New Graph Bipartizations for Double-Exposure, Bright Field Alternating Phase-Shift Mask Layout*

Andrew B. Kahng, Shailesh Vaya[†], and Alexander Zelikovsky[‡]

UCSD CSE and ECE Departments, La Jolla, CA 92093-0114
[†]UCLA Computer Science Department, Los Angeles, CA 90095-1596
[‡]Department of Computer Science, Georgia State University, Atlanta, GA 30303
abk@cs.ucsd.edu, vaya@cs.ucla.edu, alexz@cs.gsu.edu

Abstract— We describe new *graph bipartization* algorithms for layout modification and phase assignment of bright-field alternating phase-shifting masks (AltPSM) [25]. The problem of layout modification for phase-assignability reduces to the problem of making a certain layout-derived graph bipartite (i.e., 2-colorable).

Previous work [3] solves bipartization optimally for the *dark field* alternating PSM regime. Only one degree of freedom is allowed (and relevant) for such a bipartization: *edge deletion*, which corresponds to increasing the spacing between features in order to remove phase conflict. Unfortunately, dark-field PSM is used only for contact layers, due to limitations of negative photoresists. Poly and metal layers are actually created using positive photoresists and bright-field masks.

In this paper, we define a new graph bipartization formulation that pertains to the more technologically relevant *bright-field* regime. Previous work [3] does not apply to this regime. This formulation allows two degrees of freedom for layout perturbation: (i) increasing the spacing between features, and (ii) increasing the *width* of critical features. Each of these corresponds to *node deletion* in a new layout-derived graph that we define, called the *feature graph*. Graph bipartization by node deletion asks for a minimum weight node set A such that deletion of A makes the graph bipartite. Unlike bipartization by edge deletion, this problem is NP-hard. We investigate several practical heuristics for the node deletion bipartization of planar graphs, including one that has 9/4 approximation ratio. Computational experience with industrial VLSI layout benchmarks shows promising results.

I. INTRODUCTION

Alternating phase-shifting mask (AltPSM) technology is enabling to subwavelength process technology, the roadmap for which will last at least 7-10 years (from the 180nm generation through sub-50nm processes) [24, 11]. AltPSM uses destructive interference between opposite-phase light (e.g., 0 phase and 180 phase) to improve contrast on the wafer between exposed and unexposed regions [13, 14]. AltPSM affects circuit layout because there is no longer any concept of a "complete" design rules set: layout is correct if and only if a given layout-derived graph can be 2-colored. Since 2-colorability of this derived graph is difficult to maintain during layout creation, all proposed solutions (e.g., [21, 12, 3]) use post-processing of layout to identify required perturbations, followed by layout compaction to achieve a phase-assignable final layout.

Today's most viable AltPSM technology is due to Wang and Pati [25] (see, e.g., documentation at [19]), and involves double exposure (two masks) on positive photoresist. With positive photoresist, development removes photoresist material from all regions that have been exposed with sufficient energy. Hence, areas defining features should be protected from light and phases should be assigned to clear areas of the mask outside the features (i.e., "bright field" or "clear field"). The AltPSM technology of [25] is illustrated in Figure 1. In the figure, a poly feature that includes a critical-width gate[1] is formed by exposing two masks: (i) a "locally bright-field" AltPSM mask, followed by (ii) a binary (non-phase-shifting, standard chrome on glass) mask that protects the critical portion of the feature from light while also defining the non-critical width portions of the feature.

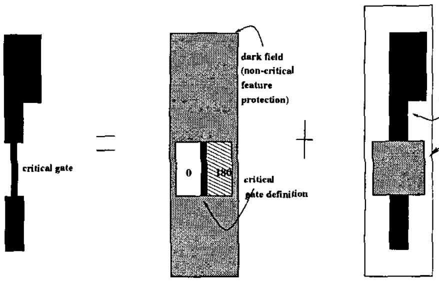

Fig. 1. AltPSM style of Wang and Pati (Numerical Technologies, Inc.). The critical portion of the feature is created with an AltPSM mask; the second binary mask protects the critical portion and defines the non-critical portions of the feature.

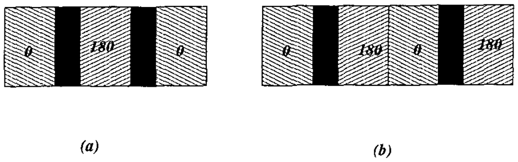

Fig. 2. When two vertical critical features are closely spaced, their phase shifters overlap and must be assigned the same phase (180 phase shifter in (a)). When the features are widely spaced (b), their phase shifters can be assigned phases independently.

