On Mask Layout Partitioning for Electron Projection Lithography *

Ruiqi Tian[1], Ronggang Yu[2], Xiaoping Tang[3], D. F. Wong[4]

1. Motorola Inc., Mail Drop K10, 3501 Ed Bluestein Blvd., Austin, TX 78721
2. Dept. of CS, University of Texas at Austin, Austin, TX 78712
3. Silicon Perspective, a Cadence Company, San Jose, CA 95134
4. Dept. of ECE, University of Illinois at Urbana-Champaign, Urbana, IL 61801

Abstract

Electron projection lithography (EPL) is a leading candidate for next generation lithography (NGL) in VLSI production. The membrane mask used in EPL is divided into sub-fields by struts for structural support. A layout must be partitioned into these sub-fields on mask and then stitched back together by the EPL tool on wafer. To minimize possible stitching errors, partitioning of a mask layout should minimize cuts of layout features in the overlapping area between two adjacent sub-fields. This paper presents the first formulation of the mask layout partitioning problem for EPL as a graph problem. The graph formulation is optimally solved with a shortest path approach. Two other techniques are also presented to speed up computation. Experimental runs on data from a real industry design show excellent results.

1. Introduction

VLSI production currently uses deep ultra-violet (DUV) lithography. Wave length in today's DUV stepper is typically 248nm. Only recently and in the coming years, DUV steppers with wave lengths of 193nm and 157nm are and will be introduced. In comparison, the minimum feature size for a state-of-the-art CMOS production is already at 130nm today. Because feature sizes in the deep-submicron regime can be smaller than wave lengths used to print them, numerous advanced, and highly complex, resolution enhancement techniques are explored to extend the life of DUV lithography tools. Yet, because of the physical limit due to diffraction of light, many experts in the industry are predicting that the ultimate limit for optical lithography tools is fast approaching [1]. At technology nodes down to about 50nm, further scaling of feature sizes will be seriously limited by optical diffraction [2]. To facilitate the continued aggressive scaling of feature sizes according to Moore's law, new lithography schemes are required for the next generation lithography (NGL) after DUV.

Current leading candidates for NGL are extreme ultra-violet (EUV) lithography and electron projection lithography (EPL). EUV light with a wave length of about 13nm, which is soft X-ray, will be able to solve the resolution problem for many future technology nodes below 50nm. Unfortunately, EUV light is strongly absorbed by solid-state materials, so reflective masks, with tens of layers of reflective coatings each, is the most difficult technological challenge for EUV [3]. In addition, search for new resist materials is another serious challenge for EUV. In contrast, although EPL, with a wave length of just a few angstroms for a 100KeV source, has its own challenges like charged particle interaction and mask making, EPL is a relatively mature technology that has been investigated since the 1970's. EPL does not suffer from the same technical problems such as source, control, and resist, as those of EUV. The most difficult problem for EPL is throughput. Many industry leaders proposed ways to solve the throughput problem. Both Lucent's SCALPEL [4] and IBM's PREVAIL [5] are the most promising recent, production-worthy EPL approaches.

Regardless of details in setup, all leading EPL approaches use membrane masks. These EPL masks are composed of metallic shapes attached to the surface of a thin membrane. The metallic shapes block or scatter incident electron beam to define layout features on wafer. The supporting membrane is very thin so that little resistance is introduced to electrons not scattered by the metallic shapes. Hence, large EPL membrane masks intrinsically lack structural integrity. Struts, which act like a network of support beams, are inserted onto the masks to make them functional. The struts usually divides the EPL mask into sub-fields of equal sizes. Figure 1 shows a schematics of a membrane mask for EPL.

