# Curvilinear Optical Proximity Correction via Cardinal Spline

Su Zheng[1], Xiaoxiao Liang[2], Ziyang Yu[1], Yuzhe Ma[2], Bei Yu[1], Martin Wong[3]
[1]Chinese University of Hong Kong, [2]Hong Kong University of Science and Technology (GZ), [3]Hong Kong Baptist University

***Abstract*—This paper presents a novel curvilinear optical proximity correction (OPC) framework. The proposed approach involves representing mask patterns with control points, which are interconnected through cardinal splines. Mask optimization is achieved by iteratively adjusting these control points, guided by lithography simulation. To ensure compliance with mask rule checking (MRC) criteria, we develop comprehensive methods for checking width, space, area, and curvature. Additionally, to match the performance of inverse lithography techniques (ILT), we design algorithms to fit ILT results and resolve MRC violations. Extensive experiments demonstrate the effectiveness of our methodology, highlighting its potential as a viable OPC/ILT alternative.**

## I. Introduction

Integrated circuit (IC) manufacturing is a highly complex process that transforms raw materials into tiny chips. It begins with the design of the circuit layout, which is then transferred onto a photomask. Lithography plays a central role in printing the mask patterns onto the silicon wafer. As IC designs become more complex and feature sizes continue to shrink, the demand for well-optimized masks increases significantly.

Mask optimization [1]–[12] is a crucial process in semiconductor manufacturing, involving techniques like rule-based optical proximity correction (OPC), model-based OPC, sub-resolution assist features (SRAF), and inverse lithography technology (ILT). Rule-based OPC [13]–[15] uses predefined rules to correct distortions. Model-based OPC [16]–[18] adopts lithography simulation to guide the correction. SRAFs [19] are small, sub-wavelength features added to the mask that impact the behavior of light passing through them. ILT [20]–[30] treats mask optimization as an inverse problem and iteratively modifies the mask to get desired wafer patterns.

Although ILT is recognized for its high pattern fidelity, model-based OPC offers several advantages in terms of efficiency, reliability, and mask manufacturability. OPC is more computationally efficient and converges better, as the optimization problems of ILT are more complex and resource-intensive. Additionally, OPC primarily relies on geometric operations, whereas ILT typically requires intensive image-based processes. This allows researchers to integrate mask rule checking (MRC) into model-based OPC flows, resulting in mask patterns with improved manufacturability. While multi-beam mask writers (MBMW) can support curvilinear mask patterns, ILT-generated masks often suffer from poor manufacturability due to complex non-Manhattan geometries. Therefore, a methodology that combines the efficiency, reliability, and mask manufacturability of OPC with the high pattern fidelity of ILT would be highly beneficial.

Curvilinear OPC [31], [32] aims to bridge the gap between OPC and ILT by allowing curvilinear mask shapes in OPC. It maintains the ability to handle curvilinear shapes and any-angle edges while having lower computational complexity than ILT. Furthermore, it can produce more manufacturable masks, as the mask patterns can be optimized to better comply with mask rules. This approach strikes a balance between the simplicity of OPC and the precision of ILT, providing a more practical solution for mask optimization.

In this paper, we propose a novel curvilinear OPC framework, **CardOPC**. By representing mask patterns with control points connected through cardinal splines, our approach allows for flexible and precise mask representation. This method leverages the inherent smoothness and adaptability of splines to accurately capture intricate geometries. As illustrated by Fig. 1, the control point representation in curvilinear OPC can inherit the advantage of fewer variables from traditional OPC while gaining the flexibility of curvilinear shapes from ILT. Compared to the existing methods using Bézier splines [31], [32], cardinal splines offer significant advantages, particularly in terms of computational efficiency, without sacrificing the strengths of Bézier splines like local control and smoothness. This reduces the computational overhead and simplifies the optimization process.

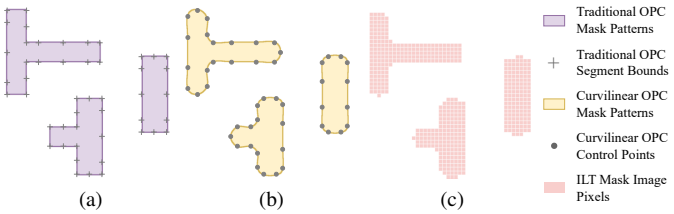



Fig. 1 Illustration of the mask pattern representation in (a) traditional OPC; (b) curvilinear OPC; (c) ILT. Curvilinear OPC combines the reduced number of variables from traditional OPC and the flexibility of curvilinear shapes from ILT.

The iterative adjustment of control points, guided by lithography simulation, achieves high-quality mask optimization. This process ensures that the wafer patterns closely match the target design, minimizing deviations that could affect the quality of products. The utilization of control points and splines can maintain a small number of variables while being able to optimize complex shapes, offering superior efficiency and adaptability in mask optimization.

For the manufacturability of masks, we propose comprehensive methods for checking MRC violations, covering width, space, area, and curvature rules. We demonstrate that the analytical spline-based representation simplifies mask rule checking and makes curvilinear OPC superior to ILT in terms of MRC. Additionally, our framework incorporates techniques to fit ILT results and resolve MRC violations, positioning it as a promising alternative to traditional ILT methods. By integrating ILT fitting capabilities, our approach can leverage the strengths of ILT while mitigating its limitations. This hybrid approach allows for more robust and effective mask optimization, combining the best aspects of both OPC and ILT.

The major contribution of this paper can be summarized as follows:

- We propose a curvilinear OPC framework based on cardinal splines. It combines the computational efficiency of traditional OPC and the flexibility of ILT.
- We derive the methods for curvilinear mask rule checking, which are the foundation of curvilinear OPC's advantages in addressing MRC violations.
- By integrating ILT fitting capabilities, the framework leverages the strengths of ILT while mitigating its limitations, offering a more robust and effective mask optimization solution.
- Extensive experiments reveal the superiority of the proposed framework over existing methods, showcasing higher precision, better scalability, and greater efficiency.

