Benchmarking of Mask Fracturing Heuristics

Tuck Boon Chan[1], Puneet Gupta[3], Kwangsoo Han[1], Abde Ali Kagalwalla[3], Andrew B. Kahng[1,2] and Emile Sahouria[4]
[1]ECE and [2]CSE Departments, University of California, San Diego
[3]EE Department, University of California, Los Angeles
[4]Mentor Graphics Inc.
tuck@vlsicad.ucsd.edu, puneet@ee.ucla.edu, kshan@vlsicad.ucsd.edu, abdeali@ucla.edu, abk@ucsd.edu, emile_sahouria@mentor.com

ABSTRACT

Aggressive resolution enhancement techniques such as inverse lithography (ILT) often lead to complex, non-rectilinear mask shapes which make mask writing extremely slow and expensive. To reduce shot count of complex mask shapes, mask writers allow overlapping shots, due to which the problem of fracturing mask shapes with minimum shot count is NP-hard. The need to correct for e-beam proximity effect makes mask fracturing even more challenging. Although a number of fracturing heuristics have been proposed, there has been no systematic study to analyze the quality of their solutions. In this work, we propose a new method to generate benchmarks with known optimal solutions that can be used to evaluate the suboptimality of mask fracturing heuristics. We also propose a method to generate tight upper and lower bounds for actual ILT mask shapes by formulating mask fracturing as an integer linear program and solving it using branch and price. Our results show that a state-of-the-art prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition can be suboptimal by as much as $3.7\times$ for generated benchmarks, and by as much as $3.6\times$ for actual ILT shapes.

1. INTRODUCTION

Photomasks are one of the most significant contributors to semiconductor manufacturing cost. The use of aggressive resolution enhancement techniques (RETs) has made mask manufacturing extremely expensive and challenging. Moreover, the number of critical masks required for a particular design has increased due to the use of multiple patterning. As a result, controlling the cost of mask manufacturing is urgently needed to sustain benefits derived from Moore's-Law scaling of patterning technologies.

Masks are fabricated using variable-shaped electron beam (VSB) writing tools. These tools directly expose *shots*, i.e., axis-parallel rectangles of any size. Mask fracturing is used to obtain a set of shots from the mask pattern, which can then be input to a VSB tool. Since the total shot count strongly affects mask fabrication time, the key objective of mask fracturing tools is to minimize the number of shots.

Traditionally, mask fracturing has been formulated as rectilinear polygon partitioning, which is a very well-studied problem. Imai and Asano propose an $O(n^{1.5}\log(n))$ algorithm to optimally partition a polygon into the smallest number of rectangles [14]. Since such theoretical approaches are unable to handle additional manufacturing constraints such as minimization of slivers, Kahng et al. [18] propose an ILP based fracturing method. A faster heuristic based on selection of rays from concave corners is also proposed by the same authors [19]. Jiang and Zakhor propose a recursive algorithm to minimize a weighted sum of shot count and slivers [15].

Due to aggressive RET techniques such as ILT, mask shapes are now often curved and non-rectilinear [21] [6]. Fracturing these polygons using traditional methods with acceptable fidelity can dramatically increase the shot count [24]. To manage the shot count of such complex patterns, Chua et al. propose model-based fracturing [7]. Two key features of model-based fracturing distinguish it from traditional mask fracturing:

1. shots are allowed to overlap, which allows greater flexibility in determining shot locations and hence lower shot count; and
2. e-beam proximity effects in VSB mask writers are simulated during the mask fracturing itself to ensure that the final pattern on the mask accurately matches the intended target.

In addition to overlapping shots and proximity effect correction, Galler et al. propose to adjust the dose of each shot independently [12]. Jiang and Zakhor propose an algorithm based on matching pursuit to solve this problem [16]. The use of L-shaped shots for reducing shot count has been suggested by Yu et al. [23]. Elayat et al. [10] analyze the benefits and disadvantages of different mask fracturing strategies. They conclude that, among the alternatives studied, model-based mask fracturing with fixed dose is the most viable candidate for improvement of shot count without significant changes in mask writing tools. Hence, in this work we only focus on the fixed-dose problem.

A consequence of allowing overlapping shots is that model-based mask fracturing becomes similar to the rectilinear covering problem, which is known to be NP-hard [9]. In fact, there is no known constant-factor approximation algorithm for rectilinear covering [2]. Although several recent works on model-based mask fracturing have demonstrated improvements in shot count over traditional partitioning-based approaches [24] [20], the gap between existing methods and optimal solutions remains unclear.

Benchmarking of heuristics used to solve NP-hard EDA problems such as placement [8] and gate sizing [13] enables the development of better methods for solving these problems. The goal of our present work is to enable the benchmarking of model-based fracturing as a foundation for further research towards more effective heuristics. To the best of our knowledge, this is the first work that attempts to benchmark model-based mask fracturing. The key contributions of this work are the following:

- We propose a systematic method to generate benchmarks with known optimal shot count. Using this method, we generate a set of benchmarks to quantify suboptimality of a state-of-the-art prototype [version of a] capability within a commercial EDA tool for e-beam mask shot decomposition.
- We propose an ILP formulation to optimally solve the model-based mask fracturing problem. We then develop a branch

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2014, November 2-6, 2014, San Jose, California, USA
Copyright 2014 ACM 978-1-4799-1071-7/13 ...$15.00.

and price method that, in practice, generates strong upper and lower bounds for benchmarking.

The rest of this paper is organized as follows. Section 2 defines the mask fracturing problem. Section 3 introduces our method for benchmark generation with known minimum shot count. Section 4 describes an ILP-based method to obtain tight upper and lower bounds on the optimal shot count. Section 5 provides experimental results and analysis. Section 6 concludes the paper.

