Fracturing-aware Curvilinear ILT via Circular E-beam Mask Writer*

Xinyun Zhang
Chinese University of Hong Kong

Su Zheng
Chinese University of Hong Kong

Guojin Chen
Chinese University of Hong Kong

Binwu Zhu
Chinese University of Hong Kong

Hong Xu
Chinese University of Hong Kong

Bei Yu
Chinese University of Hong Kong

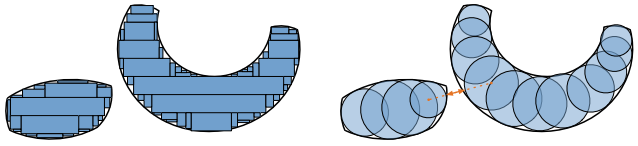
Abstract

Inverse lithography technology (ILT) plays a crucial role in optical proximity correction, tending to generate curvilinear masks for optimal process windows. Traditional curvilinear mask manufacturing involves fracturing into rectangles, requiring expensive mask write times. A novel E-beam mask writer that writes variable radius circles per shot significantly reduces the shot count for curvilinear masks. To exploit this mask writer's benefits, we present two methods to generate circular fracturing-aware masks. The first one converts pixel-based masks from existing ILT methods into circle-based masks using predefined rules. The second one integrates circular constraints into the ILT process, generating circle-based masks directly via optimization. Extensive experimental results validate both approaches' effectiveness.

1 Introduction

As technology nodes shrink, lithography, the process of transferring circuit designs onto wafers, experiences deviations between printed shapes and designs due to optical effects. Resolution enhancement techniques (RETs), such as optical proximity correction (OPC), are employed to improve mask fidelity and printability. OPC, either model-based or inverse lithography-based (ILT), modifies mask patterns and inserts assist features to compensate for optical effects. Model-based methods [1] suffer from a limited solution space and poor performance, while ILT-based methods [2–6] optimize masks through lithography simulation, offering pixel-level flexibility and generating curvilinear masks with superior process window and critical dimension uniformity [7]. ILT is now widely used for 193i and EUV mask patterns across different nodes.

Despite the enhanced quality of curvilinear masks achieved through ILT [8–10], their manufacturing presents challenges. Traditional Variable Shaped-Beam (VSB) machines utilize varying sizes of rectangular shapes, requiring Mask Data Preparation (MDP) to fracture these shapes into non-overlapping rectangles or VSB shots for printability. However, the Manhattanization-based fracturing of curvilinear shapes, especially small sub-resolution assist features (SRAFs), results in a substantial increase in the number of shots (Figure 1(a)). This increase not only escalates mask manufacturing costs but also affects mask yield, thereby impeding high volume wafer manufacturing [7]. Additionally, the rectangular fractured mask shapes are prone to writing errors due to short-range e-beam blur in the 20-40 nm range [7]. These factors limit the broader application of curvilinear masks, despite their potential for superior mask quality.



(a) Rectangular Fracturing (b) Circular Fracturing

Figure 1: Fracturing pattern comparison. Circular fracturing requires much fewer shots for curvilinear masks compared with rectangular fracturing and is MRC-friendly.

To address the aforementioned issues, [7] introduces a new circular e-beam mask writer that writes a variable-radius circle for each shot and allows overlapping writing, as shown in Figure 1(b). This writer is more suitable for curvilinear mask manufacturing, as it requires much fewer shots than rectangular-based ones. Besides, the fractured curvilinear masks are also mask rule checking (MRC)-friendly since we can effortlessly check the distances between the circular shots with their positions and radii. Therefore, we are interested in developing ILT methods based on this writer to fully exploit its advantages. With this in mind, we define a novel research problem, called circular fracturing-aware OPC (CFAOPC), which aims to produce masks fractured into overlapped circles from a target pattern. CFAOPC differs from traditional OPC in two main aspects. First, while traditional OPC focuses solely on mask generation, CFAOPC takes into account both mask generation and fracturing, resulting in well-fractured masks represented by sets of shots, instead of pixel-level masks. Second, CFAOPC does not assume rectangular fracturing like most traditional OPC methods [11], but rather uses overlapped circular fracturing.

We propose two ILT-based solutions for CFAOPC. The first rule-based method integrates with existing state-of-the-art (SOTA) pixel-based ILT methods [4, 10, 11], extracting the skeleton of each pixel-level mask shape and filling it with evenly spaced circles. The second is a two-stage optimization-based method, which generates rough mask shapes and SRAFs in the first stage and optimizes sparse circular representations in the second stage using a novel circle-level ILT optimization method that incorporates circular constraints into the pixel-level ILT process. The radius and positions of the circular

*This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14208021).