Today, the AltPSM style shown in Figure 1 is most popular for improved length reduction poly gates (e.g., down to 25nm gates using 248nm wavelength illumination in the stepper) [6]. This "gate-shrink" modality improves circuit speed and reduces power budgets, and is relatively straightforward from the layout perspective since the application of phase shifters is "sparse". However, for true sub-130nm processes the full poly layer (and possibly local interconnect (salicide) layers as well) will need to be phase-shifted in order to maintain the transistor densities prescribed by the Roadmap [24].[2] In other words, AltPSM will become a lever for die area and die cost, in addition to speed and power.

When the majority of features are at critical width then the incidence of phase shifters becomes "dense": the layout must leave room for phase shifters around nearly every feature, and finding compatible assignments of phases to shifters must be ensured. The latter task is quite difficult, and maintaining design productivity for logic applications requires automated phase-mask layout tools.

Figure 2 shows that when two vertical critical features are closely

*This work was partially supported by Cadence Design Systems, Inc., by the MARCO Gigascale Silicon Research Center, by NSF Grant CCR-9988331, and by a GSU Research Initiation Grant.

[1] In our discussion, a *critical* feature is one that requires phase-shifting to be successfully printed.

[2] Improvements in k_1 and numerical aperture factors within the exposure system (i.e., the stepper) will not only reduce process windows, but by themselves are not enough to achieve the targeted 50nm processes with 157nm CaF steppers. AltPSM has been *officially* part of the technology Roadmap's required solution technologies since the publication of the 1999 ITRS six months ago [24].

0-7803-6633-6/01/$10.00 ©2001 IEEE.

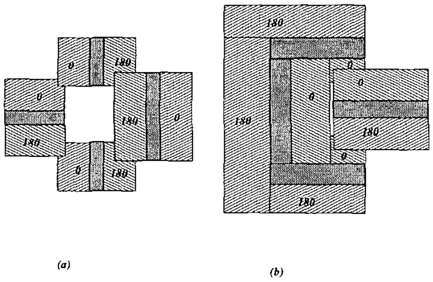

Fig. 3. Small layouts that illustrate the "odd cycle" problem of phase mask layout. There is no assignment of 0 and 180 phases to the shifters, such that (i) there are opposite-phase shifters on either side of each feature, and (ii) any shifters that overlap are assigned the same phase.

spaced, their phase shifters *overlap*, and must be assigned the same phase. On the other hand, when the features are widely spaced, their phase shifters can be assigned phases independently. We see that the *overlap* between shifters introduces dependencies between the phase assignments to shifters of corresponding features. Figure 3 gives simple layout examples for which there is no assignment of 0 and 180 phases to the shifters, such that (i) there are opposite-phase shifters on either side of each feature, and (ii) any shifters that overlap are assigned the same phase. In general, to minimize layout area, the phase assignment to shifters should eliminate or reduce the number of cases when adjacent shifters get opposite phases.

Phase Assignment Problem. Given a layout, find a phase assignment such that the following Conditions (1) and (2) are satisfied:

- Condition (1): Phase shifters on opposite sides of each critical feature are assigned opposite phases; and
- Condition (2): Any pair of overlapping shifters is assigned the same phase.

We know from Figure 3 that conditions (1)-(2) cannot always be satisfied. Such situations are caused by *odd cycles* of phase dependencies. In graph-theoretic terms, only graphs free of odd cycles can be properly colored into two colors[3] – meaning that shifters of the corresponding layout can be properly assigned 0 and 180 phases. In such cases, for the chip to be manufacturable the *layout must be modified* so that it becomes phase-assignable. A violation of condition (1) can be corrected via layout modification that *increases the width* of the corresponding critical feature, i.e., the feature must become sufficiently wide that it can be manufactured without phase shifting. With current photomask technology and 248nm DUV steppers, this results in increasing the width from 90-110nm up to 180-250nm (such a modification is potentially costly in terms of performance, e.g., if the critical feature represents a gate on a timing-critical path). A violation of condition (2) is corrected by layout modification that *increases the spacing* between critical features, from approximately 100-200nm to 200-400nm in 248nm DUV lithography. Note that the "odd cycle" problem illustrated in Figure 3 can in general be interpreted as a violation of *either* condition 1 or condition 2 (!) – and hence can be corrected by increasing *either* feature width or feature spacing. As noted in [3], we would like to minimize the total cost of the layout modifications applied:

Minimum Distortion Problem. Given a layout, find a solution to the Phase Assignment Problem which requires minimum layout modification.