2. MASK FRACTURING PROBLEM

The goal of mask fracturing is to find the minimum number of *rectangular* shots required to construct a mask *target shape*. Although each shot is rectangular, the *e-beam proximity effect* blurs its boundary [7]. As a result, the developed mask pattern is different from the union of rectangular shots. Note that the blurring due to the e-beam proximity effect is significantly smaller than the spacing between different shapes. Hence, each shape in the mask can be fractured independently. *Moreover, to better understand which target shapes are more challenging, the suboptimality of mask fracturing heuristics should be evaluated for individual mask target shapes rather than for the entire mask.*

We define S as the set of all possible candidate shots that could be used to reconstruct the target shape, i.e., the dictionary of candidate shots. S consists of all the different shot sizes that are allowed and all the shifted copies of each shot size. E-beam proximity effect is modeled using a low pass filter, typically a Gaussian or sum of Gaussians [22]. In this work, we model the proximity effect by a single 2D Gaussian low-pass filter, described by Equation (1). However, our proposed methods for benchmarking can be easily extended to handle other proximity effect models.

$$K(x,y) = \begin{cases} \frac{1}{F}\exp^{-\frac{x^2+y^2}{\sigma^2}} & \text{if } -3\sigma \leq \sqrt{x^2+y^2} \leq 3\sigma \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Here, x and y are the coordinates of a particular point on the mask, which we refer to as a pixel ($p(x,y)$). $K(x,y)$ is the kernel function of the Gaussian filter, F is a normalization factor and σ is a parameter which characterizes the spreading of the e-beam. For any rectangular shot s, the intensity at a pixel can be computed by convolving the *ideal rectangular function* ($\psi(\hat{x},\hat{y})$) [5] with the kernel function. That is,

$$I(x,y,s) = K(x,y) \otimes \psi((x-x_{c,s})/W_s, (y-y_{c,s})/H_s)$$

$$\psi(\hat{x},\hat{y}) = \begin{cases} 1 & \text{if } |\hat{x}| < 0.5 \text{ and } |\hat{y}| < 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $I(x,y,s)$ represents the intensity at pixel $p(x,y)$ due to the shot s. W_s and H_s are the width and height of the shot. $x_{c,s}$ and $y_{c,s}$ are the x and y coordinates of the center of the shot. In this paper, all dimensions are in wafer scale.[1]

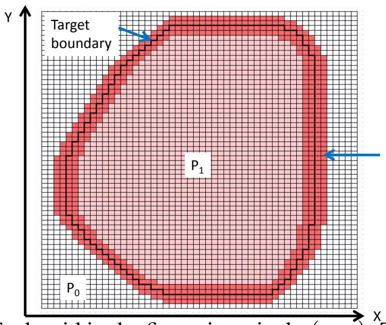


Figure 1: Each grid in the figure is a pixel $p(x,y)$. The thick black line is the target boundary. In this figure, the CD tolerance is $\gamma = 2nm$ and the grid size is $1nm \times 1nm$. $p(x,y) \in P_d$ if $p(x,y)$ is within $2nm$ of the target boundary.

[1]Typically, mask scale is $4\times$ wafer scale.

We model the e-beam resist using a constant-threshold model with threshold value of R_t. Any pixel ($p(x,y)$) on the mask will be exposed if and only if the total intensity at that pixel resulting from all shots exceeds the resist threshold R_t.[2]

As shown in Figure 1, we divide the set of pixels on the mask into three disjoint sets: P_1, P_0 and P_d. P_d is the set of pixels within a given critical dimension (CD) tolerance, γ, of the boundary of the target shape. We define P_1 as the set of the pixels within the closed boundary of the shape which do not belong to P_d. The pixels in P_1 must have intensity greater than or equal to R_t. Similarly, we define P_0 as the set of the pixels outside the target shape which do not belong to P_d. The pixels in P_0 must have intensity less than R_t.

The mask fracturing problem is formally defined as follows.
Goal: Minimize the total number of mask shots $N = |S_{min}|$.
Inputs: Mask target shape, set of all candidate shots S, R_t, σ, γ.
Outputs: Set of rectangular shots, S_{min}.
Constraints:

$$\sum_{s \in S_{min}} I(x,y,s) \geq R_t \text{ if } p(x,y) \in P_1$$

$$\sum_{s \in S_{min}} I(x,y,s) < R_t \text{ if } p(x,y) \in P_0$$

The mask writing process may also require additional constraints to avoid resist over-heating and minimize CD variation. In this work, we do not consider the imposition of maximum intensity constraints to model resist over-heating, since the over-heating is an effect at length scales on the order of microns [11] [3].

3. GENERATION OF BENCHMARKS WITH KNOWN OPTIMUM

The goal of our benchmark generation method is to construct target shapes for which the minimum shot count is known. Our benchmark generation method is based on the key observation that there is a set of *boundary segments*[3] each of which requires at least two shots in *any* fracturing solution. Here we use B^n to denote the set of all boundary segments such that each boundary segment $b^n \in B^n$ requires at least n shots. For example, the union of green and red lines in Figure 5 is a boundary segment b^2.

In our benchmark generation method, we first use exactly two shots to generate a target shape which contains a boundary segment b^2. By the definition of b^2, we need at least two shots to produce the boundary segment. Since we use exactly two shots to generate the target shape, our solution is optimal and the minimum shot count is two. To extend the target shape, we then add a new shot adjacent to one of the existing shots. We select the location of the new shot such that there is a new b^2 in the extended target shape. Note that we only increase the total shot count by one (and reuse an existing shot) to produce the new b^2 which requires two shots. Because the extended target shape cannot be produced by stretching or shifting the shots in the previous solutions (i.e., at least one more shot is required), the solution corresponding to the extended target shape remains optimal with respect to shot count.

In the remainder of this section, we describe the details of our benchmark generation method and prove that a generated target shape has known minimum shot count.

3.1 Boundary Segment

To determine the set of boundary segments which require at least two shots, we analyze the relationship between straight/concave boundary segments and the image produced by a shot. We do not analyze the case of convex boundary segments because it is not used in our benchmark generation.

[2]The exposed pixels will form the mask shape.
[3]A boundary segment is a contiguous part of the boundary of a target shape.

Straight boundary segment. Since a mask shot must be isothetic, a single mask shot cannot produce a long straight boundary at an angle (θ) which is not a multiple of $90°$.[4] Figure 2 shows a straight boundary segment b_{seg} (black solid line) at an angle θ. The dashed lines parallel to b_{seg} are the inner and outer boundaries. The inner (resp. outer) boundary is obtained by shrinking (resp. expanding) the target boundary towards the inside (resp. outside) of the target shape by the value of γ. To produce the straight boundary using a single shot, we must place a corner of the shot close to b_{seg}. The longest straight boundary covered by the single shot is the length ($L^{\theta}_{lin}(W, H)$) between the crossing points (blue cross marks in Figure 2) of the inner target boundary and the image boundary.[5] To maximize the coverage of a single shot, we must shift the shot and therefore the image boundary to touch the outer boundary as shown in Figure 2. The shot must not be shifted beyond the outer boundary because $I(x, y, s)$ must be less than R_t for all pixels in P_0.

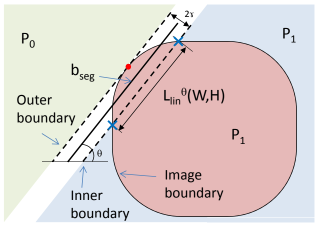


Figure 2: Definition of the length $L^{\theta}_{lin}(W, H)$ of a straight-line target boundary covered by a single shot.