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3655926

representations will be optimized directly. We validate the effectiveness of the two proposed methods on the ICCAD-13 contest benchmark [12]. Experimental results demonstrate that both proposed methods reduce the shot count compared to previous ILT methods that use Manhattanization-based fracturing. Furthermore, the optimization-based method outperforms the rule-based method due to the proposed circle-level ILT optimization in terms of mask quality.

To summarize, our contributions are four-fold:

- First, we introduce a new research problem, Circular Fracturing-Aware OPC (CFAOPC), which aims to generate masks fractured into overlapped circles for circular e-beam mask writers [7].
- Second, we present a rule-based method for CFAOPC, integrating with existing SOTA pixel-level ILT methods.
- Third, we propose an optimization-based method for CFAOPC problem that incorporates circular constraints into the pixel-level ILT process and directly optimizes masks using circular representations.
- Fourth, we demonstrate that our methods significantly reduce shot count, with the optimization-based method surpassing the rule-based method in mask quality.

2 Preliminaries and Problem Formulation

2.1 Lithography Simulation

Lithography simulation consists of two parts: a model for how light projects through the mask and a model for how the photoresist reacts to the light. The mask has patterns $\boldsymbol{M}$ that block or allow light to pass through and reach the optical projection system. The system changes the input light intensity distribution into an aerial intensity distribution on the wafer plane. The aerial intensity distribution is often called an aerial image $\boldsymbol{I}$. The process of converting the binary mask image $\boldsymbol{M}$ to the continuous aerial image $\boldsymbol{I}$ can be described by Hopkins' diffraction theory [13]:

$$\boldsymbol{I} = \boldsymbol{H}(\boldsymbol{M}) = \sum_{k=1}^{K} \mu_k |\boldsymbol{h}_k \otimes \boldsymbol{M}|^2, \tag{1}$$

where $\boldsymbol{h}_k$ is the kth optical kernel function, μ_k is its weight, $\otimes$ denotes convolution, and $\boldsymbol{H}$ represents the optical projection model.

A convolution operation in Equation (1) involves two matrices with large sizes, e.g., 2048×2048, which can be very time-consuming. Therefore, the convolution operation is usually implemented by $\boldsymbol{h}_k \otimes \boldsymbol{M} = \texttt{IFFT}(\texttt{FFT}(\boldsymbol{h}_k) \cdot \texttt{FFT}(\boldsymbol{M}))$ to improve efficiency, where $\texttt{FFT}$ and $\texttt{IFFT}$ represent the Fast Fourier Transform (FFT) and inverse FFT operations, respectively.

The photoresist model transfers the aerial image $\boldsymbol{I}$ to the printed/resist image $\boldsymbol{Z}$. It is commonly implemented by comparing the aerial intensity to the photoresist intensity threshold, which can be expressed as:

$$\boldsymbol{Z}(x,y) = \begin{cases} 1, & \text{if } \boldsymbol{I}(x,y) > I_{th}, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where I_{th} is the intensity threshold and (x, y) represents a coordinate on the aerial or resist image.

2.2 Principles of ILT

The target of ILT algorithms is to optimize a mask $\mathbf{M}$ that can generate a printed image $\mathbf{Z}$ as close to the target pattern $\mathbf{T}$ as possible, which can be formulated as:

$$\mathbf{M}^* = \underset{\mathbf{M}}{\operatorname{argmin}} \|\mathbf{Z} - \mathbf{T}\|^2. \tag{3}$$

2.3 Evaluation Metrics

As for mask optimization, we consider domain-specific metrics related to ILT as follows.

Squared L2 loss measures the difference between the nominal resist image and the target image, defined as:

$$L_2(\boldsymbol{Z}_{nom}, \boldsymbol{T}) = \|\boldsymbol{Z}_{nom} - \boldsymbol{T}\|_2^2. \tag{4}$$

PVB evaluates the robustness of the mask against varying process conditions. It is computed by measuring the L2 distance between the printed images from the maximum and minimum process corners, formulated as:

$$\text{PVB}(\boldsymbol{Z}_{max}, \boldsymbol{Z}_{min}) = \|\boldsymbol{Z}_{max} - \boldsymbol{Z}_{min}\|_2^2. \tag{5}$$

Edge placement error (EPE) estimates the geometric distortion of the resist image. To compute the EPE, we sample multiple points on every horizontal and vertical edge of the target shapes. The EPE score is the number of points where the distance from the target shape to the printed pattern is larger than the EPE constraint.

Shot count (#Shot) [3] is the number of basic shapes, i.e., circles, for constructing the mask patterns, which represents the mask manufacturability.

2.4 Problem Formulation

Problem 1 (Circular fracturing-aware OPC)**.** Given a target pattern $\mathbf{T}$, our goal is to obtain a mask $\mathbf{M}$, which is perfectly fractured into a set of overlapping circles with a radius within a proper range, to minimize the squared L2 loss, PVB and the shot count.