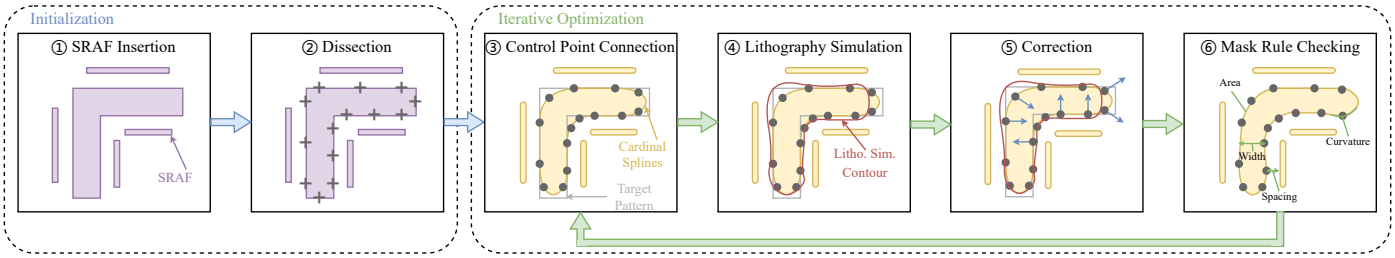



Fig. 2 Overview of our CardOPC. In the initialization phase, we ① insert SRAFs and ② dissect the polygons as traditional OPC does. The segments are converted to the control points of cardinal splines. During the optimization phase, we ③ connect the control points with cardinal splines to form the mask patterns, which are subsequently input to the ④ lithography simulation engine. According to the lithography simulation contours, we ⑤ estimate the edge displacement and correct the distortions by moving the control points. After that, ⑥ mask rule checking and MRC violation resolving are performed to ensure the mask manufacturability.

## II. Preliminaries

### A. Lithography Simulation

Lithography simulation is a crucial step in mask optimization, essential for obtaining the desired printed patterns on the wafer. The process starts with the input mask $\boldsymbol{M}$, which is projected through an optical system to form the aerial image $\boldsymbol{I}$. For example, the Hopkins diffraction model [33] can be employed to efficiently approximate the projection behavior. In this model, the aerial image $\boldsymbol{I}$ is obtained by applying a set of optical kernels $\boldsymbol{H}$ to the mask $\boldsymbol{M}$ with:

$$\boldsymbol{I}(x,y) = \sum_{k=1}^{N_h} w_k \left| \boldsymbol{M}(x,y) \otimes \boldsymbol{h}_k(x,y) \right|^2 . \tag{1}$$

$N_h$ is the number of optical kernels, $\boldsymbol{h}_k$ is the $k$-th optical kernel in $\boldsymbol{H}$, and $w_k$ is the corresponding weight. $\boldsymbol{I}$ can be binarized using the intensity threshold $I_{th}$ that indicates the exposure level. After the binarization, the contour $\boldsymbol{C}$ of the lithography simulation result can be extracted and utilized to guide the OPC process.

### B. Optical Proximity Correction

When light passes through a photomask and projects onto a wafer, various optical phenomena such as diffraction and interference can cause deviations from the intended pattern. These deviations, known as optical proximity effects, can lead to issues such as line width variations and pattern displacement. These effects become more significant as feature sizes shrink, making it challenging to achieve the desired pattern fidelity. OPC addresses this issue by modifying the mask patterns to compensate for distortions brought by lithography. Model-based OPC [17] typically involves the following steps:

1) Dissection (or fragmentation): The edges of the input polygons are dissected into smaller segments, usually with more segments around corners to allow for finer control.
2) SRAF insertion: SRAFs are inserted based on rules regarding the width and distance of the assist features from the main features, in order to improve the printability and process window.
3) Iterative optimization: The algorithm iteratively retrieves lithography simulation results and adjusts the segments to minimize edge placement error (EPE) violations.

EPE is a metric that evaluates the difference between the lithography simulation contour and the target pattern. To calculate EPE, a series of points are sampled along the edges of the target design, and the distance between each point's position in the target pattern and its position in the lithography simulation contour is measured. If the EPE of a point exceeds a specified threshold, it is counted as an EPE violation. L2 is another way to estimate the error of the printed patterns. It is the sum of the squared errors between the lithography simulation result and the target pattern. Furthermore, under varying lithography conditions, the printed patterns can be different. Therefore, process variation band (PVB) is also widely adopted as a metric of OPC, which evaluates the maximum discrepancy between the printed patterns under varying process conditions.

### C. Mask Rule Checking

Mask Rule Checking (MRC) is a common practice in the semiconductor industry to ensure the manufacturability of photomask data before mask writing. Traditional MRC checks the rectilinear mask patterns for compliance with mask rules, such as minimum feature sizes, minimum distances, corner-to-corner distance, etc. As the industry moves towards curvilinear mask shapes to improve printed patterns, the traditional MRC approach needs to be adapted.

Reference [34] introduces the following curvilinear mask rules:

1) Spacing and width constraints: These rules ensure minimum distances between patterns and minimum widths of the shapes.
2) Area constraint: It accounts for the resolution limits, as the patterns may fail to print reliably if they are too tiny.
3) Curvature constraint: This rule limits the range of the curvilinear patterns' curvatures to ensure reliability.

Combining these constraints provides a comprehensive set of rules to ensure the manufacturability of curvilinear mask patterns.

## III. CardOPC Methodology

### A. Overview

Fig. 2 outlines the proposed curvilinear OPC flow. These segments are then transformed into the control points. During optimization, these control points are connected using cardinal splines to generate mask patterns, which are subsequently fed into lithography simulation. Based on the lithography simulation contours, edge displacements are estimated, and distortions are corrected by adjusting the control points. Finally, mask rule checking and MRC violation resolving are performed to ensure the manufacturability of the mask.

### B. Initialization

Fig. 3 summarizes the initialization phase of CardOPC. Fig. 3(a) shows an example of simple rule-based SRAF insertion. It creates an SRAF with a length of $l_s = r \times l_m$, where $l_m$ is the length of the main pattern edge and $r$ is a constant that controls the ratio between the SRAF and main pattern. The SRAF is placed at a distance of $d_{ms}$ from the main pattern. Note that SRAF insertion can also be completed by calling external tools (e.g. Calibre) or fitting ILT results (see Section III-G). Fig. 3(b) demonstrates our basic dissection

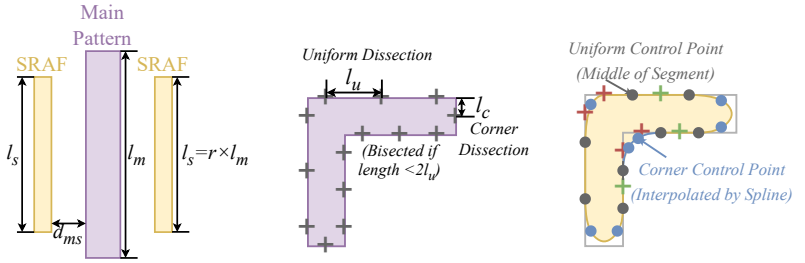



Fig. 3 Illustration of the initialization phase of our curvilinear OPC flow. (a) An example of simple rule-based SRAF insertion. The SRAF is placed at a certain distance from the main pattern, with a shorter length than the main pattern. (b) Basic dissection method of our OPC flow. It creates shorter segments around corners, and longer segments for the remaining parts of the edges. (c) Generation of control points. Most control points are chosen as the middle points of the segments. However, around corners, it interpolates the segment boundary points with cardinal splines to generate the control points.