Concave boundary segment. Figure 3(a) shows a concave boundary b_{seg} and its inner (b_{seg_in}) and outer (b_{seg_out}) boundaries. For a concave target boundary, the maximum boundary length covered by a single shot is defined by the straight line between the points of intersection between b_{seg_in} and the shot image boundary (i.e., the blue cross marks in Figures 3(a) and 3(b)). From the straight line between the points of intersection, we define a "virtual" straight line (b_{vir}) and its inner (b_{vir_in}) and outer (b_{vir_out}) boundaries. Note that because of the concavity of b_{seg}, any point along b_{vir_in} is always closer than b_{seg_in} to the point that touches the target boundary (i.e., p_c in Figure 3(a)). Thus, b_{vir_out} is always in P_0, outside the boundary of the shot image. This means that the shot and its corresponding image can be shifted until the shot image boundary touches b_{vir_out} as shown in Figure 3(b). As a result, *the length of the virtual straight line, which is the same as $L^{\theta}_{lin}(W, H)$ at the same θ, is always larger than the length $L^{\theta}_{con}(W, H)$ of the concave target boundary.*

Maximum length covered by a shot. As mentioned above, the rounded corner of a single shot image determines the maximum length covered by the shot. As the shot size increases, the corner rounding due to the e-beam proximity effect saturates. As a result, the $L^{\theta}_{lin}(W, H)$ does not change further with respect to the shot size. For example, Figure 4 shows that for shot sizes larger than $14nm \times 14nm$, the corresponding $L^{\theta}_{lin}(W, H)$ remains the same, and is the maximum $L^{\theta}_{lin}(W, H)$ value over all shots. Therefore, we can calculate the L^{θ}_{max} by increasing W and H iteratively, and stopping when L^{θ}_{max} does not increase.

$$L^{\theta}_{max} = \max_{s \in S}\{L^{\theta}(W_s, H_s)\} \tag{3}$$

Since $L^{\theta}_{con}(W, H) < L^{\theta}_{lin}(W, H)$ for any shot s, the maximum value of $L^{\theta}_{lin}(W, H)$ is an upper bound on $\max_{s \in S}\{L^{\theta}_{con}(W_s, H_s)\}$.

[4]ILT masks can have non-orthogonal target shapes, especially if methods such as *level set* are used for ILT [21].

[5]W and H correspond to the width W_s and height H_s of the shot s under consideration.

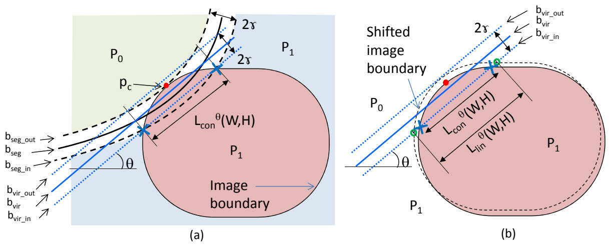


Figure 3: (a) Definition of the length $L^{\theta}_{con}(W, H)$ of a concave target boundary covered by a single shot. (b) Comparison of the lengths covered by a single shot for concave vs. straight-line target boundaries.

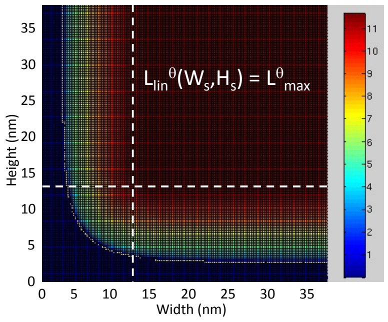


Figure 4: $L^{\theta}_{lin}(W_s, H_s)$ for different shot sizes with $\{\sigma = 6.25nm, \gamma = 1nm, \theta = 45^o\}$ (wafer scale). This figure is obtained by enumerating all possible shot sizes.

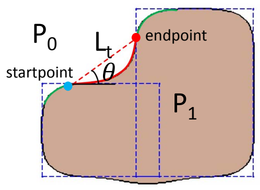


Figure 5: L_t is the Euclidean distance between the startpoint and the endpoint on the target boundary, provided that the target boundary from the startpoint to the endpoint is concave or a straight line.

LEMMA 3.1. *For a mask fracturing problem with finite γ and σ, if a target boundary segment is a straight line or concave shape with length L_t (defined in Figure 5) larger than L^{θ}_{max}, more than one mask shot is required to pattern the target boundary segment.*

PROOF As mentioned above, the corner of the image produced by a shot does not change beyond a certain shot size, and there exists an L^{θ}_{max} for straight boundary which is also the upper bound for concave target boundary. By definition, L^{θ}_{max} is the maximum length on the target boundary which can be covered by a single shot. Therefore, when a target boundary segment has length $L_t > L^{\theta}_{max}$, we require more than one shot to produce the boundary segment. Note that to check whether a boundary segment has length $> L^{\theta}_{max}$, it suffices to calculate the L_t for all combinations of startpoint and endpoint along the boundary. □

3.2 Construction of a Target Shape

We now describe a systematic method to construct a target shape with known minimum shot count. We first construct a b_{seg} using two shots by placing the second shot to the top right of the first shot as shown in Figure 6. We define the top left boundary (e.g., the union of green and red lines in Figure 6) as the *main boundary* (b_{main}).[6] By placing the second shot far enough from the first shot, we create a *critical boundary segment* $b_{cri} \in B^2$, which is part of the b_{main}. The b_{cri} is a straight line or a concave segment

[6]b_{main} is at the top left boundary because we place the next shot to the top right of previous shots.

with length L^{θ} larger than L^{θ}_{max} (Lemma 3.1). Note that although there can be many boundary segments $\in B^2$, only those overlapping with b_{main} are considered as the critical boundary segments. For example, the yellow boundary segment in Figure 6, while an element of B^2, is not considered to be a b_{cri} because it does not overlap with b_{main}.

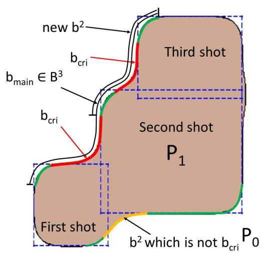


Figure 6: Example of benchmark generation with three shots. b_{main} is the union of green and red lines and contains two b_{cri}.

LEMMA 3.2. *Given a boundary segment b^n of a target with $n-1$ critical boundary segments, and its corresponding shots, we can add a shot to obtain b^{n+1} with an optimal $(n+1)$-shot solution if the addition satisfies the following conditions:*

(i) Adding a shot does not affect the critical boundary segments of b^n.

(ii) b_{main} of the new target shape is continuous.

(iii) There is a b^2 in the b_{main} of the new target shape which cannot be made by extending the shots which produce b^n without altering the critical boundary segments of b^n.