3 Rule-based method

Since many efforts have been made to fully exploit the potential in pixel-level ILT [2–4, 10, 14], the motivation of the rule-based method, CircleRule, is to utilize circles to tessellate the curvilinear masks generated by these delicate ILT methods. Besides, CircleRule also serves as a baseline for comparison with our proposed optimization-based method.

To fracture a mask into overlapped circles, the key insight of CircleRule is first to split the mask shapes into several connected regions, then find the skeleton of each region, and finally sample points from the skeleton and tessellate circles at these positions. Let $\mathbf{M}$ be a binarized mask, A be the point set of the pixel coordinates of the effective mask regions, and $C(p, r)$ be the set of points in the circle centered at point p with a radius r. Our target is to find a new shape $\tilde{A}$, which can be well fractured into a group of circles, denoted as $\tilde{A} = \cup_i C(p_i, r_i)$. We further denote the minimum and maximum radius for each shot as $R_{\min}$ and $R_{\max}$, respectively. With a given sample distance m (the distance between two consecutive circles) and a cover rate threshold I, the overall flow of the rule-based method is shown in Algorithm 1.

DFS-based point sampling.. We sample points to span circles after segmenting the mask shape into connected regions and finding the skeleton for each region. The skeleton is a connected curve in the pixel grid with at least one point in the neighbor positions (the eight pixels around each position) for any point on the skeleton.

Algorithm 1 The Rule-based Method for CFAOPC

1: **Input**: Raw mask A, sample distance m, maximum radius $R_{\max}$, minimum radius $R_{\min}$, cover rate threshold I.
2: **Output**: Circular fractured mask $\tilde{A}$.
3: Initialize an empty stack t;
4: $V \leftarrow \emptyset, \tilde{A} \leftarrow \emptyset$;
5: $\{A_1, \cdots, A_n\} \leftarrow$ `findConnectedRegions`(A);
6: **for** $i \in \{1, \cdots, n\}$ **do**
7: $\quad S_i \leftarrow$ `findSkeleton`(A_i);
8: $\quad$ Randomly sample a point p_i in S_i;
9: $\quad$ Push $(p_i, 0)$ to t; ▷ DFS-based point sampling
10: $\quad$ **while** t is not empty **do**
11: $\quad\quad (u, cnt) \leftarrow$ `Pop` t;
12: $\quad\quad$ **if** u not in V **then**
13: $\quad\quad\quad V \leftarrow V \cup u$;
14: $\quad\quad\quad N \leftarrow$ `findNeighborPoints`(u);
15: $\quad\quad\quad$ **for** $n \in N$ **do**
16: $\quad\quad\quad\quad$ **if** $n \notin V$ **then**
17: $\quad\quad\quad\quad\quad$ Push $(n, cnt + 1)$ to t;
18: $\quad\quad\quad$ **if** $cnt \mod m == 0$ **then**
19: $\quad\quad\quad\quad$ **for** $r \in \{R_{\min}, \cdots, R_{\max}\}$ **do** ▷ Circle radius selection
20: $\quad\quad\quad\quad\quad$ cover rate $\leftarrow \dfrac{|C(u,r) \cap A_i|}{|C(u,r)|}$;
21: $\quad\quad\quad\quad\quad$ **if** cover rate $< I$ **then**
22: $\quad\quad\quad\quad\quad\quad \tilde{A} \leftarrow \tilde{A} \cup C(u,r)$;
23: $\quad\quad\quad\quad\quad\quad$ **break**;

Therefore, we can construct the skeleton as a graph, where each point represents a node and is connected to the nodes at the neighbor pixels, as shown in Figure 2(a). Then, we perform a depth-first search-based (DFS-based) point sampling algorithm to sample points as circle centers, as shown from line 10 to line 18 in Algorithm 1. To ensure the sampled points are evenly distributed on the skeleton, we accompany a counter for each iteration and sample the points with a fixed sample distance m.

Circle radius selection. We decide the radius for each circle to fit the mask shape with points sampled from our DFS-based algorithm (see lines 19–23 in Algorithm 1). Specifically, we traverse all possible radii from the minimum to the maximum until the cover rate (defined in line 20 in Algorithm 1) drops to a threshold I, considering the radius value constraints. The cover rate is 1 when the radius r is small, and all circles are enclosed by the mask. The cover rate decreases when r exceeds the mask boundaries, as shown in Figure 2(b). Therefore, the hyper-parameter I is a critical factor in the area of the fractured shape. In our experiment, we find that setting I to 0.9 is a good choice and adopt it in our experiments.

4 Optimization-based method

The optimization-based method, CircleOpt, consists of two stages. The first one is a simple pixel-level initialization aiming at generating the rough mask shapes and SRAFs. The second one is a circle-based optimization that transforms the pixel-level mask into sparse circular representations and performs circle-level ILT for optimal mask quality while maintaining the circular fracturing constraints. The overall flow of CircleOpt is shown in Figure 3.