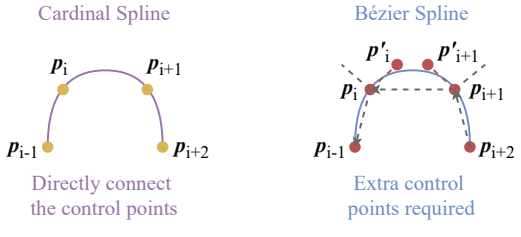



Fig. 4 Comparison between cardinal splines and Bézier splines in curvilinear OPC. Using cardinal splines, the curve can pass through all the control points $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$, and $\boldsymbol{p}_{i+2}$. However, when using Bézier splines, two extra control points $\boldsymbol{p}'_i$ and $\boldsymbol{p}'_{i+1}$ should be generated to ensure that the curve can pass through $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$. This introduces non-negligible computational overhead.

method. Specifically, it creates shorter segments with a length of $l_c$ around corners, where finer control is needed, and longer segments with a length of $l_u$ for the remaining parts of the edges. Fig. 3(c) illustrates the generation of control points. Most control points are chosen as the middle points of the segments obtained from the dissection step. However, around corners, it interpolates neighboring segment boundary points using cardinal splines to generate the control points for the corner regions. Additionally, we also convert the SRAFs to cardinal splines to achieve a uniform pattern representation.

### C. Cardinal Splines for Control Point Connection

Cardinal spline is a type of spline that is used in mathematics and computer graphics to create smooth and flexible shapes. This type of interpolating spline can pass through a given set of control points. This feature makes cardinal splines particularly useful for creating smooth curves that follow a specified path.

In this paper, we adopt cardinal splines for the following reasons:

1) The curve passes through each control point, ensuring that the shape of the curve is influenced directly by these points.
2) Cardinal splines include a tension parameter $s$ that controls the tightness of the curve. This allows users to finetune the curvilinear shapes without moving the control points.
3) The tangents, normals, and curvatures of cardinal splines can be analytically computed, ensuring efficient edge displacement checking and mask rule verification in curvilinear OPC.

To connect two points $\boldsymbol{p}_i = (x_i, y_i)$ and $\boldsymbol{p}_{i+1} = (x_{i+1}, y_{i+1})$ with cardinal spline, we need their predecessor $\boldsymbol{p}_{i-1}$ and successor $\boldsymbol{p}_{i+2}$. Any point between $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ can be given by:

$$\boldsymbol{p}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -s & 0 & s & 0 \\ 2s & s-3 & 3-2s & -s \\ -s & 2-s & s-2 & s \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{i-1} \\ \boldsymbol{p}_i \\ \boldsymbol{p}_{i+1} \\ \boldsymbol{p}_{i+2} \end{bmatrix},$$
$$t \in [0,1],\ \boldsymbol{p}(0) = \boldsymbol{p}_i,\ \boldsymbol{p}(1) = \boldsymbol{p}_{i+1}. \tag{2}$$

In practice, we connect each pair of control points $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ by sampling multiple points in $t \in [0,1]$ evenly. For simplicity, we can represent equation (2) using $\boldsymbol{p}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \boldsymbol{S}_{card} \boldsymbol{P}_{i-1:i+3}$. The computation can be accelerated by GPU using PyTorch.

Fig. 4 compares cardinal splines and Bézier splines in curvilinear OPC. With cardinal splines, the curve can seamlessly traverse all the control points, namely $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$, and $\boldsymbol{p}_{i+2}$. Conversely, Bézier splines, necessitate the generation of two additional control points, $\boldsymbol{p}'_i$ and $\boldsymbol{p}'_{i+1}$, to guarantee the passage through $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ [34]. This requirement incurs a non-trivial computational overhead.

### D. Lithography Simulation

Our experiments involve two lithography simulation tools. One of them is from the ICCAD-13 contest [35], which is implemented using equation (1) with GPU acceleration [36]. The other one is Calibre, with the same settings as the previous works [8], [9].

### E. Correction

As introduced by [37], the primary goal of OPC is to minimize the edge placement error (EPE) by modifying mask edges. Given $E$ EPE checking points, we can represent the EPEs with a function $\boldsymbol{f}_{EPE}(\boldsymbol{M})$, which contains the distances between checking points' positions in the target pattern and their positions in the lithography simulation contour. It can be decomposed using Taylor expansion:

$$\boldsymbol{f}_{EPE}(\boldsymbol{M}_{\tau+1}) = \boldsymbol{f}_{EPE}(\boldsymbol{M}_\tau) + \boldsymbol{f}'_{EPE}(\boldsymbol{M}_\tau) \times \boldsymbol{\Delta M}_\tau + \epsilon(\boldsymbol{M}_\tau), \tag{3}$$

where $\tau$ is the current iteration, $\boldsymbol{M}_\tau$ is the current mask, $\boldsymbol{\Delta M}_\tau$ is the modification of the mask at the current iteration, and $\epsilon(\boldsymbol{M}_\tau)$ represents the high order terms. To guarantee a manageable complexity, the high-order terms can be ignored, and equation (3) becomes:

$$\boldsymbol{f}_{EPE}(\boldsymbol{M}_{\tau+1}) \approx \boldsymbol{f}_{EPE}(\boldsymbol{M}_\tau) + \boldsymbol{f}'_{EPE}(\boldsymbol{M}_\tau) \times \boldsymbol{\Delta}\text{Positions}_\tau. \tag{4}$$

$\boldsymbol{\Delta M}_\tau$ in equation (3) becomes $\boldsymbol{\Delta}\text{Positions}_\tau$ in equation (4), which represents the moving distance of the segments or control points. The matrix $\boldsymbol{f}'_{EPE}(\boldsymbol{M}_\tau)$ reflects how movements affects the EPE. Given $D$ segments or control points, $\boldsymbol{f}'_{EPE}(\boldsymbol{M}_\tau)$ can be defined as:

$$\boldsymbol{f}'_{EPE}(\boldsymbol{M}_\tau) = \begin{bmatrix} \frac{\partial e_1}{\partial d_1} & \frac{\partial e_1}{\partial d_2} & \cdots & \frac{\partial e_1}{\partial d_D} \\ \frac{\partial e_2}{\partial d_1} & \frac{\partial e_2}{\partial d_2} & \cdots & \frac{\partial e_2}{\partial d_D} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_E}{\partial d_1} & \frac{\partial e_E}{\partial d_2} & \cdots & \frac{\partial e_E}{\partial d_D} \end{bmatrix}, \tag{5}$$

where $\frac{\partial e_i}{\partial d_j}$ is the partial derivative of the $i$-th EPE with respect to the moving distance of the $j$-th segment or control point. Since our target