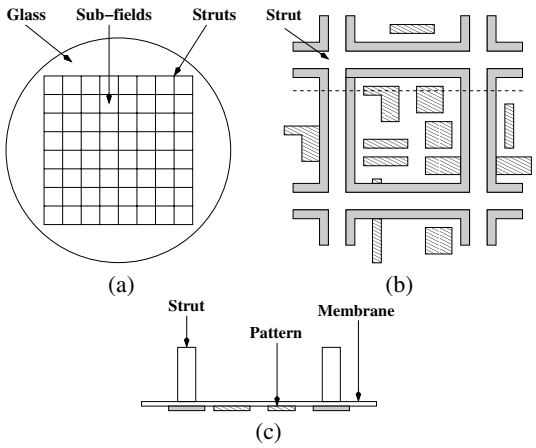

Figure 1: A membrane mask for EPL. (a) shows the top view of the mask. (b) shows one sub-field on the mask with adjacent sub-fields only partially shown. The shaded polygons are metallic shapes, and the gray regions are parts of overlapping regions used in stitching. (c) shows a cross-sectional view along the dashed line in (b).

As a consequence, a mask layout is partitioned into sub-fields divided by the struts on the membrane mask. The EPL process then "stitches" together on wafer the features in the adjacent sub-fields. The stitching is done by diverting the electron beams slightly after

*The work of D.F. Wong was partially supported by the National Science Foundation under grant CCR-9912390.

0-7803-7607-2/02/$17.00 ©2002 IEEE

the beam passes through the mask. The edge regions of adjacent sub-fields are therefore overlapped slightly in the stitching process. Although the stitching process in EPL offers a high degree of accuracy for features separated into adjacent sub-fields, there are still possible small errors associated with the process. Abutting shapes along the cut line can be shifted slightly. This small alignment error may be costly if the shapes are minimum features. Figure 2 shows how stitching occurs between adjacent sub-fields on a mask for EPL and two possible alignment errors. Nakasuji *et al.* [6] proposed in theory that features should be avoided when partitioning the layout, but no detailed approach is currently available in the public domain. Figure 3 shows how different partitions can avoid a possible alignment error.

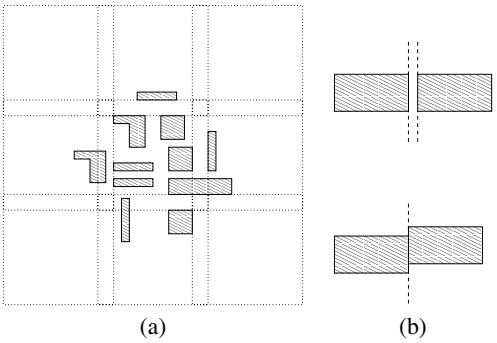

Figure 2: Stitching in EPL. (a) shows a portion of a sample layout with large dotted squares indicating locations of exposure of sub-fields. The sub-fields are exposed with overlaps. Only relevant sub-fields are shown for the sub-field at the center. (b) shows two possible errors in stitching for a rectangle divided into two sub-fields along the dashed line.

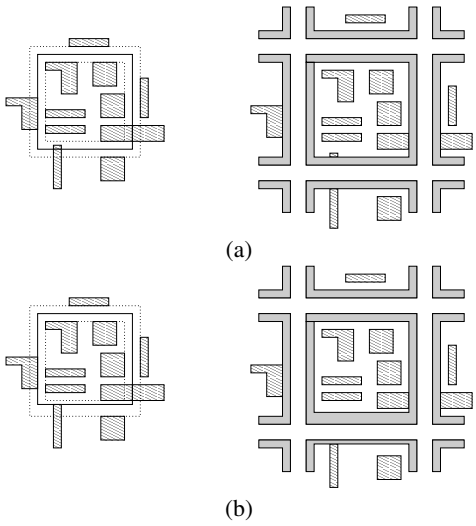

Figure 3: Mask definition in EPL. (a) and (b) show two slightly different partitions and corresponding mask layouts of the sample layout in Figure 2(a). The region between dotted squares are the overlapping region for the sub-field at the center. Solid lines indicate partition cuts. The lower-left rectangle in (b) is not cut, and the corresponding metallic shapes are different from those in (a).

In this paper, the mask layout partitioning problem for EPL is first analyzed and formulated as a graph problem in Section 2. Then, in Section 3, a shortest path approach is proposed to solve the problem optimally followed by two methods to speed up the computation. Experimental results for some real data are given in Section 4. Finally, some concluding remarks are in Section 5.