While our present work and the work of [3] share the Minimum Distortion Problem statement, their solution and ours are completely different due to the available degrees of layout freedom in the bright-field context. We next define the *feature graph* for which we seek bipartization; note that this is very different from the *conflict graph* discussed in [3].

II. The Feature Graph

In this section, we propose a new *feature graph* to represent relationships between adjacent layout features and their corresponding shifters. The feature graph allows us to reduce the Phase Assignment Problem to graph bicoloring. Furthermore, the structure of the feature graph allows *both* types of layout modifications (feature width increase, and feature spacing increase) to be applied, along with recent advanced discrete algorithmic methods. Previous methods [3] have addressed only the feature spacing degree of freedom (and only in the dark-field regime).

The Minimum Distortion Problem asks to minimize the cost of correcting all violations of conditions (1)-(2), since each violation results in increasing area or slowing down the chip. We will model the layout modification used to correct violations of condition (1) as deletion of a node corresponding to the critical feature. Layout modification used to correct violations of condition (2) will correspond to either edge or node deletion. Both edge and node deletion help to eliminate odd cycles. Following is the formal description for how to construct the *feature graph*.

Given a layout, the *feature graph* $G = (F \cup C \cup S, E)$ consists of the three types of nodes F, C and S and edges E:

(F) For each critical feature f^* we put into correspondence a *feature* node $f \in F$;

(C) For each overlap of two shifters we put into correspondence a *conflict* node $c \in C$;

(E) Any feature node f is connected to all conflict nodes representing overlaps of the shifters which are on the sides of the corresponding feature f^*;

(S) Edges between feature node f and conflict nodes of one of its shifters (arbitrarily chosen) are subdivided into paths of length 2 by *shifter* nodes $s \in S$;

Note that all conflict and shifter nodes have degree 2, and only feature nodes may have arbitrary degree. Figure 4 shows the feature graph for a layout with four critical features.

The useful properties of the feature graph are justified by the following

Theorem 1 *Let G be the feature graph of the layout L. Then*

(i) the Phase Assignment Problem has a feasible solution for L if and only if G is 2-colorable (i.e., G is bipartite);

(ii) increasing the width of a feature f^* in L is equivalent to deleting the corresponding feature node f from G;

(iii) increasing the spacing between two features in L that have overlapping shifters is equivalent to deleting the corresponding conflict node c from G or deleting any of the edges (f,c), (f,s) or (c,s), where f corresponds to either of the two features and s is the shifter node that possibly subdivides the f-to-c connection.

Proof. We will first prove (i). The two colors in the bicoloring of G correspond to the two phases of shifters in L.

Let c and c' be two conflict nodes corresponding to the overlaps of the same shifter with some other shifters. Then any bicoloring of G will assign c and c' the same color because they are connected with

[3] One cannot color the nodes of an odd cycle into two colors, such that all pairs of adjacent nodes receive different colors.

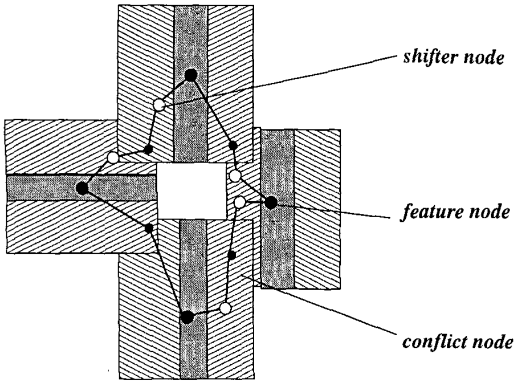

Fig. 4. Feature graph for four features: four feature nodes are large and filled, four conflict nodes are small and filled and and five shifter nodes are large and empty.

a path of length 2 (case (C)) or 4 (case (S)) through the same feature node. This means that any bicoloring of G will color all conflict nodes corresponding to the same shifter with the same color.