PROOF Since there are $n-1$ critical boundary segments in b^n and the newly added shot does not affect the critical boundary segments in b^n, the new target shape still requires n shots for the $n-1$ critical boundary segments. To create a b^2 in the new boundary which cannot be made by extending shots which produce b^n, we need exactly one more shot. Thus, the new target shape has a boundary segment b^{n+1}. $\square$

Based on Lemma 3.2, we then add a shot at the top right of the existing target shape. This ensures that we have a continuous b_{main}. Moreover, the top-left coordinate of the newly added shot is selected such that there is a b^2 in the new b_{main}. Since the new b^2 is always at the top of the target shape, it cannot be made by extending previous shots unless the existing critical boundary segments are altered. Also, placing the shot at the top right does not affect the existing critical boundary segments. By adding $n-2$ shots to the target shape generated by two shots, we can obtain a target shape $\in B^n$ based on Lemma 3.2.

An important property of our method is that the critical boundary segments are defined only by the top-left coordinates of the shots. Therefore, we may freely place the bottom-right coordinates of the shots to create different target shapes as long as they do not affect the critical boundary segments.

3.3 Merging Target Shapes

LEMMA 3.3. *Given two target shapes with critical boundary segments $b_a \in B^{n_a}$ and $b_b \in B^{n_b}$, which have, respectively, $n_a - 1$ and $n_b - 1$ critical boundary segments, we can merge b_a and b_b by stretching a shot to create $b_c \in B^{n_a+n_b-1}$ if the following conditions are satisfied:*

(i) The stretched shot must not alter the critical boundary segments in b_a or b_b.

(ii) The stretched shot must merge a shot from b_a with a shot from b_b.

(iii) The non-stretched shots in b_a must be far apart from or misaligned from the non-stretched shots in b_b so that any two non-stretched shots cannot be merged to reduce the number of shots.

PROOF Since stretching the shot does not alter the critical boundary segments in b_a and b_b, we need at least n_a shots for target shape b_a and n_b shots for the target shapes b_b. Since the merged b_c contains both b_a and b_b, which share one shot, b_c requires $n_a + n_b - 1$ shots. Therefore, b_c belongs to $B^{n_a+n_b-1}$. $\square$

The first condition in Lemma 3.3 imposes a tight constraint on merging the target shapes generated by the method described in Section 3.2. This is because we can only stretch a shot by moving the lower right corner of the shot in either the rightward and/or downward direction, such that the critical boundary segments are not affected. However, stretching a shot of a target shape to the right and/or down directions will affect the critical boundary segments on the other target shape. This problem can be solved by rotating the target shapes before merging them.

LEMMA 3.4. *A b^n rotated by 90° is still an element of B^n.*

PROOF After applying Gaussian blur, the intensity of a shot is symmetric about the x- and y-axes. Therefore, rotating a target shape by a multiple of 90^o does not affect the number of shots. As a result, if any boundary segment b is in B^n, the boundary segment $b' = b$ rotated by 90^o (or any multiple of 90°) is also in B^n. $\square$

Figure 7 shows an example in which we use Lemmas 3.3 and 3.4 to merge a target shape and its rotated copy into a larger and more complex target shape.

Using the incremental target boundary extension (Lemma 3.2) and merging/rotation of optimal target shapes, we can generate a variety of different benchmarks with arbitrary values of optimal shot count.

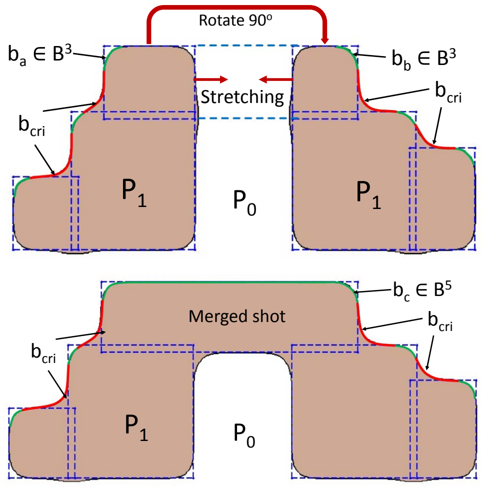


Figure 7: Example of rotating a target shape for merging.

4. ILP-BASED BENCHMARKING

The benchmark generation method proposed in Section 3 cannot generate all possible shapes. To evaluate the suboptimality of fracturing heuristics on any given mask shape, we apply an optimal ILP formulation. The straightforward ILP formulation requires a large number of binary variables, even for small target shapes. As a result, even commercial ILP solvers can run out of memory on high-performance computers. To circumvent this, we propose two strategies, described later in this section: (1) pruning the set of candidate shots, and (2) solving the ILP using branch and price. With these two strategies, we can obtain strong upper and lower bounds on the optimal solution within feasible runtimes. *Note that although the proposed ILP can be used to inspire effective mask fracturing heuristics, the goal of this work is benchmarking. Hence, runtime is important only to the extent of making the method tractable.*

4.1 Optimal ILP Formulation

We define a binary selection variable z_s for each candidate shot $s \in S$, where $z_s = 1$ if shot s is used and $z_s = 0$ otherwise. Then, based on the problem description in Section 2, we may formulate an optimal ILP to solve the fracturing problem as

$$\begin{aligned} \text{Minimize} \quad & \sum_s z_s \\ \text{subject to} \quad & \sum_s \{z_s \cdot I(x,y,s)\} \geq R_t, p(x,y) \in P_1 \\ & \sum_s \{z_s \cdot I(x,y,s)\} < R_t, p(x,y) \in P_0 \end{aligned} \quad (4)$$

Clearly $|S|$ can be very large even for small target shapes. For a target shape with a bounding box of $T_X \times T_Y$, if the minimum and maximum allowed shot sizes are m and M respectively, and the shot granularity is ΔT, then the size of the set of candidate shots would be $\left(\frac{(M-m)}{\Delta T}\right)^2 \cdot (T_X - \frac{M+m}{2}) \cdot (T_Y - \frac{M+m}{2})$. Even for a small post-ILT contact shape as shown in Figure 8(a) ($m = 13$, $M = 55$, $T_X = T_Y = 60$, $\Delta T = 1$), the number of candidate shots is $1.19M$.

Attempting to solve such a large ILP, even with commercial solvers, is very challenging due to long runtime and large memory usage. In fact, the *CPLEX v12.5* solver [26] runs out of memory when we attempt to solve the instance of Figure 8(a) on an Intel Xeon L5420 server with 128GB RAM.