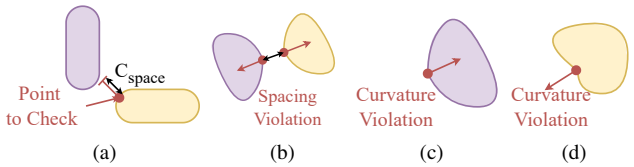



Fig. 5 Illustration of mask rule checking and MRC violation resolving. (a) shows how to create a line segment to test the spacing rule violation. (b) moves the control points to resolve spacing rule violations. (c) and (d) resolve curvature rule violations by moving the control points in and out, respectively.

is to make $\boldsymbol{f}_{EPE}(\boldsymbol{M}_{\tau+1}) = 0$, the moving distance of segments or control points at each iteration can be given by:

$$\boldsymbol{\Delta}\text{Positions}_{\tau} = -\boldsymbol{f}'_{EPE}(\boldsymbol{M}_{\tau})^{-1} \times \boldsymbol{f}_{EPE}(\boldsymbol{M}_{\tau}). \tag{6}$$

In a basic OPC solver, we can check EPE for each segment or control point, which means that $E = D$. Moreover, we can assume that $\frac{\partial e_i}{\partial d_i}$ is a constant and $\frac{\partial e_i}{\partial d_j} = 0$ if $i \neq j$. Under these settings, $\boldsymbol{f}'_{EPE}(\boldsymbol{M}_{\tau})^{-1}$ becomes a diagonal matrix $\boldsymbol{f}'_{EPE}(\boldsymbol{M}_{\tau})^{-1} = \gamma \boldsymbol{I}$, where $\gamma$ is the moving distance of each segment or control point. To improve the moving smoothness and mimic multi-segment solvers, we can take a weighted average on the moving distances/directions of neighboring points. For instance, given moving distance $\Delta d_i$ from equation (6) and $2W + 1$ neighboring points $\{d_{i-W}, d_{i-W+1}, ..., d_{i+W}\}$ from the same shape, the final moving distance $\overline{\Delta d_i}$ is:

$$\overline{\Delta d_i} = \sum_{k=-W}^{W} w_k \Delta d_k. \tag{7}$$

In curvilinear OPC, to guarantee the support of any-angle edges, we need to calculate both the moving distance $\Delta d_i$ and the moving direction $\boldsymbol{n}_i$ for each control point. We use the normal vectors $\boldsymbol{n}(t)$ of the cardinal splines as the moving directions, which can be derived from the derivatives, $\boldsymbol{g}(t) = \boldsymbol{p}'(t)$, with the following equations:

$$\boldsymbol{g}(t) = \begin{bmatrix} 0 & 1 & 2t & 3t^2 \end{bmatrix} \boldsymbol{S}_{card} \boldsymbol{P}_{i-1:i+3}. \tag{8a}$$

$$\overline{\boldsymbol{g}}(t) = (\overline{g}_x(t), \overline{g}_y(t)) = \frac{\boldsymbol{g}(t)}{\|\boldsymbol{g}(t)\|}, \tag{8b}$$

$$\boldsymbol{n}(t) = (n_x(t), n_y(t)) = (-\overline{g}_y(t), \overline{g}_x(t)). \tag{8c}$$

The moving direction $\boldsymbol{n}_i$ can be given by the normal vector at the corresponding $t$, and the moving step is $\Delta d_i \boldsymbol{n}_i$. In practice, we can take the weighted average among $\Delta d_i \boldsymbol{n}_i$ rather than $\Delta d_i$ to allow the control points to move in more directions, providing a higher degree of freedom. Note that GPU acceleration is also applicable for computing moving directions. Additionally, our curvilinear OPC framework is compatible with other gradient-based OPC algorithms like the MEEF-based OPC [38] and differentiable OPC [12] methods.

### *F. Mask Rule Checking and MRC Violation Resolving*

**Mask Rule Checking**. We focus on the following mask rules:

1) Spacing rule: The spacing between two shapes should be larger than the spacing constraint $C_{space}$.
2) Width rule: The width of each shape should not be smaller than the width constraint $C_{width}$.
3) Area rule: The area of each shape should exceed the area constraint $C_{area}$.
4) Curvature rule: The curvature at each point should be less than the curvature constraint $C_{curv}$.

**Algorithm 1** Fitting Method for ILT-Optimized Masks

**Input:** ILT-optimized mask image $\boldsymbol{M}$, cardinal spline function $\boldsymbol{F}(\cdot)$, ratio $r_Q$ and $r_R$, learning rate $\alpha$.

```
//Initialization;
Q = ∅;   //Control point set;
R = ∅;   //Reference point set;
for each shape S_i in the mask image M do
    Extract the boundary points of S_i, denoted by P_i;
    Sample r_Q|P_i| points from P_i evenly, add them to Q;
    Sample r_R|P_i| points from P_i evenly, add them to R;
end for
//Optimization;
for k ∈ {1, 2, ....., K} do //K iterations in total;
    Interpolate Q with F(·) to have |R| points;
    Compute the loss L(Q, R) = ‖F(Q) − R‖²;
    Optimize Q with Q ← Q − α ∂L(Q,R)/∂Q;
end for
return Q;
```

Among these rules, we provide basic solutions for space, width, and area checking, which are common topics in existing works like mask rule checking [39] and design rule checking [40]. We leave the well-optimized checking of these rules for future works. As for the curvature rule, we derive the analytical method for its checking, which is efficient because it can be accelerated by GPU.

For spacing and width checking, we construct an R-tree [41] using all shapes of the mask patterns. R-tree is a type of data structure used for indexing multi-dimensional information such as geographical coordinates, rectangles, or polygons. It stores spatial objects in a hierarchical manner using bounding rectangles. Each node in the tree represents a bounding rectangle that contains its child nodes. This structure allows for efficient querying of spatial data, such as querying the nearest shape and finding all objects that intersect a given area.

