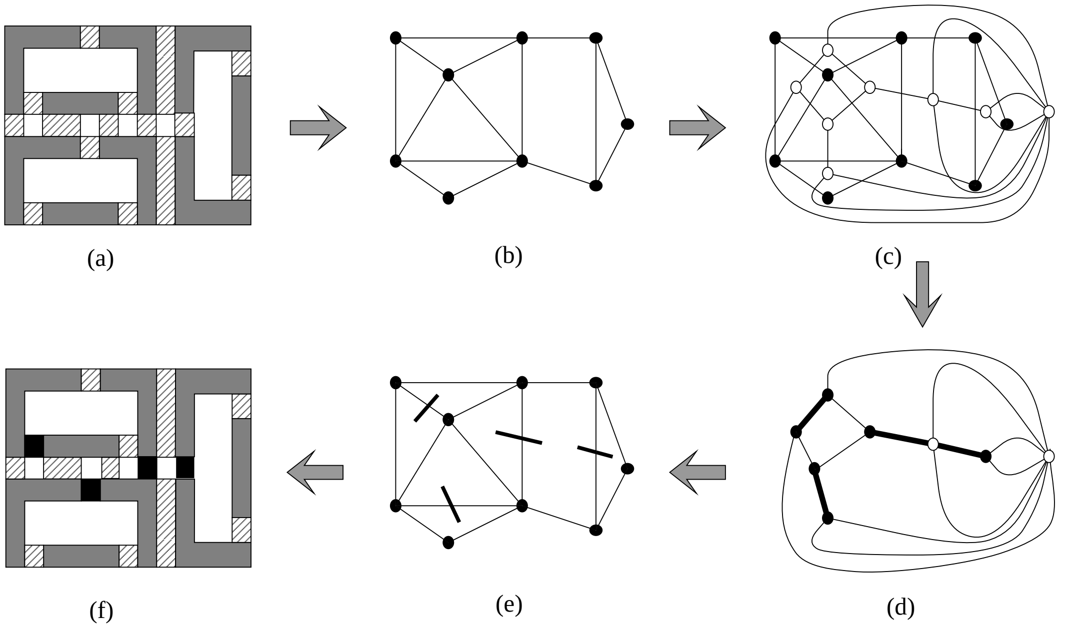

Figure 8. From the conflicts between features (a), the conflict graph is derived (b). The dual graph (c) is constructed. The vertices of odd degree are matched using paths in the dual graph (d), and the corresponding conflict edges are determined (e). Finally, the minimum set of conflicts to be deleted is determined (f).

2. Observe that faces of G that are defined by an odd number of edges correspond to the odd-degree vertices of G'. We use Odd to denote this set of vertices in G'. Partition all vertices of G' into the *Voronoi regions*[7] of the vertices in Odd, such that:

   (i) each Voronoi region contains exactly one vertex from Odd, which we call the *center* of the Voronoi region, and

   (ii) each even-degree vertex belongs to the Voronoi region whose center is the closest according to shortest-path distance (break ties arbitrarily).

3. Construct the *Voronoi graph* $Vor(G')$ in which each vertex represents a Voronoi region, and two vertices R_1 and R_2 are adjacent if a shortest path in G' between the corresponding region centers is completely contained in the two regions (see Figure 9).

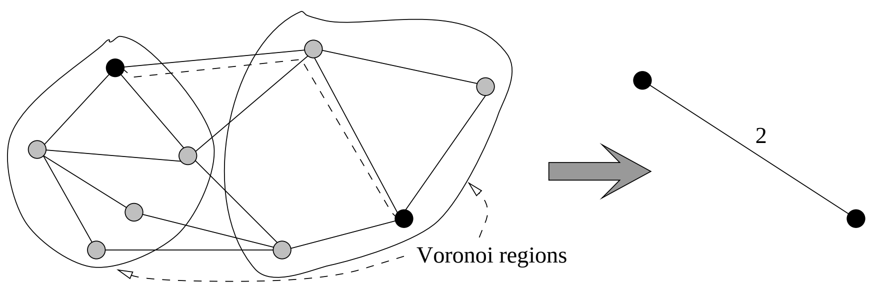

Figure 9. The geometric dual graph G' (left) and the Voronoi graph of its odd-degree vertices (right). The weight of an edge in $Vor(G')$ is the total cost of edges in the shortest path (dashed edges) between the centers of the Voronoi regions (black vertices).

4. Find a minimum weighted matching in $Vor(G')$.

5. Find edge sets in G that correspond to *paths* in G', which in turn correspond to the edges of the minimum weighted matching in $Vor(G')$. These are the edges to be deleted in the conflict graph.

Step 1 can be performed in time $O(n)$, where n is the number of features. Construction of $Vor(G')$ requires time $O(n \log n)$. Step 4 requires time $O(|Odd|^2))$, and Step 5 can be done in time $O(n)$. Thus, the total runtime of our algorithm is $O(n \log n + |Odd|^2)$.