If c and c' correspond to overlaps of the opposite shifters, then they are connected with the path of length 3 through the corresponding feature node, and they should attain different colors. This ensures that opposite shifters of the same feature will get opposite phases and Condition (1) is satisfied. Condition (2) is satisfied since conflict nodes are shared by overlapping shifters.

On the other hand, if L has feasible phase assignment, then we color all conflict nodes in the color (phase) of their shifters; the feature nodes will get the color opposite to the color of adjacent conflict nodes. By the argument above, opposite conflict nodes will get opposite colors and the conflict nodes of the same shifter will attain the same color.

When a critical feature is widened, then we should drop the corresponding feature node from G as well as its conflict nodes. Note that after deletion of a feature node its conflict nodes become leaves or adjacent to leaves. Property (ii) follows immediately from the fact that two-colorability of a graph is not affected by leaves or by nodes of degree 2 adjacent to leaves.

Finally, property (iii) is true since edge deletion makes the corresponding conflict node either of degree 1 or adjacent to a leaf. □

We also may supply nodes and edges of the feature graph G with weights reflecting the relative costs of the spacing enforcement and the critical feature widening layout perturbations. Then Theorem 1 implies that the Minimum Distortion Problem is equivalent to the following

Graph Bipartization Problem Given an edge and node weighted graph G, find the minimum weight edge or node set D, such that the graph $G - D$ is bipartite.

Justification of the node deletion formulation.

To prevent any misunderstanding, we briefly discuss why it is acceptable to use *both* the widening and spacing degrees of freedom in layout perturbation.

- We note that if (polygon-level) layout perturbation is a degree of freedom for the designer, then the (static) timing and signal integrity verification will typically be done by a transistor-level tool such as Synopsys PathMill (as opposed to a gate-level tool such as PrimeTime). Although there are issues of design convergence, performance tuning flows involving incremental polygon layout and incremental transistor-level static timing/SI analysis are fairly well-understood.[4]
- After static timing and signal integrity analysis, poly gates that are not performance-critical can be assigned a low cost of widening, while gates that are performance-critical can be assigned a high cost. These costs can be driven by the same sensitivity analyses that are already available within performance optimization tools.
- Widening of (poly) interconnects (which are phase-shifted in a full-chip PSM methodology; recall the discussion of the transistor density roadmap in Section 1) will generally maintain a constant RC product and thus not affect performance significantly. In general, for both poly and local interconnect, widening of a critical-width geometry while maintaining spacing to neighbors will tend to improve performance, since the ratio of fringing capacitance to area capacitance decreases.

III. Bipartization of Planar Graphs

The Graph Bipartization Problem is NP-hard for general graphs in the edge-deletion and node-deletion versions. In this section, we will first show that the feature graph constructed in the previous section is planar. Note that the result and the argument are different from those of Theorem 2.1 in [3] (proving planarity of the conflict graph in dark-field AltPSM): the contexts are quite different. After proving planarity of the feature graph, we will consider various heuristic algorithms for node-deletion bipartization and edge-deletion bipartization of planar graphs.

Theorem 2 *The feature graph is planar if the maximum width of a shifter is less than half the minimum length of a feature.*

Proof. Consider the following embedding of the feature graph $G = (F \cup C \cup S, E)$ into the Euclidean plane:

1. For any feature f^*, the corresponding *feature node* $f \in F$ is placed at the geometric center of the corresponding feature rectangle.
2. For every pair of overlapping shifters, the corresponding *conflict node* $c \in C$ is placed at the geometric center of the overlapping area of the shifters.
3. We connect with straight lines each feature node $f \in F$ to the conflict nodes representing overlaps of the shifters which are on the sides of the corresponding feature f^* according to step (E) of the definition of the feature graph; if necessary, we subdivide this line with the shifter node $s \in S$ according to step (S).

We show that no two edges of the feature graph G embedded in the Euclidean plane as described above can cross without violating the condition that the length of a shifter is less than half the length of a feature rectangle. Suppose two edges in the graph G cross. Without loss of generality let it be the edge e as shown in Figure 5. Since a feature rectangle can't overlap another feature rectangle or another shifter, an edge f (note that every edge completely lies inside a shifter and its corresponding feature rectangle can cross edge e only when one of the following cases occur, without loss of generality(symmetric cases):

(a) When the edge e corresponds to the overlap of a vertical and a horizontal shifter; Figure 5(a).

[4] E.g., Motorola and IBM tuning flows reported at recent DAC and ICCAD conferences; Cadence CoreMaster; Synopsys AMPS; etc. are all in production.