4.1 Pixel-level Initialization

We perform simple pixel-level ILT as an initialization for the later circle-based ILT process. We adopt the simplest ILT baseline MOSAIC [2] and perform only a few steps to roughly generate some

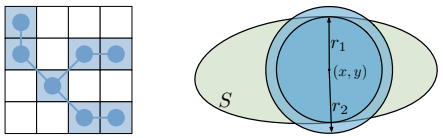


Figure 2: Illustration of the key steps in CircleRule. (a) Construction of the skeleton graph; (b) the spanning process of the circle radius.

good shapes. The loss function and the binary function used in this stage are detailed in the following paragraphs.

Loss function. We take the simplest form of ILT loss function, consisting of a squared L2 loss and a PVB loss, which can be formulated as:

$$L = L_2 + L_{pvb}. \tag{6}$$

The detailed definition of the squared L2 loss and the PVB loss can be found in Equation (4) and Equation (5), respectively.

Binary function. To smoothly generate SRAFs patterns, we adopt a shifted sigmoid function as the binary function, same as [10].

4.2 Circle-based ILT

The key challenge in CFAOPC is to generate masks with the best printability and manufacturability under circular constraints. To address this, we propose a brand new circle-based ILT method, which leverages the advantages of mask quality of ILT and satisfies the circular fracturing constraints at the same time. Specifically, circle-based ILT consists of three stages: sparse circular reparameterization, differentiable circle-to-pixel transformation, and pixel-to-circle optimization, as shown in Figure 3.

Sparse circular reparameterization.. We transform the pixel-level mask into sparse circles for circle-based ILT optimization. Let $\mathbf{M}'$ be the pixel-level mask and A' be the point set for the mask shapes. We fracture them into circles as $A' = \cup_i C(p_i, r_i)$ using Algorithm 1, where $p_i = (x_i, y_i)$ represents a pixel. We add a learnable parameter $q_i \in \mathbb{R}$ to each circle as the circle activation. Smaller q_i lowers the probability of the circle's existence. We initialize q_i to 1 for all the circles. With this reparameterization, we can obtain the sparse circular representation of a mask consisting of a set of four-element tuples $\{(x_1, y_1, r_1, q_1), \cdots, (x_n, y_n, r_n, q_n)\}$, where n is the number of circles. Note that the sparse circular representation can recover a full mask by unioning all the circles with $q_i > 0.5$, and this mask definitely meets the circular constraints for CFAOPC since each circle serves as one shot. Therefore, our target is to optimize these $4n$ parameters to achieve the best mask performance.

Differentiable circle-to-pixel transformation.. Since the lithography simulation, as shown in Equation (1), only accepts pixel-level mask $\mathbf{M}$, we need to convert the sparse circular representation to a dense mask so that we can perform simulation and adopt gradient-based optimization to modify the mask. The key challenge is that this transformation must be differentiable to allow the gradient flow to reach the sparse circular representation. On the one hand, the coordinates x_i and y_i and the radius r_i must be integers since they are all metrics on the pixel grid, so we can not directly optimize them through a conventional approach. On the other hand, we need to design a framework to let the update on the dense mask take effect on the sparse circular representation. To address these issues, we

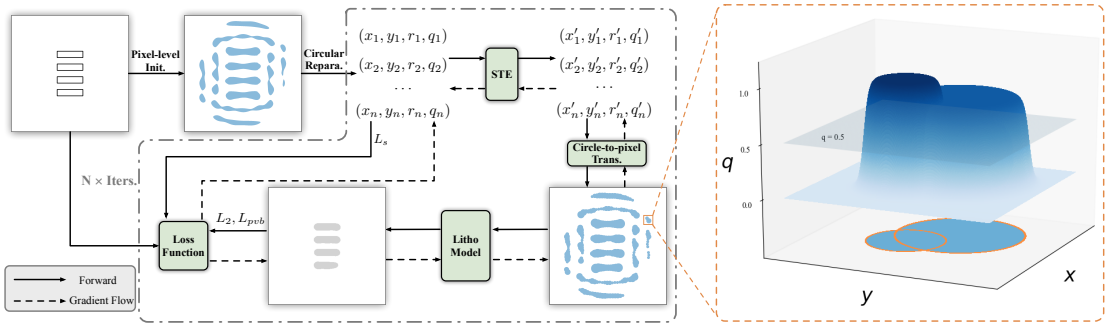
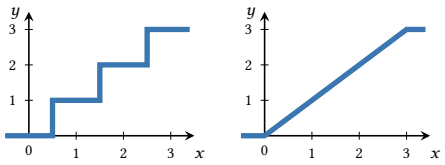


Figure 3: Overall flow of CircleOpt.