4.2 Pruning Candidate Shot Dictionary

Reducing $|S|$ can significantly help in making the above ILP tractable for benchmarking. Here we highlight two simple rules that can be used to reduce $|S|$:

1. For any candidate shot s, if there exists a pixel $p(x,y) \in P_0$ such that $I(x,y,s) \geq R_t$, then s can be removed from the set S. This pruning criterion obviously does not affect optimality because any candidate shot that satisfies this condition cannot be a part of a feasible solution of the ILP. Depending on the specific target shape, this pruning strategy can significantly reduce $|S|$.

2. If a candidate shot s is inside the target shape and none of its four edges are close to the target boundary, then we remove s from set S. If s is a part of the optimal solution, then we can replace s with a larger shot that covers s and has at least one boundary close to the edge of the target shape, without affecting the optimality of the solution.

An interesting side-effect of pruning candidate shots is that the LP relaxation of the ILP becomes a stronger lower bound for the optimal shot count. After applying these pruning rules to the contact shape of Figure 8(a) we can reduce $|S|$ to $591K$ for a Gaussian proximity effect model ($\sigma = 1nm$). Using *CPLEX v12.5* solver, the final optimal shot count is just four (shown in Figure 8(b)).

Note that the reduction in $|S|$ due to these pruning rules depends strongly on the specific target shape and the e-beam proximity effect model. For many target shapes, the number of variables even after pruning could be more than 10^6, making it difficult to solve the problem efficiently with commercial ILP solvers.

4.3 Branch and Price Method

Branch and price (B&P) is a well-known method for solving large ILPs [4]. The key feature that distinguishes B&P from typical ILP solvers is that the LP relaxation at each node of the branch and bound tree is solved using column generation. To solve the LP relaxation, which contains too many variables to handle efficiently, a reduced master problem (RMP) containing only a small subset of the variables is solved first. To confirm the optimality of this RMP, a separate pricing subproblem is solved to find any new variables that must be inserted back into the RMP. If no variable is found by the pricing subproblem, then the LP relaxation is optimal and branching can be done to obtain the integral solution to the original ILP.

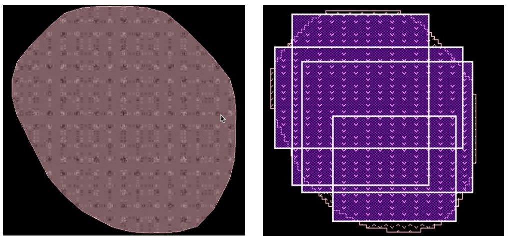

(a) (b)

Figure 8: (a) ILT mask target shape and (b) optimal mask fracturing solution obtained from ILP.

The selective insertion of variables based on the pricing subproblem in B&P means that most variables are never inserted into the LP relaxation. As a result, the LP relaxation solver does not consume too much memory. This is the main reason why we choose to apply this technique to solve the ILP described in Equation (4). The runtime of B&P is known to be limited by the pricing subproblem for most problems [4]. Hence, we propose a novel pricing mechanism comprising a fast, approximate pricer and a slower, optimal pricer.

The goal of the pricing subproblem is to identify additional variables that should be inserted into the RMP. For the mask fracturing problem, let λ_p^* be the optimal value of the dual variable corresponding to the CD constraint (Equation 4) at pixel $p(x,y) \in P_1 \cup P_0$, obtained after an iteration of the RMP. The pricing subproblem (derived from the dual of the RMP) reduces to finding a new candidate shot s such that $\sum_p I(x,y,s) \cdot \lambda_p^* \leq -1$. This candidate shot must also satisfy the pruning rules discussed above. Moreover, additional constraints imposed by the branching rules of the branch and bound tree must be met. The *reduced cost* of any candidate shot s is given by $R_s = 1 + \sum_p \{I(x,y,s) \cdot \lambda_p^*\}$. For the sake of brevity, we shall refer to any candidate shot that has $R_s \leq 0$ and satisfies all the pruning and branching constraints as an *insertable candidate shot* (ICS).

To ensure that the LP relaxation is solved optimally, the pricing subproblem must guarantee that no ICS exists. If there are several ICSs, the pricing subproblem only needs to find a subset of all the ICSs in an iteration. The maximum number of candidate shots that are inserted in each pricing iteration can be tuned to improve the convergence of B&P. In this work, we limit the maximum number of variables that can be inserted in each iteration to $N_C = 500$.

One simple strategy to solve the pricing problem is to iterate over all possible sizes and locations of candidate shots and insert any shot that has a negative R_s and satisfies pruning and branching rules. To improve the efficiency of this naive pricing strategy, we carefully analyze the dual variables of the RMP. Based on the well-known Karush-Kuhn-Tucker (KKT) conditions, we note the following key features of the dual variables:

1. Due to complementary slackness, $\lambda_P^* \neq 0$ if and only if $\sum_s \{z_s \cdot I(x,y,s)\} = R_t$. Since this is likely to occur only close to the boundary of the target shape, λ_p^* is nonzero only for a small number of pixels that lie very close to the target boundary. We shall refer to the set of pixels with nonzero dual values as *dual points*.

2. To ensure dual feasibility, $\lambda_p^* \geq 0$ for $p(x,y) \in P_0$ and $\lambda_p^* \leq 0$ for $p(x,y) \in P_1$. This implies that all negative dual points (P_{neg}) are located inside the target shape.

That negative dual points are sparse and are located close to the target shape boundary is illustrated in Figure 9 for a particular pricing iteration of a target shape. Based on this insight, we propose two pricing strategies to effectively find ICSs, as we now describe.

4.3.1 *Fast Pricer*

The basic idea behind the fast pricer is to look for ICSs in the vicinity of $p(x, y) \in P_{neg}$ because if any candidate shot s has negative reduced cost, then s must be located such that it covers or is close to at least one negative dual point. The steps involved in finding ICSs are summarized in Algorithm 1.

Algorithm 1 Fast Pricer Heuristic

```
Input: Target shape $T$, and list of pixels with negative dual values $P_{neg}$.
Output: Set of candidate shots $S_D$ that must be inserted into the RMP.
1: for all $p(x, y) \in P_{neg}$ do
2:     Draw vertical/horizontal line from $(x, y)$ to find $y_{low}$, $y_{high}$, $x_{low}$ and
       $x_{high}$ (illustrated in Figure 9).
3:     Find all candidate shots in vicinity of $(x, y)$ that satisfy Equation (5) below.
4:     Insert (up to $\frac{N_C}{|P_{neg}|}$) candidate shots that satisfy reduced cost, pruning and
       branching constraints to $S_D$.
5: end for
```

$$
\begin{aligned}
x_{low} - \alpha \le x_{bl} \le x + \beta \quad &, \quad x - \beta \le x_{tr} \le x_{high} + \alpha \\
y_{low} - \alpha \le y_{bl} \le y + \beta \quad &, \quad y - \beta \le y_{tr} \le y_{high} + \alpha
\end{aligned} \tag{5}
$$

where x_{bl} (bottom left x-coordinate), y_{bl} (bottom left y-coordinate), x_{tr} (top right x-coordinate) and y_{tr} (top right y-coordinate) correspond to the coordinates of a candidate shot under consideration. α and β are distance margins that depend on the e-beam proximity effect model and resist threshold. α is the maximum distance outside the target boundary where a candidate shot corner can lie without exposing any pixel outside the boundary. β is the maximum distance outside the shot at which the intensity is nonzero.[7]

The intuition behind constraining x_{bl}, y_{bl}, x_{tr} and y_{tr} as shown in Equation (5) is that such candidate shots will have nonzero intensity at the negative dual point under consideration and are likely to obey the first pruning rule (not exposing any pixel in P_1).