4. IMPLEMENTATION EXPERIENCE

We have implemented the "one-shot" and "iterative" methods from Section 2.2; other proposed methods are in earlier stages of investigation. The flows for the one-shot and iterative methods are shown in Figure 10.

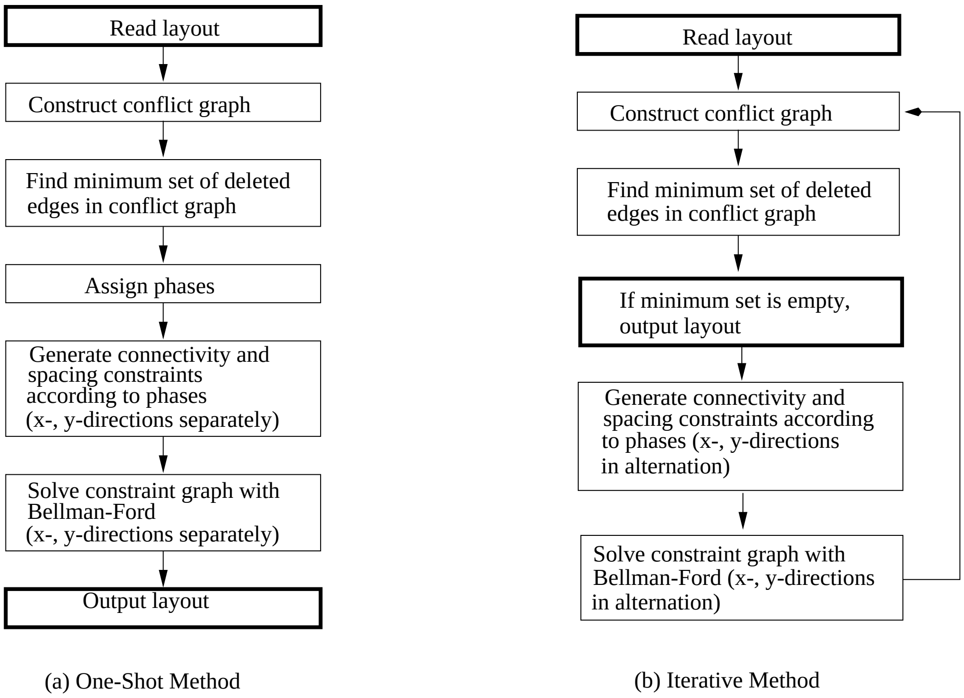

Figure 10. Flows for the "one-shot" and iterative methods. Note that the one-shot approach applies compaction separately in the x- and y-directions to enforce the given phase assignment. The iterative approach alternates between x- and y-compaction, and recomputes the minimum set of deleted conflict edges (and resulting compaction constraints) after each compaction.

4.1. Software Implementation

Our system has been implemented in C++ on the Solaris 2.6, Sun CC 4.2 platform. Input is (hierarchical) GDSII that is converted to CIF, then read into an internal polygon database. There are two major software elements.

- The *phase generator* finds a minimum-cost set of conflict edges for deletion, induces a phase assignment, and generates compaction constraints such that the layout remains consistent with the phase assignment. The code for fast minimum-cost odd cycle elimination uses an imported C code for maximum weighted matching in a general graph, implementing the algorithm of Gabow (1973) and written by Edward Rothberg, which we obtained by anonymous ftp from dimacs.rutgers.edu.

In the one-shot flow, the phase generator creates compaction constraints as follows: if two features are assigned different phases, they have minimum separation b; otherwise, they have minimum separation B. In the iterative flow, constraints are specified only for pairs of features corresponding to broken conflict edges.

- The *graph-based compactor* (i) generates constraints between feature edges according to standard design rules and an efficient sweepline approach, (ii) adds constraints produced by the phase generator, and (iii) stores all constraints as edges of a constraint graph. The compactor then applies the Bellman-Ford algorithm to solve the constraint graph, i.e., obtain optimal x-coordinates of all edges of all features. Our implementation generally follows the description of leaf-cell compaction given by Bamji and Varadarajan.[1] We consider three types of constraints: (i) *shape* constraints fix the shape of features which cannot be changed according to design rules; (ii) *overlap* constraints ensure that features will remain electrically connected after compaction, and/or properly aligned between different layers; and (iii) *spacing* constraints enforce separation design rules (including PSM-specific separation rules between features of the same or of the different phase).

  Note that we perform y-compaction the same way that we do x-compaction (after temporarily swapping x and y coordinates). In the one-shot flow, we output the new positions of all features after compacting in x and y directions. In the iterative flow, we return control to the phase generator and continue the iteration until we obtain a valid phase assignment for all features.