2. Problem Formulation

As shown in Fig.1(a) and Fig.2(a), all overlapping areas between two adjacent rows of sub-fields form a long and narrow rectangular region in the horizontal direction. This long and narrow region is called a "tape". A continuous partition line running from the left end to the right end within this tape separates the tape into two vertical parts. The top part defines the bottom edges for the top row of sub-fields, and the bottom part defines the top edges for the bottom row. Similarly, overlapping areas between adjacent columns form vertical tapes, and partition lines in those tapes divide the layout horizontally. Because sizes of sub-fields and overlapping areas are set by the EPL process, locations of tapes for a mask layout are predetermined. Hence, partitionings in parallel tapes are indepedent. However, within one particular tape, partitions of its sections, as defined by tapes perpendicular to the current tape, are not independent, because partition lines in two adjacent sections cannot be connected without considering the cost of such a connection. Also, the interaction between two perpendicular tapes usually can be safely ignored for simplicity. The mask layout partitioning problem is then reduced to the partitioning problem of a single tape. In the rest of this paper, a single tape in the horizontal direction is assumed without loss of generality.

All mask layout features can be decomposed into rectangles, assuming that only rectilinear shape is allowed in layout. To minimize stitching error, a partition line in a single tape should detour as many of these rectangles as possible. The following is the problem's formal statement.

> *Problem Statement: Given a tape covering a section of a mask layout consisted of rectanglur features, find a rectilinear partitioning line (a cut) within the tape from the left end to the right, such that the cut intersects with the minimum number of layout features.*

Figure 4 illustrates a portion of a tape with an optimal cut.

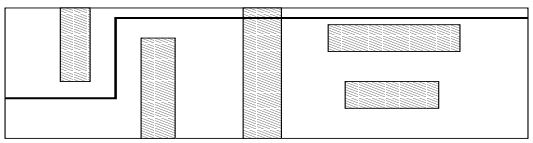

Figure 4: An optimal cut in a portion of a tape. Layout features are shaded rectangles. The optimal cut is drawn with a bold line, and it detoured as many rectangles as possible on this portion of the tape.

In practice, a cut has to be kept a small distance away from layout features, so features will not be adversely affected in mask processing. Expanding all features in all directions by the small distance before their decomposition into rectangles satisfies this requirement. After expansions, cuts can go along boundaries of the extended rectangles. Figure 5 shows extended rectangles.

Rectilinear cut lines are confined to a design grid to avoid snapping. The design grid is too fine to be used as the search graph for an optimal cut. Hanan grid graph [7] is naturally considered to facilitate the computation of an optimal cut. Every rectangle on the tape has four vertices. Let S denote the collection of all vertices

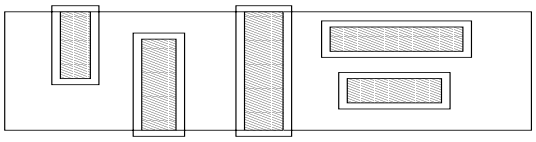

Figure 5: Extended rectangles with original features shaded.

generated from all rectangles on the tape. The Hanan grid $H(S)$ of S is obtained by constructing vertical and horizontal lines through each points in S. Figure 6 is the Hanan grid based on the tape in Fig. 5. A cut can go along Hanan grid from one end of the tape

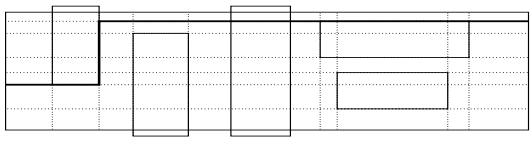

Figure 6: An optimal cut in Hanan grid

to the other. If an optimal cut is not on the Hanan grid, it is obvious that this optimal cut can be converted to an equivalent optimal cut on the Hanan grid. Therefore, there must exist one optimal cut along Hanan grid edges.