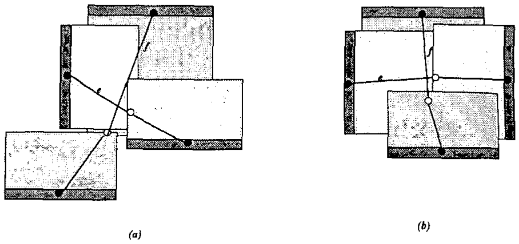

Fig. 5. Two cases of self-intersection of the feature graph

```
Input: Planar graph $G$
Output: The bipartite subgraph $H$

For each odd face $f$ of $G$, initialize $age(f) = 0$
While there are odd faces in the graph $G$ do
    For each odd face $f$ of $G$, increment $age(f) = age(f) + 1$.
    For each node $v$ in the graph $G$ set $weight(v)$ to the sum of ages
        of all odd faces with the node $v$ on the boundary.
    Delete the node $v$ with the largest weight.
    If a new face $f$ is odd, then initialize $age(f) = 0$.
In the reverse order of deletions do
    Add node $v$ with all adjacent edges to the graph
    If an odd face appears, then delete $v$ from the graph $G$
```

Fig. 6. Primal-Dual Algorithm (Goemans-Williamson)

(b) When the edge e corresponds to the overlap of two vertical shifters, Figure 5(b).

But as can be seen, these cases are possible if no shifter length(same as the corresponding feature length) is less than the twice the width of a shifter. Thus, we arrive at a contradiction. Hence the graph G must be planar under the assumption $2 \cdot width < length$. □

Four Heuristics

In the remainder of this section, we propose four distinct heuristics for feature graph bipartization which are experimentally studied in the next section. The *planarity* of the feature graph greatly reduces complexity of bipartizing. In fact, edge-deletion bipartization can be solved in polynomial time for planar graphs [22, 10]. An efficient implementation of the optimal algorithm for edge-deletion bipartization is suggested in [3]. On the other hand, node-deletion bipartization of planar graphs is NP-hard [27], but provably better approximate solutions can be found in planar graphs rather than in general graphs.

The best approximation algorithm yet known for (weighted) node-deletion bipartization in planar graphs is by Goemans and Williamson [9], and guarantees a solution that is at most $\frac{9}{4}$ times worse than optimum (see Figure 6). The runtime of the Goemans-Williamson algorithm is $O(V^2)$, where V is the number of nodes.

Figure 7 describes a simpler Greedy Vertex Covering heuristic for node-deletion. This provides a fast $O(V \log V)$ solution, but with no guarantees of solution quality. (The basic idea of the Greedy Vertex Covering algorithm is to find an approximate minimum vertex cover of the set of edges that it cannot color properly.[5]) We can implement the Greedy Vertex Covering Algorithm to run in time $O(E \log V)$, where $O(\log V)$ is the worst case running time for updating the violation degree of a node on the violation heap. Since the *feature graph* is planar, its number of edges is $\|E\| < 5 * V$, hence Greedy Vertex Covering can be implemented to run in $O(V \log V)$ time.

```
Input: Planar graph $G$
Output: The bipartite subgraph $H$

Color all nodes into two colors 1 and 2 traversing all nodes using
    breadth-first search
Find the set $T$ of all violating edges for which both endpoints
    attained the same color.
While there are violating edges do
    Delete the node with the maximum violation degree, i.e., the
        degree in the edge set $T$.
```

Fig. 7. Greedy Vertex-Covering Algorithm

We have also considered and implemented two pure edge-deletion bipartization methods from [3]. These methods cannot exploit the feature-widening layout degree of freedom, but obviously can still achieve phase-assignable solutions.

- The *Edge-Gadget* algorithm optimally finds the minimum set of shifter overlaps that should be forbidden (i.e., removed) in order to obtain a feasible solution for the Phase Assignment Problem. In our implementation we use the efficient algorithm from [3] for optimal edge-deletion bipartization.
- The *Edge-Greedy* algorithm for edge deletion just follows the Greedy Vertex Covering Algorithm, except that it deletes *all* violation edges. The approach is very fast, but is known to delete more than twice the optimum number of edges.