4.2. Computational Experience

Table 1 summarizes our computational experience with seven layouts of different sizes and densities. All layouts were derived from industry standard-cell layouts. All runtimes are CPU seconds on a 300 MHz Sun Ultra-10 workstation with 128MB RAM, except for Layout 7 results, which are CPU seconds on a 140MHz Sun Ultra-1 with 320MB RAM. We see that our code can handle very large flat designs in reasonable time.

Test		Layout1	Layout2	Layout3	Layout4	Layout5	Layout6	Layout7
# polygons		3769	6914	9778	13796	18252	22954	36227
# rectangles		4549	8691	12971	18325	24255	30625	36227
conflict graph	# nodes	4549	8691	12971	18325	24255	30625	36227
	# edges	15204	12204	38342	55194	74924	97488	160400
	runtime	1.88	1.40	4.76	7.15	19.85	27.22	19.99
dual graph	# nodes	3250	1650	7348	10634	14642	19341	36926
	# edges	10530	2516	20292	30044	42026	56602	82962
	runtime	4.45	0.23	10.12	19.02	80.41	133.75	42.63
Voronoi graph	# nodes	934	0	1378	2046	2898	3968	2764
	# edges	3232	0	4164	6430	9430	13328	7064
	runtime	0.06	0	0.09	0.15	0.47	0.68	0.18
Matching	size	1402	0	2562	3529	5197	7289	5672
	runtime	1.1	0.26	1.79	4.91	22.68	63.67	5.96

Table 1. Computational results for phase assignment of layouts with various sizes.

In conclusion, we have suggested new, efficient and optimal algorithms for minimum-cost layout perturbation and conflict elimination in the dark field (negative photoresist, single exposure) alternating PSM context. Our approach has been integrated with a GDSII reader, polygon database and layout compactor. Our preliminary computational tests show that our code can assign phases to comparatively large designs in reasonable time. Other, potentially more powerful approaches to layout modification and phase assignment for alternating PSM are currently under investigation.

ACKNOWLEDGMENTS

We thank P.K. Vasudev for introducing the topic of phase-shifting masks to us in April 1997. We are also grateful to Frank Schellenberg for guidance and encouragement over the past year. Andrew Caldwell, Igor Markov and Devendra Vidhani provided critical assistance in our software development.

REFERENCES

1. C. Bamji and R. Varadarajan, *Leaf Cell and Hierarchical Compaction Techniques*, Kluwer Academic Publishers, Boston, 1997.
2. F. O. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", *SIAM J. Computing* 4(3) (1975), pp. 221-225.
3. A. B. Kahng, G. Robins, A. Singh, H. Wang and A. Zelikovsky, "Filling and Slotting : Analysis and Algorithms", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1998, pp. 95-102.
4. A. B. Kahng, G. Robins, A. Singh and A. Zelikovsky, "Filling Algorithms for Layout Density Control", manuscript, April 1998.
5. A. B. Kahng and H. Wang, "Toward Lithography-Aware Layout: Preliminary Litho Notes", manuscript, July 1997.
6. M. D. Levenson, N. S. Viswanathan and R. A. Simpson, "Improving Resolution in Photolithography with a Phase-Shifting Mask", *IEEE Trans. on Electron Devices* ED-29(11) (1982), pp. 1828-1836.
7. K. Mehlhorn, "A Faster Approximation Algorithm for the Steiner Problem in Graphs", *Information Processing Letters* 27 (1988), pp. 125-128.
8. A. Moniwa, T. Terasawa, N. Hasegawa and S. Okazaki, "Algorithm for Phase-Shift Mask Design with Priority on Shifter Placement", *Jpn. J. Appl. Phys.* 32 (1993), pp. 5874-5879.
9. A. Moniwa, T. Terasawa, K. Nakajo, J. Sakemi and S. Okazaki, "Heuristic Method for Phase-Conflict Minimization in Automatic Phase-Shift Mask Design", *Jpn. J. Appl. Phys.* 34 (1995), pp. 6584-6589.
10. J. Nistler, G. Hughes, A. Muray and J. Wiley, "Issues Associated with the Commercialization of Phase Shift Masks", *SPIE 11th Annual BACUS Symposium on Photomask Technology*, SPIE 1604 (1991), pp. 236-264.
11. K. Ooi, S. Hara and K. Koyama, "Computer Aided Design Software for Designing Phase-Shifting Masks", *Jpn. J. Appl. Phys.* 32 (1993), pp. 5887-5891.
12. K. Ooi, K. Koyama and M. Kiryu, "Method of Designing Phase-Shifting Masks Utilizing a Compactor", *Jpn. J. Appl. Phys.* 33 (1994), pp. 6774-6778.
13. G. I. Orlova and Y. G. Dorfman, "Finding the Maximum Cut in a Graph", *Engr. Cybernetics* 10 (1972), pp. 502-506.