(a) **(b)**

Figure 4: Visualization of the straight-through estimator. (a) STE forward; (b) STE backward.

first let x_i, y_i, and r_i be real numbers for ease of optimization, and then we quantize them into integers through

$$x_i' = \text{STE}(x_i), y_i' = \text{STE}(y_i), r_i' = \text{STE}(r_i), \tag{7}$$

where $\text{STE}(\cdot)$ is a straight-through estimator (STE) [15] defined as

$$\text{STE}(x) = \text{Round}(\text{Clip}(x, X_{\min}, X_{\max})), \tag{8}$$

$$\frac{\partial \text{STE}(x)}{\partial x} = \mathbb{1}_{\{X_{\min} \le x \le X_{\max}\}}(x). \tag{9}$$

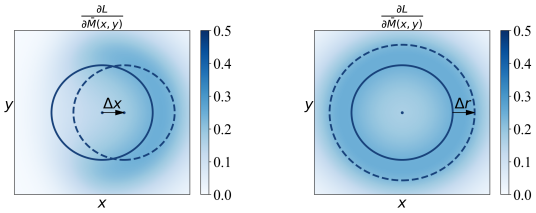
In Equation (8) and Equation (9), $X_{\min}$ and $X_{\max}$ denote the lower bound and upper bound value of x, respectively. In our implementation, we set $X_{\min}$ and $X_{\max}$ to 0 and the height/width W of the mask for x_i and y_i, respectively. For r_i, $X_{\min}$ and $X_{\max}$ are set to $R_{\min}$ and $R_{\max}$, respectively. The key insight of STE is to use the gradient of the output to estimate the gradient of the input in the backward stage, as shown in Figure 4. By adopting straight-through estimators, we can eliminate the integer constraints for the coordinates and radius.

Then, with the quantized coordinates x_i' and y_i' and the radius r_i', we propose a circular window function for each circle:

$$f_{x_i', y_i', r_i'}(x, y) = \frac{1}{1 + e^{-\alpha(-\sqrt{(x-x_i')^2+(y-y_i')^2}+r_i')}}, \tag{10}$$

where (x, y) are the variables and can be any position in a 2D dense mask, and α is a hyper-parameter for adjusting the steepness of the window function. If a given point (x, y) is within the radius of the circle $C((x_i', y_i'), r_i')$, the window activation will be close to 1. Otherwise, the window activation will be close to 0 if (x, y) is beyond the radius of the circle $C((x_i', y_i'), r_i')$. Denoting the learnable dense mask transformed from the sparse circular representation as $\bar{\mathbf{M}}$, we can define the transformation as:

$$\bar{\mathbf{M}}(x, y) = \max_{i \in \{1, \cdots, n\}} \{q_i f_{x_i', y_i', r_i'}(x, y)\}. \tag{11}$$



(a) Gradient $w.r.t\ x$ **(b) Gradient $w.r.t\ x$**

Figure 5: Schematic illustration of the updates of the circular representations from the gradient of the dense mask $\bar{\mathbf{M}}$.

The insight behind Equation (11) is, for a given point (x, y) in the dense mask $\bar{\mathbf{M}}$, the activation of this point is the maximum activation of all the circles that enclose this point. In other words, a point in $\bar{\mathbf{M}}$ will be considered part of the mask if there exists at least one circle that covers this point. A visualization of a two-circle case is shown in Figure 3. We only need to traverse all the sparse circle tuples to recover the dense mask, which is lightweight in terms of computation. More importantly, this circle-to-pixel transformation is totally differentiable, which enables us to update the sparse circle representation via gradient on $\bar{\mathbf{M}}$. Let $g(x) = x(1 - x)$ and $h_{x_i', y_i', r_i'}(x, y) = (g \circ f_{x_i', y_i', r_i'})(x, y)$. According to the chain rule, the gradient of one pixel in $\bar{\mathbf{M}}$ with respect to x_i can be calculated as:

$$\frac{\partial \bar{\mathbf{M}}(x, y)}{\partial x_i} = \begin{cases} 0, & \text{if } i \neq \underset{i \in \{1, \cdots, n\}}{\text{argmax}} \{q_i f_{x_i', y_i', r_i'}(x, y)\} \\ \dfrac{\alpha q_i h_{x_i', y_i', r_i'}(x, y)(x - x_i)\mathbb{1}_{[0, W]}(x_i)}{\sqrt{(x - x_i')^2 + (y - y_i')^2}}, & \text{o/w} \end{cases} \tag{12}$$

The gradient $w.r.t\ y_i$ can be obtained by simply replacing x_i with y_i in Equation (12). Similarly, the gradient $w.r.t\ r_i$ and q_i can be calculated as:

$$\frac{\partial \bar{\mathbf{M}}(x, y)}{\partial r_i} = \begin{cases} 0, & \text{if } i \neq \underset{i \in \{1, \cdots, n\}}{\text{argmax}} \{q_i f_{x_i', y_i', r_i'}(x, y)\} \\ \alpha q_i h_{x_i', y_i', r_i'}(x, y)\mathbb{1}_{[R_{\min}, R_{\max}]}(r_i), & \text{o/w} \end{cases} \tag{13}$$

$$\frac{\partial \bar{\mathbf{M}}(x, y)}{\partial q_i} = \begin{cases} 0, & \text{if } i \neq \underset{i \in \{1, \cdots, n\}}{\text{argmax}} \{q_i f_{x_i', y_i', r_i'}(x, y)\} \\ f_{x_i', y_i', r_i'}(x, y), & \text{o/w} \end{cases} \tag{14}$$