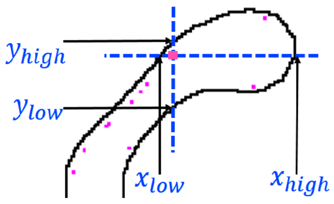


Figure 9: Illustration of negative dual points (pink dots) for part of a target shape. Coordinates x_{low}, x_{high}, y_{low} and y_{high} (points of intersection of blue dashed lines with target shape boundary) for a particular negative dual point (point of intersection of the two dashed lines) are also shown.

4.3.2 *Optimal Pricer*

Although the pricing heuristic we described above is effective in identifying most ICSs which can be inserted into the RMP, it does not guarantee that if no ICS is found, then there does not exist any ICS. Hence, if the heuristic fails to find any ICS, we call the optimal pricer that iterates over all the different shot sizes and locations and inserts the first N_C candidate shots which satisfy all the pruning rules, branching constraints and have negative reduced cost. The runtime of this strategy is dominated by the computation of intensity (which is required to compute the reduced cost and check the pruning rules) for such a large number of candidate shots. To improve this runtime, we compute the intensity of only those candidate shots that satisfy the following two filtering criteria:

- The smallest distance between the candidate shot and the entire set of negative dual points must be less than β, otherwise the reduced cost will be positive.
- The distance between each of the four corner points of the candidate shot and the target shape must be less than α, otherwise the shot will violate the first pruning rule.

[7]For the Gaussian proximity effect model we use $\alpha = \sigma$ and $\beta = 2\sigma$.

4.4 Initialization and Overall Summary

In addition to solving the pricing subproblem efficiently, B&P can benefit significantly from a good initial feasible solution. Although B&P is capable of discovering feasible solutions using Farkas pricing [1], it can take many iterations of pricing to do so. In this work, we use the results of a prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition as the initial solution for B&P.

B&P is known to suffer from a "tailing off" effect, where the upper and lower bounds improve very slowly once they are close to the optimum [4]. Since our objective is to evaluate suboptimality, we choose to run B&P with a fixed time limit and report the best upper and lower bounds on the optimal shot count. The overall method we use to solve LP relaxations within B&P is summarized in Figure 10.

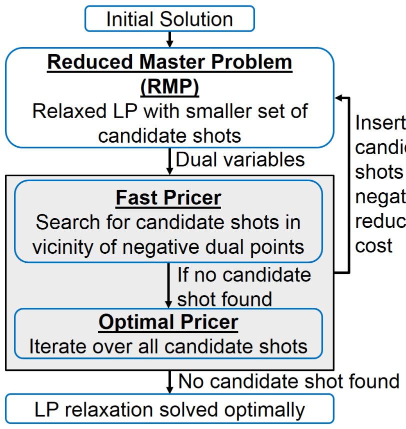

Initial Solution
Reduced Master Problem (RMP)
Relaxed LP with smaller set of candidate shots
Insert candidate shots with negative reduced cost
Dual variables
Fast Pricer
Search for candidate shots in vicinity of negative dual points
If no candidate shot found
Optimal Pricer
Iterate over all candidate shots
No candidate shot found
LP relaxation solved optimally

Figure 10: Flowchart of steps involved in solving LP relaxation for any node in B&P.

5. EXPERIMENTAL RESULTS

Our benchmark generation method and our B&P based suboptimality evaluation method have both been implemented in C++. We use the OpenAccess API to parse layouts [29], Boost Polygon Library to perform polygon operations [25] and Eigen Library to perform matrix operations [27]. To implement B&P, we use the SCIP framework [1], along with *CPLEX v12.5* as the LP solver [26].

In this work, we use a Gaussian e-beam proximity effect model with $\sigma = 6.25nm$. This is consistent with recent work on mask fracturing [16] [17]. We set the CD tolerance $\gamma = 2nm$, and the minimum and maximum dimensions of a shot are $13nm$ and $1000nm$, respectively. Using this setup, we have evaluated the suboptimality of a prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition (denoted as PROTO-EDA in what follows) for three types of mask target shapes:

1. **Arbitrary generated benchmarks (AGB):** We generate five shapes with known optimal shot count using the method described in Section 3.2. These benchmarks are shown in Figure 11(a).
2. **Realistic generated benchmarks (RGB):** Since generated benchmarks can often be unrealistic compared to actual ILT mask shapes, we also generate five mask shapes that look similar to actual ILT shapes with known optimal shot count, again using the method described in Section 3.2. We manually select shot locations so that the generated benchmarks are similar to actual ILT mask shapes. These benchmarks are illustrated in Figure 11(b).

3. **Actual ILT mask shapes:** We apply ILT to benchmark pre-RET layouts from the 2013 ICCAD contest [28], using a commercial EDA tool. From the ILT solutions, we select ten representative mask shapes for evaluation. These benchmarks are illustrated in Figure 12.

For each of the 20 target shapes, we run B&P on an eight-core machine with a time limit of 24 hours. In any branch and bound based search method for integer programs, the upper bound corresponds to the best integral solution that has been discovered so far. The lower bound corresponds to the LP relaxation at a particular level of the branch and bound tree. We report the upper and lower bounds reached by B&P within the set time limit.

Table 1: Comparison of shot count for generated benchmarks with known optimal solution.

Clip ID		Shot Count			
		Optimal	PROTO-EDA	Branch and Price	
				Lower bound	Upper bound
AGB	1	3	6	2	3
	2	16	38	10	34
	3	17	50	1	48
	4	7	21	5	7
	5	3	7	2	3
RGB	1	5	10	3	5
	2	7	26	4	10
	3	5	12	3	5
	4	9	20	6	20
	5	6	15	4	8

We compare the shot count of our generated benchmarks with known optimal solutions to PROTO-EDA in Table 1. For the ten target shapes that we analyze, the suboptimality ranges from $2\times$ (clip AGB-1) to $3.7\times$ (clip RGB-2). It is interesting to note that the suboptimality for the RGB clips is generally higher than for the AGB clips. The AGB-5 testcase shows that the suboptimality can be as large as $2.3\times$ even for a simple target shape. For comparison, we also report the lower and upper bounds obtained from B&P for our generated benchmarks in Table 1. The results show that for testcases AGB-{1,4,5} and RGB-{1,3}, the B&P method can find the optimal solution. However, B&P may not find any better solution within the set time limit (testcase RGB-4).

