

DevelSet: Deep Neural Level Set for Instant Mask Optimization

Guojin Chen, Ziyang Yu, Hongduo Liu, Yuzhe Ma, Bei Yu
The Chinese University of Hong Kong

Abstract—With the feature size continuously shrinking in advanced technology nodes, mask optimization is increasingly crucial in the conventional design flow, accompanied by an explosive growth in prohibitive computational overhead in optical proximity correction (OPC) methods. Recently, inverse lithography technique (ILT) has drawn significant attention and is becoming prevalent in emerging OPC solutions. However, ILT methods are either time-consuming or in weak performance of mask printability and manufacturability. In this paper, we present DevelSet, a GPU and deep neural network (DNN) accelerated level set OPC framework for metal layer. We first improve the conventional level set-based ILT algorithm by introducing the curvature term to reduce mask complexity and applying GPU acceleration to overcome computational bottlenecks. To further enhance printability and fast iterative convergence, we propose a novel deep neural network delicately designed with level set intrinsic principles to facilitate the joint optimization of DNN and GPU accelerated level set optimizer. Experimental results show that DevelSet framework surpasses the state-of-the-art methods in printability and boost the runtime performance achieving instant level (around 1 second).

I. INTRODUCTION

As minimum feature size continues to shrink, the optical diffraction and proximity effects in lithography become not negligible, which could seriously degrade the yield of integrated circuits. To compensate for pattern distortion and improve process window in the lithography process, optical proximity correction (OPC) is used to ensure pattern transfer fidelity. Typical OPC approaches encompass rule-based methods [1], model-based methods [2], [3], inverse lithography techniques [4], [5], and DNN-based methods [6]–[8].

In model-based OPC procedure, the edges of the initial mask are fragmented into segments, which are moved iteratively under the guidance of lithography simulation. Inverse lithography techniques (ILT) also leverage rigorous simulation to perform mask printability enhancement. Moreover, ILT can achieve pixel-level optimization and thus find a better solution in a larger solution space by seeing mask optimization as an inverse imaging problem. Gao *et al.* [4] derived a closed-form gradients descent algorithm through direct edge placement error and process window optimization. In recent years, DNN-based methods have drawn great attention as they can attain significant speedup while preserve comparable mask printability by incorporating previous experience. Yang *et al.* [6] proposed a generative model to produce an initial mask solution, which greatly lowers the number of iterations required in traditional ILT methods.

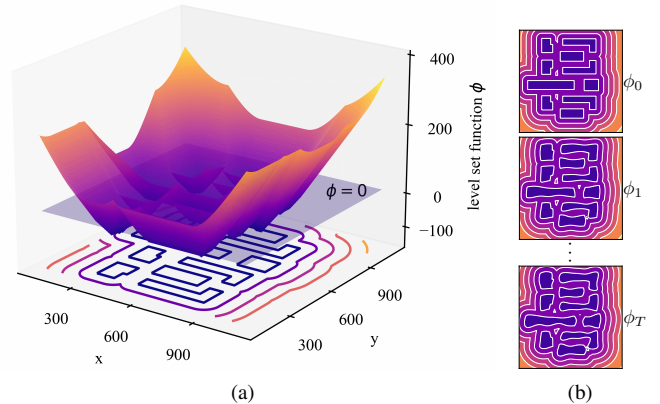


Fig. 1 (a) The 3D illustration of level set function ϕ . The mask is shaped as the cross-section of the level set continuum with the zero plane. The contours on the x-y plane are the projected level set. (b) The level set evolution process.

In the past decades, level set-based ILT methods have been actively explored as a feasible alternative to pixel-based ILT methods in OPC tasks. As illustrated in Fig. 1, the implicit representation of level set method is naturally more effective in dealing with complex topology changes and lithography development [9]. Shen *et al.* [10] solved the inverse lithography problem using a level set time-dependent model with finite difference schemes. Yu *et al.* [11] proposed a momentum-based conjugate gradient (CG) method and accelerated the level set evolution with GPU-enabled Fast Fourier Transform (FFT) algorithm.

Briefly, there are two main approaches for the inverse lithography techniques: parametric and implicit. The parametric methods [4]–[8] use pixel-wise tensors to generate the mask (Fig. 2(a)). While the implicit approaches represent the mask as a zero level set cross-section [10]–[14] (Fig. 2(b)). So far, due to the simplicity and flexibility of pixel-based gradient descent methods, the parametric methods have been thoroughly researched through the perspectives of objective function, optimization method, and the DNN acceleration, achieving state-of-the-art (SOTA) runtime performance and mask print fidelity. However, as depicted in Fig. 2(c), the parametric methods inevitably generate unnecessary isolated stains or edge glitches with zigzagging and tortuous complex mask boundaries while the level set implicit representation is accomplished in mask boundary continuity and curvature control (Fig. 2(d)). Unfortunately, due to the extra computational overhead introduced by the level set evolution, the application of level set-based ILT method has been greatly

underestimated.