The problem of finding an optimal cut on the tape using a Hanan grid can be converted to a single source shortest path (SSSP) problem. First, two auxiliary vertices s and t are added to the left and right of the Hanan grid as the source and sink, respectively. Directed edges from s to the left-most vertices $\{lm\}$ and from the right-most vertices $\{rm\}$ to t are added as well. Every edge in $H(S)$ is made bi-directional to form $H'(S)$. Figure 7 shows a single source directed graph derived from Hanan grid in Fig. 6. Because

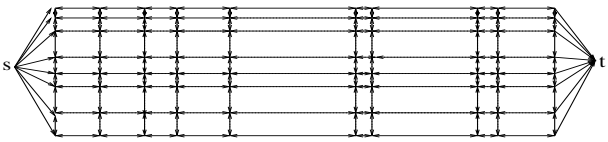

Figure 7: Single source directed graph

the objective is to avoid as many rectangles as possible, weights for Hanan grid edges located inside the rectangles must be larger than weights for edges outside. The weight of each connected edge $[v_i, v_j]$ in the single source directed graph is set by the following formula:

$$w[v_i,v_j] = \begin{cases} 0 & \text{if } v_i = s, \text{ or } v_j = t \\ \beta * dis[v_i,v_j] & \text{if } (v_i,v_j) \text{ is inside a rectangle} \\ \alpha / dis[v_i,v_j] & \text{otherwise} \end{cases} \quad (1)$$

where α and β are positive constants with $\beta >> \alpha$, and $dis[v_i, v_j]$ is the linear distance between v_i and v_j. Weights of edges starting from s and those arriving at t are set to 0, because they are connected to auxiliry vertices. Values for β are set large enough so that any path outside of all rectangles has a smaller cost than one that cuts inside any rectangles. Weight of a Hanan grid edge, that is outside or on the boundary of rectangles, is inversely proportional to its length to favor long straight cuts over jagged ones. Cost inside a rectangle, like for those spanning height of the tape, is proportional to cut length. All weights set by this formula are non-negative. The SSSP problem is—given a directed Hanan grid graph $G = (V,E)$ with $V = \{s\} \cup \{t\} \cup S$, $E = H'(S) \cup \{(s,v) | v \in \{lm\}\} \cup \{(v,t) | t \in \{rm\}\}$, and weights for edges w specified by Eq.(1), find a path from s to t with minimum cost. A shortest path solution to the SSSP problem forms a cut line. The cut line produced may not be monotonic in direction, but it will not contain any cycle, which is forbidden in mask processing.

The SSSP problem is the first formulation of the mask layout partitioning problem. The weight function for the Hanan grid graph can be refined to consider other important factors from process, such as cutting through a large feature may not be as costly as cutting through a minimum feature, or, the cut line through a metal shape should be aware of the landing locations of nearby vias.

3. Problem Solutions

3.1 Dijkstra's Algorithm

For the SSSP problem on a weighted, directed graph $G(V,E)$ with all weights non-negative, Dijkstra's algorithm [8] is an optimal solution. With a Fibonacci heap as implementation for the priority queue, the running time for Dijkstra's algorithm is $O(|V|\lg|V| + |E|)$. Because $|V| < |E| < 4|V|$ for a Hanan grid graph, the running time for Dijkstra's algorithm is $O(|V|\lg|V|)$. Details of Dijkstra's algorithm is omitted here for space reasons. Please refer to [9] for details and analysis of Dijkstra's algorithm.

Although Dijkstra's algorithm can gaurantee an optimal solution to the SSSP problem with $O(|V|\lg|V|)$ complexity, the algorithm does not scale well when $|V|$ is large. Yet, in the Hanan grid graph proposed in Section 2, each rectangle contributes four vertices, and intersections of horizontal and vertical lines originated from all vertices form new vertices, so in the worst case, the number of vertices in the Hanan grid can be of the same order as vertices for the dense design grid in the background. Furthermore, construction of graph $G(V,E)$ is inefficient for a large $|V|$. To assign weights for $4|V|$ edges with $|R|$ rectangles inside the tape, the edges are sorted to see if they are inside or outside the rectangles. The sorting takes $O(4|V|\lg|R|)$, or $O(|V|\lg(|R|^4))$ time, which can be of the same order as running the algorithm! Since the number of vertices in a Hanan grid is large, more efficient algorithms are needed to speed up computation.