With a closer look at Equation (12), we can find that only the points near the boundary of the circle, i.e., $\sqrt{(x - x_i')^2 + (y - y_i')^2} \approx r_i'$, will affect the updates of the coordinates x_i' and y_i' since $h_{x_i', y_i', r_i'}(x, y)$ are almost zeros excepts for points near the circle boundary. Similarly,

Table 1: Comparison between CircleRule and SOTA pixel-based OPC methods. We show the averaged results for each metric on ICCAD 2013 benchmark.

Model	L_2	PVB	EPE	#Shot
Develset [4]	39992.9	46251.7	8.2	699.8
Develset+CircleRule	49231.1	43407.1	14.4	123.8
NeuralILT [11]	37878.9	51092.9	8.1	332.1
NeuralILT+CircleRule	41720.6	50420.7	11.9	149.9
MultiILT [10]	27171.0	39854.0	2.9	271.5
MultiILT+CircleRule	35790.0	40725.0	8.3	260.0

for r_i, the updates are also determined by the points near the boundary due to the existence of $h_{x_i', y_i', r_i'}(x, y)$ as shown in Equation (13). Meanwhile, the changes on q_i will be pushed mainly by points inside the circle, as shown in Equation (13). Figure 5 provides an example of how the gradient flow is passed to the sparse circular representations from the dense mask $\bar{\mathbf{M}}$. As shown in Figure 5(a), given a circle $C((x_i', y_i'), r_i')$, if the pixels near the right circle boundary have large gradients (these pixels are likely to be part of mask), the gradient of these pixels will be aggregated as shown in Equation (12) and push x_i' to the right. Figure 5(b) shows another example for r_i'. If the pixels outside the circle have large gradients, the gradient of the radius r_i' will be positive, thus making the circle larger.

Pixel-to-circle optimization.. With the dense mask $\bar{\mathbf{M}}$ obtained from the sparse circular representation and the differentiable circle-to-pixel transformation, we can perform pixel-to-circle optimization to update the circle parameters, as shown in Figure 3. The loss function is defined as:

$$L = L_2 + L_{pvb} + \gamma L_s, \tag{15}$$

where L_2, L_{pvb}, and L_s represent the squared L2 loss, the PVB loss and the sparsity regularizer, respectively. γ is a hyper-parameter tuning the weight of the sparsity regularizer. The L_2 loss and PVB loss are calculated from the pixel-based mask $\bar{\mathbf{M}}$, through a conventional pixel-based ILT process. The gradient flow from the squared L2 loss to x_i in the circular representation can be formulated as:

$$\frac{\partial L_2}{\partial x_i} = \sum_{(x,y)\in\mathcal{U}} \frac{\partial L_2}{\partial \bar{\mathbf{M}}(x,y)} \frac{\partial \bar{\mathbf{M}}(x,y)}{\partial x_i}. \tag{16}$$

The gradient of the squared L2/PVB loss $w.r.t$ y_i, r_i and q_i can be formulated in a similar way. In Equation (16), $\mathcal{U}$ denotes a region where the gradients are aggregated for a given circle. In our implementation, we limit this region to a square with a width marginally larger than the diameter of the circle. This has two rationales behind it. On one hand, as shown in the aforementioned analysis, pixels too far from the circle center will not significantly affect the updates of sparse circular representation. On the other hand, involving too many pixels will lead to huge computational graphs requiring much GPU memory footprint and time.

The sparsity regularizer L_s is defined as:

$$L_s = \sum_{i=1}^{n} |q_i|, \tag{17}$$

which is used to further reduce the number of circles (shots) in the final mask. As we can see, Equation (17) is in essence a Lasso regularizer on all the circle activations. Being a widely-used technique for creating sparsity in tensors, leveraging this regularizer can effectively increase the sparsity in $(q_1, \cdots, q_n)$ so as to reduce the shot count since circles with small q values will not be considered in the final mask.

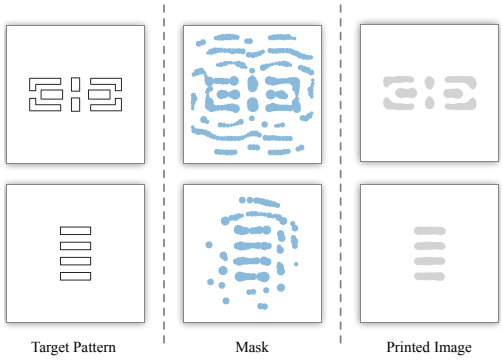


Figure 6: Visualization of masks generated by CircleOpt.