As demonstrated by Fig. 5(a), to check whether a point has a spacing violation, we build a line segment with a length of $C_{space}$ that starts from the checking point, with its direction specified by its normal vector. If the line segment touches a shape, it has a spacing violation. It can be done efficiently using R-tree's querying algorithm. We implement width checking in a similar way, but create the line segment in the directions opposite to the point's normal vector.

The area of each shape can be calculated by the shoelace formula, which enables the checking of the area constraint. Both R-tree construction and area computation are provided by Shapely.

The curvature at a point can be defined as:

$$\boldsymbol{\kappa}(t) = \begin{vmatrix} p'_x(t) & p''_x(t) \\ p'_y(t) & p''_y(t) \end{vmatrix} / \|\boldsymbol{p}'(t)\|^3, \tag{9}$$

where $\boldsymbol{p}'(t) = (p'_x(t), p'_y(t))$, $\boldsymbol{p}''(t) = (p''_x(t), p''_y(t))$, $\boldsymbol{p}'(t)$ is calculated by equation (8), and $\boldsymbol{p}''(t)$ is given by:

$$\boldsymbol{p}''(t) = \begin{bmatrix} 0 & 0 & 2 & 6t \end{bmatrix} \boldsymbol{S}_{card} \boldsymbol{P}_{i-1:i+3}. \tag{10}$$

If one point has $\boldsymbol{\kappa}(t) > C_{curv}$, it has a curvature violation. This checking is used to ensure that the mask patterns are smooth enough and friendly for mask manufacturing.

**MRC Violation Resolving**. When MRC violation occurs, we can address it according to its type. As shown in Fig. 5(b), when a spacing violation occurs between two points, we try to move the corresponding control points in the directions opposite to their normal vectors, enlarging the spacing between the points. In the trials, the

TABLE I Via-layer OPC result comparison on EPE ($nm$) and PVB ($nm^2$)

| Testcase | #Vias | DAMO [29] | | Calibre | | RL-OPC [8] | | CAMO [9] | | CardOPC | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | EPE | PVB | EPE | PVB | EPE | PVB | EPE | PVB | EPE | PVB |
| V1 | 2 | 7 | 5822 | 8 | 5837 | 6 | 5730 | 1 | 5797 | 2 | 5775 |
| V2 | 2 | 8 | 5836 | 8 | 5834 | 7 | 5813 | 5 | 5734 | 4 | 5772 |
| V3 | 3 | 14 | 8565 | 11 | 8587 | 13 | 8594 | 10 | 8470 | 5 | 8426 |
| V4 | 3 | 14 | 8621 | 12 | 8771 | 14 | 8679 | 10 | 8576 | 7 | 8550 |
| V5 | 4 | 18 | 10615 | 15 | 10775 | 16 | 10772 | 10 | 10503 | 5 | 10501 |
| V6 | 4 | 20 | 10739 | 15 | 10763 | 19 | 10659 | 15 | 10507 | 8 | 10499 |
| V7 | 5 | 28 | 12993 | 23 | 12615 | 23 | 12485 | 23 | 12097 | 7 | 12381 |
| V8 | 5 | 26 | 13047 | 19 | 12784 | 24 | 12547 | 19 | 12437 | 16 | 12270 |
| V9 | 6 | 30 | 15497 | 24 | 15454 | 26 | 15414 | 19 | 15186 | 15 | 15112 |
| V10 | 6 | 35 | 15088 | 27 | 15064 | 33 | 14588 | 26 | 14556 | 16 | 14287 |
| V11 | 6 | 39 | 15516 | 27 | 15782 | 31 | 15538 | 21 | 15333 | 9 | 15290 |
| V12 | 6 | 36 | 15424 | 23 | 15686 | 24 | 15464 | 23 | 15204 | 18 | 15073 |
| V13 | 6 | 32 | 16970 | 23 | 17035 | 39 | 17440 | 14 | 16712 | 6 | 16833 |
| Average | 4.5 | 23.6 | 11902.5 | 18.1 | 11922.1 | 21.2 | 11824.8 | 15.1 | 11624.0 | **9.1** | **11597.6** |
| Ratio | - | 156.3% | 102.4% | 119.9% | 102.6% | 140.4% | 101.7% | 100.0% | 100.0% | **60.3%** | **99.8%** |

TABLE II Metal-layer OPC result comparison on EPE ($nm$) and PVB ($nm^2$)

| Testcase | #Points | Calibre | | RL-OPC [8] | | CAMO [9] | | CardOPC | |
|---|---|---|---|---|---|---|---|---|---|
| | | EPE | PVB | EPE | PVB | EPE | PVB | EPE | PVB |
| M1 | 64 | 49 | 28728 | 104 | 29390 | 44 | 27795 | 37 | 26387 |
| M2 | 84 | 61 | 37386 | 117 | 39139 | 67 | 36467 | 49 | 34737 |
| M3 | 88 | 81 | 39430 | 137 | 41623 | 59 | 39451 | 37 | 36491 |
| M4 | 100 | 89 | 45741 | 252 | 46892 | 60 | 44961 | 44 | 40528 |
| M5 | 106 | 66 | 47220 | 336 | 47041 | 69 | 46582 | 36 | 45283 |
| M6 | 112 | 102 | 49887 | 355 | 51433 | 78 | 49438 | 23 | 47567 |
| M7 | 116 | 89 | 52584 | 325 | 50770 | 83 | 49961 | 57 | 48794 |
| M8 | 24 | 20 | 11014 | 32 | 10770 | 23 | 10928 | 1 | 10353 |
| M9 | 72 | 50 | 22531 | 197 | 22360 | 42 | 22032 | 15 | 21139 |
| M10 | 120 | 91 | 37546 | 263 | 36368 | 95 | 36849 | 11 | 37727 |
| Average | 88.6 | 69.8 | 37206.7 | 211.8 | 37578.6 | 62.0 | 36446.4 | **31.0** | **34900.6** |
| Ratio | - | 112.6% | 102.1% | 341.6% | 103.1% | 100.0% | 100.0% | **50%** | **95.8%** |

moving distance is chosen from small to large. The violation can usually be resolved in a few trials. For width violations, we use a similar method but move the control points in the directions specified by their normal vectors.

If an area rule violation occurs after moving the control points, we can cancel the move to avoid the violation. If it occurs after fitting ILT results, we remove the shape since it is usually a small and non-printable pattern.

When a curvature violation occurs at a point, we try to move the corresponding control point in and out according to the normal vector. Fig. 5(c) and Fig. 5(d) visualize this strategy. The violation can usually be resolved within a few attempts.