3.2 Divide and Conquer

A tape is long in width, but very short in height, so there may be empty spaces that span the whole height of the tape. These blank spaces divides the tape into non-overlapping blocks along the tape. It is reasonable to assume the cost for cut lines in empty spaces to be negligible, if avoiding rectangles is the ultimate objective. Consequently, a divide-and-conquer approach is used to first find the empty, or "free", spaces and thus the non-empty blocks on a tape; then each of the blocks is treated as an independent SSSP problem and solved by Dijkstra's algorithm; finally, the cut lines from each block is joined together by arbitrary connections in free spaces between the blocks. Figure 8 illustrates the divide-and-conquer approach. In addition to Dijkstra's algorithm, the divide-and-conquer

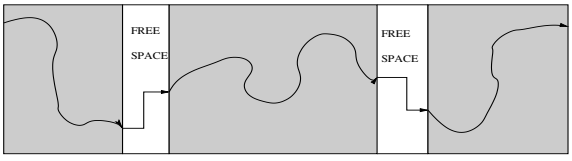

Figure 8: A divide-and-conquer approach. Shadowed areas are blocks. Cut lines from adjacent blocks are connected by a simple "s"- or "z"-shaped lines in free space.

approach uses an algorithm for finding free spaces. A simple algorithm for finding a set of free spaces F on a tape with width l and a set of rectangles R is shown below.

Algorithm FIND-FREE-SPACE (l, R)
1. $F = \{l\}$
2. **for** each rectangle r in R **do**
3. **for** each free space f in F **do**
4. reduce the size of f if it partially overlaps r
5. remove f from F if it is contained in r's range
6. replace f with f_1 and f_2 if r cuts f into f_1 and f_2
7. **end for**
8. **end for**
9. output F

The divide-and-conquer approach is much more efficient than running Dijkstra's algorithm on the whole tape. The number of vertices in the Hanan grid for a single block is much smaller, because grid lines from other blocks are not present. Also, it is simple to combine the partial solutions from the blocks with the simple strategy shown in Fig. 8. Although solutions from this approach is not optimal for the SSSP problem of the whole tape, they are optimal for the original problem by avoiding as many rectangle as possible.

The divide-and-conquer method is only useful when the tape is divided into many smaller blocks by free spaces. For cases where no free space is available on a tape or where large blocks have large numbers of vertices in the Hanan grid, the divide-and-conquer approach may not offer acceptable performance.

3.3 Dynamic Programming

A large block needs to be further divided into multiple sub-blocks to speed up computation even if there is no free space inside the block. The dividing, or split lines between sub-blocks go through layout features. Therefore, the sub-blocks are not independent subproblems like blocks in the divide-and-conquer method. Interactions between sub-blocks are important. Figure 9 illustrates that optimal solutions for the sub-blocks cannot be combined easily to arrive at the solution for the whole block. Interactions between

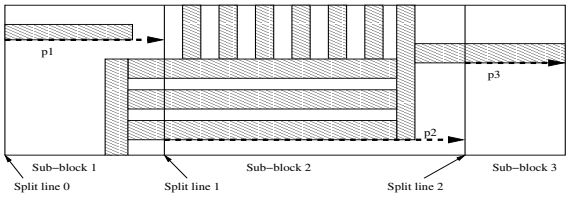

Figure 9: A block splitted into three sub-blocks. Dashed lines p_1, p_2 and p_3 are the shortest paths for the sub-blocks, but p_1 in sub-block 1 is not part of shortest path for the whole block.

neighboring sub-blocks can be modeled by the location a cut line intersects the split line. There is a finite number of locations a cut line can go through a split line. Dynamic programming (DP) can be employed to compute the path for the whole block after costs for all paths between any two possilbe locations on adjacent split lines are calculated.