Note that the generated benchmarks are more wavy (i.e., have high-frequency components in the boundary shape) compared to actual ILT shapes. This could make the suboptimality estimation more pessimistic. However, highlighting scenarios where mask fracturing heuristics perform poorly is important for developing better heuristics.

Table 2: Comparison of shot count for ILT mask shapes obtained from ICCAD-2013 contest layouts for which the optimal shot count is unknown.

Clip ID	Shot Count		
	PROTO-EDA	Branch and Price	
		Lower bound	Upper bound
1	9	3	4
2	21	5	21
3	11	2	3
4	25	7	25
5	17	4	7
6	7	2	3
7	9	3	4
8	20	5	8
9	10	5	10
10	15	4	6

We show the results of our ILP-based suboptimality analysis method for actual ILT mask shapes in Table 2. For all ten benchmark shapes, our method is able to report a lower bound based on LP relaxation.[8] Although this seems trivial, typical LP methods (simplex and barrier methods) run out of memory while trying to solve the LP relaxation of the ILP in Equation (4) for these benchmark shapes. Hence, our B&P based method appears to be enabling to the computation of this lower bound. The gap between the PROTO-EDA tool's shot count and these lower bounds ranges from $2\times$ (clip 9) to $5.5\times$ (clip 3).

[8]The fractional LP relaxation value is rounded up to the next integer to obtain the lower bound.

Since the gap between the optimal solution of an ILP and the LP relaxation can be very large, suboptimality analysis based on the lower bound may be too pessimistic. For seven of the ten ILT mask shapes, our method reports an upper bound which is lower than the shot count reported by PROTO-EDA tool. This implies that better mask fracturing solutions were discovered during the branch and bound search. If we make the optimistic assumption that these solutions are in fact optimal, the suboptimality of the PROTO-EDA tool ranges from $2.2\times$ to $3.6\times$. These results confirm that there is significant room for improving the quality of mask fracturing solutions.

6. CONCLUSIONS

The use of aggressive RET techniques such as ILT, the need for e-beam proximity effect correction, and the use of overlapping shots have transformed mask fracturing into a very challenging computational problem. Although several heuristics have been proposed in the last few years, there has been no systematic study to analyze the quality of solutions. In this work, we propose two methods to evaluate the suboptimality of mask fracturing heuristics. First, we introduce a systematic method to generate a set of benchmarks with known, provably optimal solutions. Second, to evaluate the suboptimality of fracturing heuristics on actual ILT mask shapes for which the optimal solution is unknown, we formulate the mask fracturing problem as an integer linear problem and develop a practical branch and price method to generate tight upper and lower bounds on the optimal shot count.

Our experimental results show that a state-of-the-art prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition has up to $3.7\times$ more shots compared to the optimal solution for generated benchmarks, and has up to $3.6\times$ more shots for ILT mask shapes with unknown optimal solution. These results suggest that there remains considerable opportunity to improve mask fracturing heuristics.

In the future, we plan to improve our benchmark generation methodology (e.g., to create benchmarks that remain challenging even to "adversarial" heuristics that know the benchmark generation strategy), incorporate additional mask manufacturing constraints (e.g., maximum intensity limits) as well as develop an automated benchmark generation method that matches realistic target shapes. We also plan to develop better heuristics based on insights from boundary analysis and ILP solutions. The latest versions of our source code and benchmark suite are available publicly (`http://impact.ee.ucla.edu/maskFracturingBenchmarks`). We hope that this will stimulate further research toward development of improved mask fracturing heuristics.

Acknowledgments

The UCSD and UCLA authors acknowledge the support of the IMPACT+ program (`http://impact.ee.ucla.edu/`). We also thank Pradiptya Ghosh (Mentor Graphics) for valuable discussions.

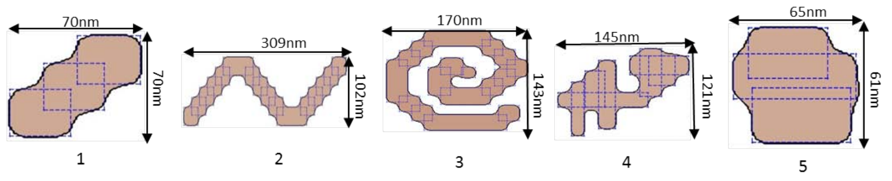
7. REFERENCES

[1] T. Achterberg, "SCIP: Solving Constraint Integer Programs", *Mathematical Programming Computation* 1(1) (2009), pp. 1-41.

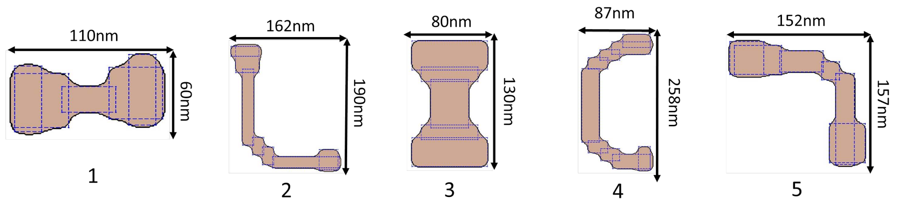
[2] S. Arora, "Approximation Schemes for NP-Hard Geometric Optimization Problem: A Survey", *Mathematical Programming* 97(1-2) (2003), pp. 43-69.

[3] S. V. Babin, A. B. Kahng, I. I. Mandoiu, and S. Muddu, "Resist Heating Dependence on Subfield Scheduling in 50kV Electron Beam Maskmaking", *Proc. SPIE Photomask Technology*, vol. 5130, 2003, pp. 718-726.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, W. P. Martin and P. H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs", *Operations Research* 46(3) (1998), pp. 316-329.



(a) Arbitrary generated benchmarks



(b) Realistic generated benchmarks

Figure 11: Illustration of generated benchmarks with the optimal mask fracturing solution shown in dashed lines (wafer scale).