### G. ILT-OPC Hybrid Approach

Continuous transmission mask (CTM) [12], [19], [42] is utilized to generate SRAFs for model-based OPC. Since CTM relies on ILT, it can serve as a hybrid method that combines ILT and OPC. However, existing methods can only generate rectilinear SRAFs according to the continuous mask intensity distribution from ILT, which can not fully utilize the ILT's ability. In this paper, we propose to fit ILT results with cardinal splines, which can mimic the complex curvilinear shapes in ILT-optimized masks, and resolve MRC violations using the strategies described in Section III-F. Our hybrid approach allows for more robust and effective mask optimization, combining the best aspects of both OPC and ILT.

As presented by Algorithm 1, our fitting method consists of the following steps:

1) Initialization: For each shape $\boldsymbol{S}_i$ in the mask image $\boldsymbol{M}$, we find the contour of $\boldsymbol{S}_i$ using the border following algorithm [43] (line 5), which is already implemented in the OpenCV toolbox. After that, we uniformly sample control points $\boldsymbol{Q}$ and reference points $\boldsymbol{R}$ from each shape's contour (lines 6-7). During optimization, we move the control points using gradient descent to fit the reference points.
2) Optimization: At each optimization step, we interpolate $\boldsymbol{Q}$ using the cardinal spline function $\boldsymbol{F}(\cdot)$ given by equation (2) (line 11). With the interpolated results, we can compute the mean squared error loss (line 12) between the interpolated points $\boldsymbol{F}(\boldsymbol{Q})$ and the reference points $\boldsymbol{R}$. Finally, we move $\boldsymbol{Q}$ using gradient descent (line 13) to minimize the distance between $\boldsymbol{F}(\boldsymbol{Q})$ and $\boldsymbol{R}$.

Note that the fitting algorithm can be implemented using PyTorch so that this process can be accelerated by GPU. Optimizers like Adam [44] can be adopted for the gradient descent in our method to accelerate the convergence.

TABLE III Large-scale OPC result comparison on EPE violations and PVB ($\mu m^2$)

| Testcase | #Tiles ($30 \times 30 \mu m^2$) | Calibre | | SimpleOPC [45] | | CardOPC | |
|---|---|---|---|---|---|---|---|
| | | EPE | PVB | EPE | PVB | EPE | PVB |
| gcd | 1 | 3657 | 35.6651 | 3454 | 37.3002 | 3507 | 34.2606 |
| aes | 144 | 2722 | 27.7226 | 2571 | 29.4301 | 2578 | 27.3485 |
| dynamicnode | 144 | 2088 | 26.1663 | 1941 | 27.1213 | 1923 | 25.5011 |
| Average | - | 2409 | 26.9746 | 2260 | 28.3069 | **2255** | **26.4519** |
| Ratio | - | 100.0% | 100.0% | 93.8% | 104.9% | **93.6%** | **98.1%** |

After fitting the ILT-optimized masks, we obtain the control points $\boldsymbol{Q}$ of all mask patterns. The MRC violation resolving algorithms discussed in Section III-F are employed to address the MRC violations in the curvilinear shapes.

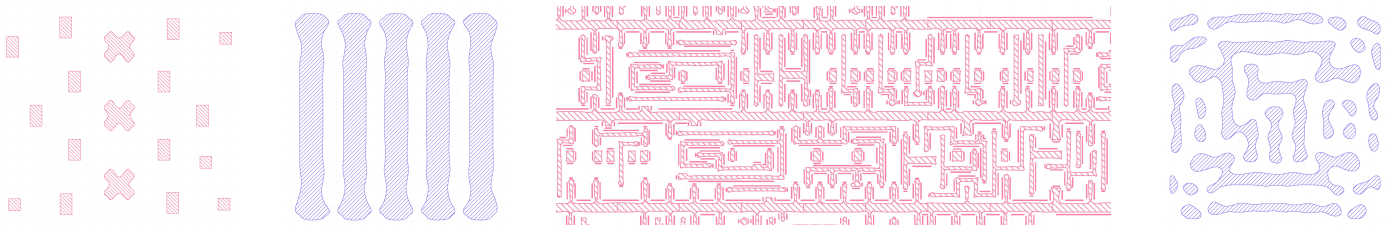
## IV. Experiments

### A. Comparison with Existing Works

Previous works focus on the OPC for metal-layer and via-layer patterns [8], [9]. We adopt the same testcases as them. For the experiments on via-layer patterns, we use 13 test layout clips [46], each of which is $2 \times 2 \mu m^2$. SRAFs are inserted by Calibre before the OPC algorithm launches. For metal-layer patterns, the dataset has 10 clips, each of which is $1.5 \times 1.5 \mu m^2$. The EPE measure points are selected following conventional strategies. For via-layer patterns, the center of each edge corresponds to a measuring point. For metal-layer patterns, the EPE measure points are evenly placed on the edges along the primary direction with $60nm$ spacing.

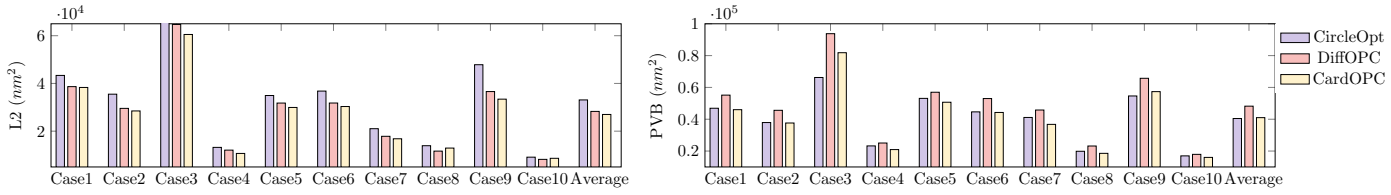
For the via layer, we dissect the polygons with $l_c = 20nm$ and $l_u = 30nm$. The moving distance is $2nm$. For the metal layer, we dissect the polygons with $l_c = 30nm$ and $l_u = 60nm$. The moving distance is $4nm$. We use 32 optimization iterations for both layers, and decay the moving distance by a factor of 0.5 at the 16-th step. A tension parameter $s = 0.6$ is adopted for the cardinal splines.

In our experiments on the via layer, we compare our CardOPC framework with existing OPC methods such as DAMO [29], RL-OPC [8], and CAMO [9], as well as the commercial tool Calibre. For the metal-layer experiments, we compare CardOPC with RL-OPC, CAMO, and Calibre. As shown in TABLE I, our framework outperforms existing methods on the via layer, achieving a 39.7% improvement in EPE and a 0.2% enhancement in PVB. Similarly, TABLE II demonstrates that our framework surpasses existing methods on the metal layer, with a 50.0% improvement in EPE and a 4.2% enhancement in PVB. Compared to these state-of-the-art (SOTA)



(a) (b) (c) (d)

Fig. 6 Examples of our curvilinear OPC results: (a) via-layer OPC; (b) metal-layer OPC; (c) large-scale OPC; (d) ILT-OPC hybrid method.