Complexity of dynamic programming method is directly proportional to the number of possible locations a cut line crosses a split line. In the worst case, if all horizontal edges in all sub-blocks from the Hanan grid graph are allowed to intersect the split line, the DP problem is the same as running Dijsktra's algorithm on the whole block. One reasonable heuristics to reduce the number of possible crossing locations is to restrict them to segments on the split line that are outside the rectangles, since crossing inside of a rectangle is unlikely in an optimal solution. These segments become the sources and destinations of possible paths through a sub-block. Figure 10 shows all pairs of sources and destinations in each sub-block based on the large block from Fig. 9. The dynamic pro-

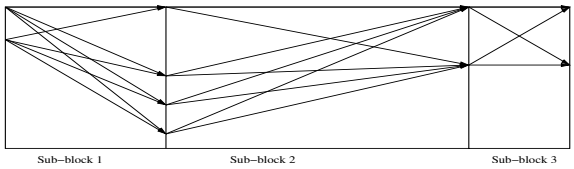

Figure 10: Paths in dynamic programming. Each line with an arrow represents a path through the sub-block it is in.

gramming method then goes from left to right and selects the best source and destination pair for the sub-blocks in sequence. Formally, suppose there are n split lines l_0, l_1, ... l_{n-1} on the tape with l_0 being the left side of the tape, and there are m nodes v_{i1}, v_{i2}, ... v_{im} on the split line l_i. The cost of each path between source v_s and destination v_t is $dis[v_s, v_t]$. Let $\sigma[v]$ denote the source of destination v, and let $d[v]$ denote the cost to arrive at node v, then the dynamic programming method is

Algorithm DP Mask-Cut-EPL
1. Initialize each node v on line l_0 with $\sigma[v] = v$ and $d[v] = 0$
2. Initialize all nodes v not on line l_0 with $d[v] = \infty$
3. **for** $i = 1$, $i < n$, i++ **do** {loop thorugh the sub-blocks}
4. **for** each destination node v on l_i **do**
5. **for** each source node v_s on $l_{(i-1)}$ **do**
6. **if** $d[v_s] + dis[v_s, v] < d[v]$ **then**
7. $\sigma[v] = v_s$, $d[v] = d[v_s] + dis[v_s, v]$
8. **end if**
9. **end for**
10. **end for**
11. **end for**

With the heuristics of only allowing crossings at segments outside rectangles, the DP method speeds up computation at the expense of possible sub-optimal solution. The DP method also requires calculation of multiple paths from sources to destinations within a sub-block. The DP method is the same as the divide-and-conquer method if all split lines are in free spaces so that each sub-block has only one source and destination.

4. Experimental Results

All three approaches are applied to data from a real industry design. The data is from the active layer of a test layout. The input data contains 21464 rectangles inside a tape of 50μm wide and 25.6mm long. Dummy fills on the active layer are not included, since stitching error on dummy shapes does not cause problems, or, dummy shapes can be added around the cut lines later. The number of rectangles in the experiment is much smaller than those on a production layout, and of all the layers in a layout, the number of rectangles on a typical active layer is smaller than those for layers of gate and lower metal lines, of the same order for layers of upper metal lines, and larger than the layers of vias and contacts.

The algorithms are implemented in C under operating system Solaris 2.8 with 64-bit C compiler support. Values for α and β in Eq.(1) in the implementation are set to length of the tape L and $10^4 * L$, respectively. Dijkstra's algorithm is run for each source and destination pair in every sub-block in the dynamic programming method. Hardware used in the experiment is a Sun Enterprise 4800 server with twelve 900MHz UltraSparcIII CPU's and 96G of

Table 1: Experimental results from a tape with 21464 rectangles

Approach	# of blocks	Avg. # of Vertices/Block	Memory Usage	CPU time
Dijkstra's Algorithm	1	37021581	$>$ 10G	$>$ 3 hr.
Divide and Conquer	549	12065	143M	740 sec.
Dynamic Programming	702	10581	143M	2427 sec.