IV. Results and Conclusions

All four phase assignment algorithms for the Minimum Distortion Problem have 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.[6] The spacing constraints induced by solution of the problem, along with the resulting phase assignment, can be directly sent to compaction. We compare edge-deletion and node-deletion bipartization on feature graphs derived from two industry layout testcases whose attributes are summarized in Table I. The experimental results in Table II lead to two main observations. First, the table shows the clear superiority of the Goemans-Williamson 9/4-approximation, over the faster Greedy Vertex Covering heuristic. We believe that the improved solution quality is well worth the extra runtime (which is still very reasonable). Second, the table shows that the trade-off between two types of layout modifications – (1) increasing spacing constraints between features that have shifters in phase conflict which is equivalent to edge deletion and (2) widening critical features which is equivalent to node deletion – can be effectively exploited by our new node-deletion heuristics. Since the relative cost of these two modifications can vary, the data

[5] There exist 2-approximations to vertex covering, but since the initial BFS-based coloring is of such uncertain value, we have not yet implemented a stronger vertex covering heuristic.

[6] Our implementation is currently restricted to rectilinearly oriented features, but there are no major obstacles to handling octilinear or all-angle geometries (e.g., slicing of polygons would be into parallelograms or trapezoids, respectively). The focus of our work is on near-optimal solution of Minimum Distortion Problem.

TABLE I

THE NUMBER OF CRITICAL FEATURE RECTANGLES (WIRES), THE NUMBER OF SHIFTERS (TWICE THE NUMBER OF FEATURE RECTANGLES), AND THE NUMBER OF CONFLICT NODES (I.E., NUMBER OF SHIFTER OVERLAPS WHEN THE SHIFTERS ARE OF DIFFERENT FEATURE RECTANGLES) FOR THE LAYOUTS.

Layout1			Layout2		
#wire	#shifter	#conflict	#wire	#shifter	#conflict
8622	17244	7805	4539	9078	5493

shows the total modification cost for various ratios of node deletion cost divided by edge deletion cost.[7] Table II shows that when the ratio is as low as 1.5, the total cost of layout modification can be significantly reduced by exploiting the node-deletion degree of freedom: e.g., for Layout1 the total cost is 268.5 versus 314 for pure edge deletion. (When the cost ratio goes to 1.0 or below, the solutions in the table have costs 228 and 187 for Layout1 and Layout2, respectively, compared to 314 and 224 for optimal pure edge deletion.) At the same time, when the cost ratio becomes large, optimal pure edge deletion is clearly cheaper than the approximate solution that uses both types of modifications.

In conclusion, we have suggested the first optimal and approximate efficient algorithms for the Minimum Distortion Problem in double-exposure, bright-field alternating phase-shift mask layout. Our approach has been integrated with a GDSII reader and polygon database, and is currently being integrated with industrial layout compaction. Our preliminary computational tests show that our code can assign phases to comparatively large designs in reasonable time, and can efficiently exploit the availability of two distinctly costed types of layout modifications.

ACKNOWLEDGMENTS

We thank the authors of [3] for source code of their edge-deletion based phase conflict resolution software. We also thank Buno Pati, Yao-Ting Wang, Hua-Yu Liu, Clive Wu and Linard Karklin of Numerical Technologies, Inc. for software licenses and valuable discussions.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft and J. D. Ulman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading, MA, 1974.

[2] C. Bamji and R. Varadarajan, *Leaf Cell and Hierarchical Compaction Techniques*, Kluwer Academic Publishers, Boston, 1997.

[3] P. Berman, A. B. Kahng, D. Vidhani, H. Wang and A. Zelikovsky, "Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks", *IEEE Trans. on CAD* 19(2) (2000), pp. 175-187.

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, *Combinatorial Optimization*, Willey Inter-Science, New York, 1998.

[5] W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings", *http://www.or.uni-bonn.de/home/rohe/matching.html*, August 1998.

[6] M. Fritze, "Application of Chromeless Phase Shifting Masks to Sub-100nm SOI CMOS Transistor Fabrication", *Proc. SPIE Microlithography*, March 2000 (50nm announcement), and M. Fritze et al., "Sub-100 nm SOI CMOS by DUV Optical Lithography", *Proc. 44th International Conference on Electron, Ion, and Photon Beam Technology & Nanofabrication (IEDM 3-Beam Conference)*, May 2000 (25nm announcement).

[7] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for general graph matching problems", Journal of the ACM 38 (1991) 815-853.

[8] M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems", SIAM Journal on Computing 24 (1995) 296-317.