5 Experiments

Implementation Details. We implement our proposed methods on PyTorch. All the experiments are done on a single Nvidia GeForce RTX 3090 GPU. The optimization step size is set to 0.1, and γ is set to 3. The maximum and minimum circle radius are set to 76 and 12, respectively. The steepness of the circular window function α is set to 8, and the sample distance is set to 32. The methods are tested on the widely used ICCAD 2013 benchmark [12], which includes ten layout tiles from industrial designs at the 32nm technology node. The proposed algorithm is developed based on OpenILT [16], providing efficient GPU acceleration and comprehensive evaluation. As described in Section 2.3, L_2, PVB, and EPE, and *#Shot* are used to evaluate the optimized masks.

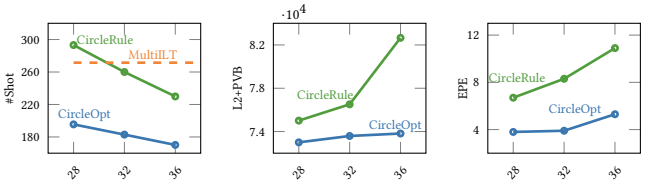
Main Results. We implement CircleRule based on three state-of-the-art methods [4, 10, 11]. On one hand, the comparison between CircleRule and SOTA OPC methods is shown in Table 1. As we can see, with circles as the lithography shots, the total shot count can be significantly reduced, validating the manufacturability advantage of the circular E-beam mask writer for curvilinear masks. Meanwhile, CircleRule suffers from significant performance loss in terms of squared L2 loss and EPE. This is due to the fact that CircleRule only tries to fit the original pixel-based masks, but the extra circular constraints will lead to imperfect fitting for arbitrarily shaped curvilinear masks. Moreover, we can observe that CircleRule obtains comparable and even better performance (for NeuralILT [11] and Develset [4]) on PVB. We conjecture that circular shapes are more robust against varying process conditions. On the other hand, the comparison between CircleRule and CircleOpt is shown in Table 2. CircleOpt outperforms CircleRule based on all the SOTA pixel-based ILT methods in terms of mask performance. Especially for EPE and squared L2 loss, CircleOpt surpasses the best CircleRule (based on MultiILT [10]) with a margin of 50% and 9%, respectively. For shot count, CircleRule based on Develset [4] achieves the minimal number since the masks do not include any SRAFs. When compared with CircleRule based on MultiILT [10] (masks with SRAFs), CircleOpt also requires 20% fewer shots while maintaining superior performance. The mask visualization of CircleOpt is shown in Figure 6.

Ablation Study. To further validate the effectiveness of our proposed methods, we conduct several ablation studies on the sample distance and the circular sparsity loss.

Table 2: Mask Printability, Complexity Comparison for CircleRule and CircleOpt.

Bench	Area(nm^2)	Develset [4]+CircleRule				NeuralILT [11]+CircleRule				MultiILT [10]+CircleRule				CircleOpt			
		L_2	PVB	EPE	#Shot	L_2	PVB	EPE	#Shot	L_2	PVB	EPE	#Shot	L_2	PVB	EPE	#Shot
case1	215344	59632	51087	23	102	53005	57775	13	170	49293	46945	8	253	43358	46905	3	214
case2	169280	53240	42993	19	80	45662	53770	13	128	39583	38565	2	236	35496	37920	1	215
case3	213504	107342	70906	69	205	93116	92536	61	213	105217	64381	69	242	75206	66241	32	194
case4	82560	21512	25623	7	44	18912	28873	6	73	14209	23649	2	164	13205	23234	1	104
case5	281958	53242	52706	4	140	52714	53575	8	192	32398	51711	0	317	34938	53110	1	172
case6	286234	52837	48595	5	203	48805	52200	7	190	38767	48290	1	313	36797	44629	0	252
case7	229149	36973	43124	0	146	24560	47353	0	146	17391	38744	0	335	21036	41118	0	175
case8	128544	18209	22917	1	65	16730	27114	2	89	12516	19968	0	181	13906	19859	0	169
case9	317581	62119	59295	8	214	53743	70986	9	221	40871	58311	1	407	47844	54625	1	251
case10	102400	27205	16825	8	39	9959	20025	0	77	9034	16694	0	152	9107	16969	0	83
Average		49231.1	43407.1	14.4	**123.8**	41720.6	50420.7	11.9	149.9	35790.0	40725.0	8.3	260.0	**33089.3**	**40451.5**	**3.9**	182.9

† L_2 and PVB unit: nm^2.



(a) (b) (c)

Figure 7: Ablation study on sample distance: (a) The shot count; (b) Performance on L_2+PVB; (c) Performance on EPE.