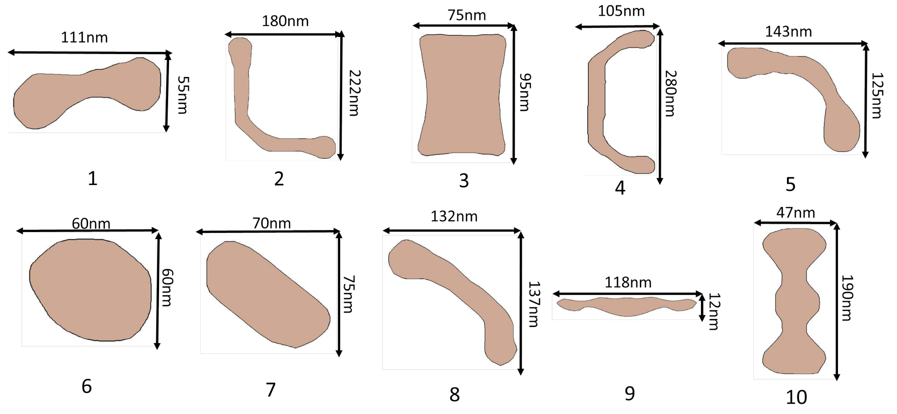


Figure 12: ILT mask shapes obtained after applying inverse lithography to layouts from the ICCAD-2013 contest [28] (wafer scale).

[5] R. Bracewell, "Rectangle Function of Unit Height and Base, PI(x)", in *The Fourier Transform and Its Applications,* McGraw-Hill, 1965.

[6] J. Choi, J. S. Pack, I. K. Shin and C. Jeon, "Inverse E-Beam Lithography on Photomask for Computational Lithography", *J. Micro/Nanolithography, MEMS, and MOEMS* 13(1) (2013), pp. 011003-1-011003-9.

[7] G. S. Chua, W. L. Wang, B. I. Choi, Y. Zou, C. Tabery, I. Bork, T. Nguyen and A. Fujimura, "Optimization of Mask Shot Count Using MB-MDP and Lithography Simulation", *Proc. SPIE Photomask Technology*, vol. 8166, 2011, pp. 816632-1-816632-11.

[8] J. Cong, M. Romesis and X. Min, "Optimality and Stability Study of Timing-Driven Placement Algorithms", *Proc. ICCAD*, 2003, pp. 472-478.

[9] J. C. Culberson and R. A. Reckhow, "Covering Polygons is Hard", *J. Algorithms* 17(1) (1994), pp. 2-44.

[10] A. Elayat, T. Lin, E. Sahouria and S. F. Schulze, "Assessment and Comparison of Different Approaches for Mask Write Time Reduction", *Proc. SPIE Photomask Technology*, vol. 8166, 2011, pp. 816634-1-816634-13.

[11] A. Fujimura, T. Kamikubo and I. Bork, "Model-Based Mask Data Preparation (MB-MDP) and Its Impact on Resist Heating", *Proc. SPIE Alternative Lithographic Technologies III*, vol. 7970, 2011, pp. 797012-1-797012-10.

[12] R. Galler, D. Melzer, M. Boettcher, M. Krueger, M. Suelzle and C. Wagner, "Modified Dose Correction Strategy for Better Pattern Contrast", *Proc. SPIE European Mask and Lithography Conference*, vol. 7545, 2010, pp. 75450F-1-75450F-12.

[13] P. Gupta, A. B. Kahng, A. Kasibhatla and P. Sharma, "Eyecharts: Constructive Benchmarking of Gate Sizing Heuristics", *Proc. DAC*, 2010, pp. 597-602.

[14] H. Imai and T. Asano, "Efficient Algorithms for Geometric Graph Search Problems", *SIAM J. Computing* 15(2) (1986), pp. 478-494.

[15] S. Jiang, X. Ma and A. Zakhor, "A Recursive Cost-Based Approach to Fracturing", *Proc. SPIE Optical Microlithography XXIV*, vol. 7973, 2011, pp. 79732P-1-79732P-18.

[16] S. Jiang and A. Zakhor, "Application of Signal Reconstruction Techniques to Shot Count Reduction in Simulation Driven Fracturing", *Proc. SPIE Photomask Technology*, vol. 8166, 2011, pp. 81660U-1-81660U-14.

[17] S. Jiang and A. Zakhor, "Shot Overlap Model-Based Fracturing for Edge-Based OPC Layouts", *Proc. SPIE Optical Microlithography XXVII*, vol. 9052, 2014, pp. 90520L-1-90520L-19.

[18] A. B. Kahng, X. Xu and A. Zelikovsky, "Yield- and Cost-Driven Fracturing for Variable Shaped-Beam Mask Writing", *Proc. SPIE Photomask Technology*, vol. 5567, 2004, pp. 360-371.

[19] A. B. Kahng, X. Xu and A. Zelikovsky, "Fast Yield-Driven Fracture for Variable Shaped Beam Mask Writing", *Proc. SPIE Photomask and Next-Generation Lithography Mask Technology XIII*, vol. 6283, 2006, pp. 62832R-1-62832R-10.

[20] T. Lin, E. Sahouria, N. Akkiraju and S. Schulze, "Reducing Shot Count Through Optimization-Based Fracture", *Proc. SPIE Photomask Technology*, vol. 8166, 2011, pp. 81660T-1-81660T-13.

[21] L. Pang, P. Hu, D. Peng, D. Chen, T. Cecil, L. He, G. Xiao, V. Tolani, T. Dam, K. Baik and B. Gleason, "Source Mask Optimization (SMO) at Full Chip Scale Using Inverse Lithography Technology (ILT) Based on Level Set Methods", *Proc. SPIE Lithography Asia*, vol. 7520, 2009, pp. 75200X-1-75200X-17.

[22] J. M. Pavkovich, "Proximity Effect Correction Calculations by the Integral Equation Approximate Solution Method", *J. Vacuum Science & Technology B* 4(1) (1986), pp. 159-163.

[23] B. Yu, J.-R. Gao and D. Z. Pan, "L-Shape Based Layout Fracturing for E-Beam Lithography", *Proc. ASPDAC*, 2013, pp. 249-254.

[24] H. R. Zable, A. Fujimura, T. Komagata, Y. Nakagawa and J. S. Petersen, "Writing Wavy Metal 1 Shapes on 22-nm Logic Wafers with Less Shot Count", *Proc. SPIE Photomask and Next-Generation Lithography Mask Technology XVII*, vol. 7748, 2010, pp. 77480X-1-77480X-10.

[25] "Boost Polygon Library", *http://www.boost.org/doc/libs/1_53_0/libs/polygon/doc/index.html*

[26] "CPLEX v12.5", *http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/*

[27] G. Guennebaud, B. Jacob et al., "Eigen v3", 2010, *http://eigen.tuxfamily.org*

[28] S. Banerjee, "ICCAD 2013 Contest on Mask Optimization", *http://cad_contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2013/problem_c/*

[29] "OpenAccess API", *http://www.si2.org*