(a) (b)

Fig. 7 Comparison between our ILT-OPC hybrid approach and SOTA methods. Our method demonstrates superior performance on the L2 metric and delivers competitive PVB results. Additionally, our approach achieves an average of 1.4 EPE violations, outperforming CircleOpt (3.9) and DiffOPC (2.2).

methods, the proposed curvilinear OPC framework offers significant advantages in EPE, along with improvements in PVB. Additionally, Fig. 6(a) and Fig. 6(b) present examples of via-layer and metal-layer OPC results, respectively, which can show that our method is able to produce smooth curvilinear main patterns.

### *B. Large-Scale OPC Results*

We conduct experiments to verify that our CardOPC can be employed in large-scale OPC applications. Following [45], we use the `gcd`, `aes`, and `dynamicnode` layouts generated by OpenROAD [47] with NanGate 45nm PDK [48]. Each layout is cropped into $30 \times 30 \mu m^2$ tiles before we perform OPC for its metal layer. In CardOPC, we dissect the polygons with $l_c = 40nm$ and $l_u = 40nm$. The moving distance is set to $8nm$. We use 10 optimization iterations, and decay the moving distance by a factor of 0.5 at the 8-th step. A tension parameter $s = 0.6$ is adopted for the cardinal splines.

We compare our method with Calibre, which uses the same configuration in [8], [9], and runs for 20 iterations. The PVB and EPE metrics are computed by Calibre. As shown in TABLE III, CardOPC surpasses Calibre, with a 6.4% improvement in EPE and a 1.9% enhancement in PVB. Our method also outperforms the previous work [45]. These results indicate that CardOPC is promising for large-scale applications, offering advantages in EPE and PVB. As an example, Fig. 6(c) shows a local region of `gcd`'s OPC result.

### *C. ILT-OPC Hybrid Approach Results*

Following SOTA methods like [7], [12], [49], we use the lithography source and testcases from ICCAD-13 contest [35] in this experiment. We build our ILT flow based on the OpenILT framework [36] and fit its results using our ILT-OPC hybrid approach described in Section III-G. The proposed method is compared with CircleOpt [49] and DiffOPC [12], each of which has mechanisms to avoid MRC violations like CardOPC. As shown in Fig. 7, our approach excels in the L2 metric and offers competitive PVB results. In term of EPE, it achieves an average of 1.4 EPE violations, surpassing CircleOpt (3.9) and DiffOPC (2.2). Additionally, our MRC resolving strategies successfully reduce the average number of violations from 43.8 to 0. These results can validate the effectiveness of our framework. Fig. 6(d) shows an example of our results, which is similar to the corresponding ILT result but without MRC violations.

### *D. Ablation Study*

In this experiment, we compare Bézier [31] and cardinal splines in curvilinear OPC. The primary difference in runtime between Bézier and cardinal splines lies in the control point connection process. For the `gcd` large-scale testcase, the Bézier method takes 3.6 seconds to connect the control points of 1,776 shapes at the first iteration, while our cardinal method only needs 1.9 seconds. This results in an 89% runtime overhead for the Bézier method compared to our approach. The reason for this overhead is that the Bézier method necessitates the creation of additional points when connecting two adjacent control points, which involves extra operations such as vector rotation. In terms of performance, the Bézier method obtains an EPE of 3,532 and a PVB of 34.9088, while our cardinal method achieves an EPE of 3,507 and a PVB of 34.2606. By combining runtime and performance results, we demonstrate the superiority of our curvilinear OPC framework based on cardinal splines.

## V. Conclusion

In this paper, we develop a novel curvilinear OPC framework, CardOPC. After representing the mask patterns with control points, cardinal splines are employed to connect the control points. Guided by lithography simulation, the efficient correction method enables our CardOPC to achieve a 50.0% improvement in EPE and a 4.2% enhancement in PVB when compared to the SOTA OPC method. We further demonstrate the scalability of CardOPC by testing it on large-scale layout tiles. Moreover, by fitting ILT results and resolving MRC violations, our ILT-OPC hybrid method outperforms other tested algorithms. Finally, the comparison between different splines reveals that cardinal splines can offer a faster speed and better performance than Bézier splines. CardOPC shows promise as a viable ILT/OPC alternative and opens various avenues for future curvilinear OPC research, including spline types, MRC methods, MRC violation resolution strategies, and large-scale optimization.

## ACKNOWLEDGEMENTS

The project is supported in part by Research Grants Council of Hong Kong SAR (No. RFS2425-4S02, No. CUHK14211824 and No. CUHK14210723), and the MIND project (MINDXZ202404).

## REFERENCES

[1] J. Ou, B. Yu, J.-R. Gao, D. Z. Pan, M. Preil, and A. Latypov, "Directed self-assembly based cut mask optimization for unidirectional design," in *Proc. GLSVLSI*, 2015, pp. 83–86.

[2] Y. Ma, X. Zeng, and B. Yu, "Methodologies for layout decomposition and mask optimization: A systematic review," in *Proc. VLSI-SoC*, 2017.

[3] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," in *Proc. ICCAD*, 2017, pp. 81–88.

[4] W. Zhong, S. Hu, Y. Ma, H. Yang, X. Ma, and B. Yu, "Deep learning-driven simultaneous layout decomposition and mask optimization," in *Proc. DAC*, 2020.

[5] H. Yang, W. Zhong, Y. Ma, H. Geng, R. Chen, W. Chen, and B. Yu, "VLSI mask optimization: From shallow to deep learning," *Integration, the VLSI Journal*, vol. 77, pp. 96–103, 2021.

[6] W. Zhao, X. Yao, Z. Yu, G. Chen, Y. Ma, B. Yu, and M. D. Wong, "AdaOPC: A self-adaptive mask optimization framework for real design patterns," in *Proc. ICCAD*, 2022.

[7] S. Zheng, H. Yang, B. Zhu, B. Yu, and M. Wong, "LithoBench: Benchmarking ai computational lithography for semiconductor manufacturing," in *Proc. NeurIPS*, 2023.

[8] X. Liang, Y. Ouyang, H. Yang, B. Yu, and Y. Ma, "RL-OPC: Mask optimization with deep reinforcement learning," *IEEE TCAD*, 2023.