[9] M. X. Goemans and D. P. Williamson, "Primal-Dual Approximation Algorithms for Feedback Problems in Planar Graphs", Combinatorica 18 (1998) 37-59.

[10] F. O. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", *SIAM J. Computing* 4(3) (1975), pp. 221-225.

[11] A. B. Kahng and Y. C. Pati, "Subwavelength Lithography and its Potential Impact on Design and EDA", *Proc. ACM/IEEE Design Automation Conf.*, 1999, pp. 799-804.

[12] A. B. Kahng, H. Wang and A. Zelikovsky, "Automated Layout and Phase Assignment Techniques for Dark Field Alternating PSM", *SPIE 18th Annual BACUS Symposium on Photomask Technology*, SPIE 3546 (1998), pp. 222-231.

[13] 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.

[14] H.-Y. Liu, L. Karklin, Y.-T. Wang and Y. C. Pati, "The Application of Alternating Phase-Shifting Masks to 140 nm Gate Patterning: Line Width Control Improvements and Design Optimization", *SPIE 17th Annual BACUS Symposium on Photomask Technology*, SPIE 3236 (1998), pp. 328-337.

[15] K. Mehlhorn, "A Faster Approximation Algorithm for the Steiner Problem in Graphs", *Information Processing Letters* 27 (1988), pp. 125-128.

[16] 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.

[17] 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.

[18] 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.

[19] Numerical Technologies, Inc. http://www.numeritech.com/

[20] 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.

[21] 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.

[22] G. I. Orlova and Y. G. Dorfman, "Finding the Maximum Cut in a Graph", *Engr. Cybernetics* 10 (1972), pp. 502-506.

[23] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[24] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, December 1999, www.itrs.net/ntrs/publntrs.nsf

[25] Y.-T. Wang and Y. C. Pati, "Phase Shifting Circuit Manufacture Method and Apparatus", United States Patent 5858580, January 12, 1999.

[7] Of course, our method allows each node and each edge in the feature graph to have its own deletion cost.

TABLE II

Computational results for phase assignment for Layout 1 and Layout 2. The following four algorithms are compared: VCG - Vertex Covering Greedy Algorithm; (Fig. III, GW - Goemans-Williamson 3-approximation algorithm (Fig. III), Edge-Greedy Algorithm which greedily deletes violating edges and Edge-Gadget Algorithm which finds optimal number of edges for deletion. # features is the number of feature nodes corresponding to features which should be widened. The number of conflict nodes is equivalent to the number of deleted conflict edges, since deletion of either will result in increasing the spacing between corresponding critical features. The *Ratio* of the cost of feature node deletion (i.e., corresponding feature being made non-critical from critical) divided by the cost of conflict node deletion (i.e., increasing the spacing between corresponding features) is varied between 1.5 and 20. The total cost of bipartizing is in the cost column. All runtimes (RT) are in seconds for a 300 MHz Sun Ultra-10 workstation with 128MB RAM.

Testcases		Layout1				Layout2			
Algorithm	Ratio	Cost	#features	#conflict nodes/edges)	RT (sec)	Cost	#features	#conflict nodes/edges)	RT (sec)
VCG	1.5	430	62	337	9	371	73	262	7
GW		268.5	81	147	231	225	77	110	130
VCG	2.0	461	62	337	9	408	73	262	7
GW		306	61	184	249	263	57	149	149
VCG	2.5	468	14	433	9	423	30	348	7
GW		341	7	323	354	302	10	277	230
VCG	3.0	475	14	433	9	438	30	348	7
GW		344	3	335	368	307	8	283	236
VCG	3.5	477.5	5	460	9	445	15	393	7
GW		344	2	337	371	312	2	305	250
VCG	4.0	480	5	460	9	453	15	393	7
GW		345	2	337	371	314	1	310	251
VCG	20.0	484	0	484	9	501	0	501	6
GW		347	0	347	383	322	0	322	267
Edge-Greedy		485	0	485	7	502	0	502	5
Edge-Gadget		314	0	314	11	224	0	224	15

[26] D. P. Williamson and M. X. Goemans, "Computational experience with an approximation algorithm on large-scale Euclidean matching instances", INFORMS Journal of Computing, 8 (1996) 29-40.

[27] M. Yannakakis, "Node and edge-deletion NP-complete problems", *Proc. ACM Symp. on Theory of Comp.*, 1978, pp. 253-264.