Table 3: Ablation study on the circular sparsity regularizer.

Method	L_2	PVB	EPE	#Shot
CircleOpt w/o Sparsity	**32595.1**	**40193.2**	4.1	208.0
CircleOpt	33089.5	40451.7	**3.9**	**182.9**

Sample distance plays a vital role in the number of shot count in CFAOPC. We test different sample distances, and the shot counts for CircleRule and CircleOpt are shown in Figure 7(a). As we can see, as the sample distance increases, the final shot count number decreases accordingly. When the sample distance is larger than 32, both CircleRule and CircleOpt gain significant improvements in terms of shot count number compared with the SOTA method, MultiILT [10]. Meanwhile, we can also observe that the shot count number of CircleOpt is less sensitive to sample distance compared with that of CircleRule, benefitted from the optimization-based nature of CircleOpt. Moreover, we also test the mask quality with different sample distances, as shown in Figure 7(b) and Figure 7(c). The larger the sample distance is, the worse the mask quality will be since we will have less room to adjust the mask shape under the circular-shaped constraints. Moreover, similar to the shot count number, CircleOpt is also more robust to the variation of sample distance in terms of mask quality compared with CircleRule, which further validates the superiority of CircleOpt.

We also verify the effectiveness of the sparsity regularizer, as shown in Table 3. With the sparsity regularizer L_s, CircleOpt can obtain 12% improvements for the shot count number while only suffering from very minor performance loss on L_2 and PVB and even obtaining some improvements on EPE. This illustrates the effectiveness of our proposed sparsity regularizer coupled with CircleOpt.

6 Conclusion

Conventional manufacturing for curvilinear masks requires fracturing them into a set of rectangles, resulting in a large shot count number and exceedingly high mask manufacturing costs. A promising alternative is the circular e-beam mask writer [7], which produces the main feature as circles thus significantly reducing the shot count for curvilinear masks. To fully leverage its advantages, we explore a novel research problem, circular fracturing-aware OPC (CFAOPC), which aims to generate masks that are optimally fractured into overlapping circles with variable radii for a given target pattern. To address this problem, we propose two methods. The first rule-based method, CircleRule, is built on SOTA pixel-level ILT methods and fractures their output into circles according to some predefined rules. The second one, CircleOpt, is an optimization-based method that integrates the circular constraints into the standard ILT flow and generates the circle-based masks directly by gradient-based optimization. Extensive experimental results validate the effectiveness of both methods and show that CircleOpt outperforms CircleRule.

References

[1] J. Kuang, W.-K. Chow, and E. F. Y. Young, "A robust approach for process variation aware mask optimization," in *Proc. DATE*, 2015.

[2] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *Proc. DAC*, 2014.

[3] B. Jiang, L. Liu, Y. Ma, B. Yu, and E. F. Y. Young, "Neural-ILT 2.0: Migrating ilt to domain-specific and multitask-enabled neural network," *IEEE TCAD*, 2022.

[4] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "DevelSet: Deep neural level set for instant mask optimization," in *Proc. ICCAD*, 2021.

[5] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Y. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE TCAD*, 2020.

[6] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *Proc. ICCAD*, 2020.

[7] A. Fujimura, D. Kim, T. Komagata, Y. Nakagawa, V. Tolani, and T. Cecil, "Best depth of focus on 22-nm logic wafers with less shot count," in *Photomask and Next-Generation Lithography Mask Technology XVII*, K. Hosono, Ed., vol. 7748, 2010.

[8] L. Pang, "Inverse lithography technology: 30 years from concept to practical, full-chip reality," *Journal of Micro/Nanopatterning, Materials, and Metrology*, 2021.

[9] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A gpu-enabled level set method for mask optimization," in *Proc. DATE*, 2021.

[10] S. Sun, F. Yang, B. Yu, L. Shang, and X. Zeng, "Efficient ilt via multi-level lithography simulation," in *Proc. DAC*, 2023.

[11] B. Jiang, L. Liu, Y. Ma, B. Yu, and E. F. Young, "Neural-ILT 2.0: Migrating ilt to domain-specific and multitask-enabled neural network," *IEEE TCAD*, 2021.

[12] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *Proc. ICCAD*, 2013.

[13] H. H. Hopkins, "The concept of partial coherence in optics," in *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 208, no. 1093, 1951.

[14] Q. Wang, B. Jiang, M. D. F. Wong, and E. F. Y. Young, "A2-ILT: GPU accelerated ilt with spatial attention mechanism," in *Proc. DAC*, 2022.

[15] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[16] S. Zheng, Y. Ma, B. Zhu, G. Chen, W. Zhao, S. Yin, Z. Yu, and B. Yu, "Openilt: An open-source platform for inverse lithography technique research," https://github.com/OpenOPC/OpenILT/, 2023.