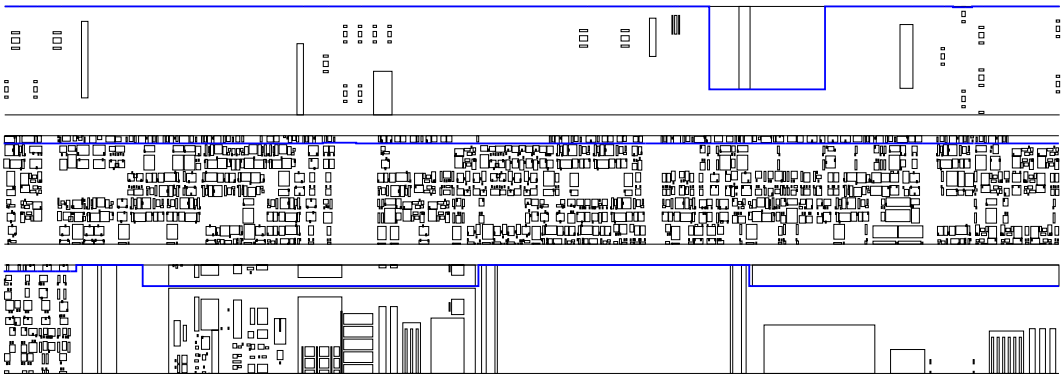

Figure 11: Three portions of the tape from the divide-and-conquer method. The cut line is bold. Each portion is 1/45th of the total wide.

memory. Although only one CPU is used for the runs in this experiment, multiple CPU's can process different tapes in parallel in production.

Table 1 lists the experimental results. Dijkstra's algorithm was stopped manually, because it is using too much memory and running for too long. The divide-and-conquer (DAC) approach is applied without dynamic programming (DP), and run time is excellent. Figure 11 shows three portions of the tape partitioned with the divide and conquer method. Although DP should be used for large blocks *in* the DAC approach, the layout data has much free space and thus relatively small blocks. So in this experiment, the DP method is applied to the whole tape independently without considering free space. The run time for the DP method is much longer than the DAC appoach because of so many more runs of Dijkstra's algorithm for the source and destination pairs in each sub-block. Memory usage is quite small for both DAC and DP approaches. The memory usages are the same because of similar array implementations in both approaches. Solution from the DP method in this experiment is also optimal by avoiding all rectangles not spanning the height of the tape.

5. Concluding Remarks

The mask layout partitioning problem for EPL is first formulated as a graph problem and is optimally solved with a shortest path approach. Divide and conquer and dynamic programming are employed to speed up computation in the shortest path solution. Experimental runs on data from a real industry design showed that the divide-and-conquer approach is very efficient, and the dynamic programming method found an optimal solution in a reasonable amount of run time as well.

Much future work is possible on mask layout partitioning for EPL. Improved weight functions can lead to more accurate modeling. The Hanan grid graph can be simplified by discarding redundent vertices and combining equivalent ones. Other problem formulations and algorithms, such as adjacency graphs and plane-sweep techniques suggested by the reviewers, can be explored.

Acknowledgements

The authors would like to thank Dr. Matthew Thompson, Dr. Jonathan Cobb, and Dr. Kevin Lucas, all of Motorola, for their helpful discussions on EPL. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] International Technology Roadmap for Semiconductors (ITRS) 1999.

[2] K. Lucas of Motorola, private communication. 2002.

[3] T.E. Jewell. "Optical System Design Issues in Development of Projection Camera for EUV Lithography." In *Proc. SPIE*, Vol. 2437, pp. 340-346, 1995.

[4] L. R. Harriott. "SCALPEL Projection Electron Beam for Sub-Optical Lithography." *J. Vac. Sci. Technol. B*, Vol, 15, No. 6, pp. 2130-2135. 1997.

[5] R. S. Dhaliwal, W. A. Enichen, S. D. Golladay, M. S. Gordon, R. A. Kendall, J. E. Lieberman, H. C. Pfeiffer, D. J. Pinckney, C. F. Robinson, J. D. Rockrohr, W. Stickel, and E. V. Tressler. "PREVAIL—Electron Projection Technology Approach for Next-Generation Lithography." *IBM J. Res. & Dev.*, Vol. 45, No. 5, pp. 615-638, Sept. 2001.

[6] M. Nakasuji, *et. al.* "Masks Used in Charged Particle Beam Projecting Apparatus and Method for Dividing Pattern", U.S. patent No. 5,958,626. Sept. 1999.

[7] M. Hanan. "On Steiner's Problem with Rectilinear Distance." *SIAM Journal on Applied Mathematics*, Vol. 14, No. 2, pp. 255-265, 1966.

[8] E. W. Dijkstra. "A Note on Two Problems in Connection with Graphs." *Number. Math.,* Vol. 1, pp. 269-271. 1959.

[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT press, Cambridge, MA. pp. 595-600. 1990.