[9] X. Liang, H. Yang, K. Liu, B. Yu, and Y. Ma, "CAMO: Correlation-aware mask optimization with modulated reinforcement learning," *Proc. DAC*, 2024.

[10] S. Zheng, Y. Ma, B. Yu, and M. Wong, "EMOGen: Enhancing mask optimization via pattern generation," in *Proc. DAC*, 2024.

[11] W. Zhao, X. Yao, S. Yin, Y. Bai, Z. Yu, Y. Ma, B. Yu, and M. D. Wong, "AdaOPC 2.0: Enhanced adaptive mask optimization framework for via layers," *IEEE TCAD*, 2024.

[12] G. Chen, H. Yang, H. Ren, B. Yu, and D. Z. Pan, "Differentiable edge-based OPC," in *Proc. ICCAD*, 2024.

[13] J.-S. Park, C.-H. Park, S.-U. Rhie, Y.-H. Kim, M.-H. Yoo, J.-T. Kong, H.-W. Kim, and S.-I. Yoo, "An efficient rule-based OPC approach using a DRC tool for 0.18/spl mu/m ASIC," in *Proc. ISQED*, 2000, pp. 81–85.

[14] S. O'Brien, R. Soper, S. Best, and M. Mason, "Rules based process window OPC," in *Design for Manufacturability through Design-Process Integration II*, vol. 6925, 2008, pp. 396–404.

[15] J. Wang, A. Wei, P. Verma, and W. Wilkinson, "Optimization of rule-based OPC fragmentation to improve wafer image rippling," in *European Mask and Lithography Conference*, vol. 9661, 2015, pp. 79–94.

[16] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process variation and pattern fidelity aware mask optimization algorithm," in *Proc. ICCAD*, 2014, pp. 238–245.

[17] J. Kuang, W.-K. Chow, and E. F. Young, "A robust approach for process variation aware mask optimization," in *Proc. DATE*, 2015, pp. 1591–1594.

[18] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE TCAD*, vol. 35, no. 8, pp. 1345–1357, 2016.

[19] Z. Yu, P. Liao, Y. Ma, B. Yu, and M. D. Wong, "CTM-SRAF: Continuous transmission mask-based constraint-aware sub resolution assist feature generation," *IEEE TCAD*, 2023.

[20] Y. Liu, D. Abrams, L. Pang, and A. Moore, "Inverse lithography technology principles in practice: Unintuitive patterns," in *BACUS Symposium on Photomask Technology*, vol. 5992, 2005, pp. 886–893.

[21] 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.

[22] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A GPU-enabled level set method for mask optimization," in *Proc. DATE*, 2021, pp. 1835–1838.

[23] ——, "A GPU-enabled level-set method for mask optimization," *IEEE TCAD*, vol. 42, no. 2, pp. 594–605, 2022.

[24] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. DAC*, 2018.

[25] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE TCAD*, vol. 39, no. 10, pp. 2822–2834, 2019.

[26] B. Jiang, L. Liu, Y. Ma, H. Zhang, B. Yu, and E. F. Young, "Neural-ILT: Migrating ILT to neural networks for mask printability and complexity co-optimization," in *Proc. ICCAD*, 2020.

[27] 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*, vol. 41, no. 8, pp. 2671–2684, 2021.

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

[29] G. Chen, W. Chen, Q. Sun, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full-chip scale," *IEEE TCAD*, vol. 41, no. 9, pp. 3118–3131, 2022.

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

[31] Y. Zhang, L. Wu, J. Rao, and Y. Wang, "Curvilinear OPC mask synthesis flow," in *SPIE Photomask Technology*, vol. 12751, 2023.

[32] Y.-Y. Chen, K.-H. Chang, W.-L. Cheng, and Y.-P. Tang, "Curvilinear mask handling in opc flow," *JM3*, vol. 23, no. 1, pp. 011 203–011 203, 2024.

[33] 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. The Royal Society London, 1951, pp. 263–277.

[34] I. Bork, A. Tritchkov, S. Shang, E. Levine, and M. Zuo, "MRC for curvilinear mask shapes," in *SPIE Photomask Technology*, vol. 11518, 2020, pp. 110–117.

[35] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *Proc. ICCAD*, 2013, pp. 271–274.

[36] S. Zheng, B. Yu, and M. Wong, "OpenILT: An open source inverse lithography technique framework," in *Proc. ASICON*, 2023.

[37] N. B. Cobb and Y. Granik, "Model-based OPC using the MEEF matrix," in *BACUS Symposium on Photomask Technology*, vol. 4889, 2002, pp. 1281–1292.

[38] J. Lei, Y. Yang, G. Lippincott, and X. Zhang, "Model-based OPC using the MEEF matrix III," in *DTCO and Computational Patterning III*, vol. 12954, 2024, pp. 343–351.

[39] M. C. Keck, W. Ziegler, R. Liebe, T. Franke, G. Ballhorn, M. Koefferlein, and J. Thiele, "Mask manufacturing rule check: how to save money in your mask shop," in *BACUS Symposium on Photomask Technology*, vol. 4186, 2001, pp. 114–118.

[40] Z. He, Y. Zuo, J. Jiang, H. Zheng, Y. Ma, and B. Yu, "OpenDRC: An efficient open-source design rule checking engine with hierarchical gpu acceleration," in *Proc. DAC*, 2023.

[41] S. T. Leutenegger, M. A. Lopez, and J. Edgington, "STR: A simple and efficient algorithm for R-tree packing," in *Proc. ICDE*, 1997, pp. 497–506.

[42] J.-C. Yu, P. Yu, and H.-Y. Chao, "Model-based sub-resolution assist features using an inverse lithography method," in *Lithography Asia*, vol. 7140, 2008, pp. 254–264.

[43] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[44] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] S. Zheng, G. Xiao, G. Yan, M. Dong, Y. Li, H. Chen, Y. Ma, B. Yu, and M. Wong, "Model-based OPC extension in OpenILT," in *Proc. ISEDA*, 2024, pp. 568–573.

[46] K. Liu, H. Yang, Y. Ma, B. Tan, B. Yu, E. F. Young, R. Karri, and S. Garg, "Adversarial perturbation attacks on ML-based CAD: A case study on CNN-based lithographic hotspot detection," *ACM TODAES*, vol. 25, no. 5, pp. 1–31, 2020.

[47] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proc. DAC*, 2019.

[48] "Nangate 45nm library," http://www.nangate.com/.

[49] X. Zhang, S. Zheng, G. Chen, B. Zhu, H. Xu, and B. Yu, "Fracturing-aware curvilinear ILT via circular E-beam mask writer," in *Proc. DAC*, 2024.