With the rapid development of GPU and deep learning, the progressive potential of level set-based ILT methods should be reconsidered. Motivated by these issues, we present the DevelSet framework, which contains two parts. The GPU accelerated level set optimizer (DevelSet-Optimizer) and the deep level set neural network (DevelSet-Net). Following the improvements proposed by the previous work, DevelSet-Optimizer (DSO) incorporates the curvature term into level set-based ILT to reduce mask complexity and develops a set of GPU friendly algorithm to overcome the computational overhead. DevelSet-Net (DSN) is designed to provide better initial solutions by leveraging the fast inference ability of neural network and compensate DSO for the curvature cost by applying a novel modulation branch. The DevelSet framework benefits from end-to-end joint optimization of DSN and DSO, achieving SOTA fast convergence and mask printability. Our main contributions are:

- We propose DevelSet, an improved level set-based ILT framework with CUDA and DNN acceleration.
- We firstly introduce curvature term into level set-based ILT methods to reduce mask complexity and leverage GPU to perform all the calculations.
- We are the first to integrate level set into deep neural network for an end-to-end joint mask optimization flow.
- We design a novel multi-branch neural network architecture with level set embeddings to further boost the performance and improve mask printability.
- Experimental results show that DevelSet achieves SOTA mask printability with predominant runtime advantage for instant mask optimization *i.e.*, around 1 second.

The rest of the paper is organized as follows: Section II lists some preliminaries about level set algorithms and mask optimization methods. Section III details the DevelSet algorithm. Section IV presents our experimental results, followed by a conclusion in Section V.

II. PRELIMINARIES

In this section, we will introduce some concepts and background related to this work. Following the traditions, we denote \mathbf{Z}_t , \mathbf{M} , and \mathbf{I} , as the target layout, mask image, and intensity of aerial image respectively. \mathbf{Z} , \mathbf{Z}_{in} , and \mathbf{Z}_{out} are the wafer image under the nominal/min/max process conditions. We use \mathbf{H} for lithography kernels and ϕ for the level set function (LSF).

A. Level Set-Based ILT Algorithms

Level set as a mathematical technique is pioneered by Osher *et al.* [15]. Recently, it has been actively explored as a feasible alternative to tackle the ILT problem. Let $\mathcal{C} : \Omega \rightarrow \mathbb{R}^2$ denote a parametric curve in 2D space Ω , the level set method implicitly represents the boundary of the mask using zero crossing of a LSF $\phi(x, y) : \Omega \rightarrow \mathbb{R}$:

$$\mathcal{C} = \{(x, y) \mid \phi(x, y) = 0\}. \quad (1)$$

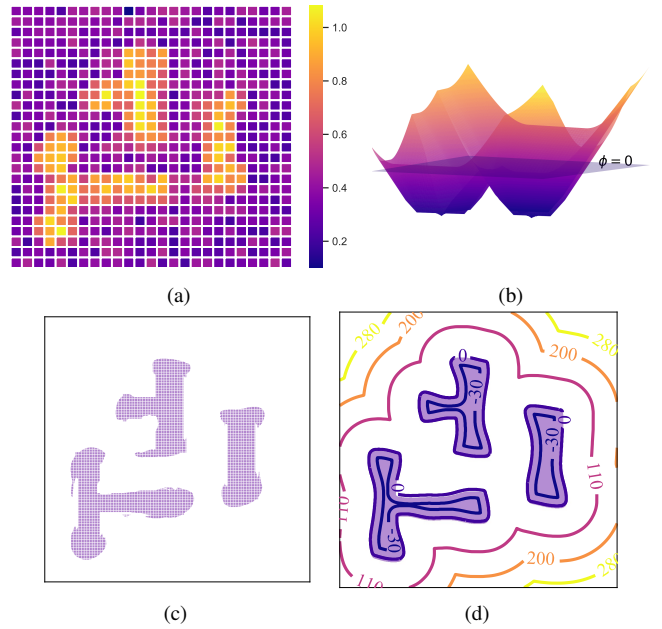


Fig. 2 Comparison of pixel-based ILT and level set-based ILT. (a) Intensity matrix of pixel-based ILT; (b) Level set-based ILT; (c) Mask generated by pixel-wise intensity threshold; (d) Mask generated by zero level set.

As depicted in Fig. 1(b), the mask optimization process can be viewed as the evolution along the descent of the LSF ϕ . The commonly used LSF ϕ is the signed distance function (SDF),

$$\phi_{\text{SDF}}(x, y) = \begin{cases} -d(x, y), & \text{if } (x, y) \in \text{inside}(\mathcal{C}), \\ 0, & \text{if } (x, y) \in \mathcal{C}, \\ d(x, y), & \text{if } (x, y) \in \text{outside}(\mathcal{C}), \end{cases} \quad (2)$$

where $d(x, y)$ is the minimum Euclidean distance from point (x, y) to the parametric curve \mathcal{C} . As illustrated in Fig. 2(d), the contours are labeled with its SDF values, and the \mathcal{C} is the contour labeled by 0. Now the mask image \mathbf{M} can be represented by ϕ as

$$\mathbf{M}(x, y) = \begin{cases} 1, & \text{if } \phi(x, y) \leq 0 \\ 0, & \text{if } \phi(x, y) > 0. \end{cases} \quad (3)$$

During the evolution, the mask boundary $\mathcal{C}(t)$ changes over time $t \in \mathbb{R}$, the curve evolution then can be formally defined as

$$\frac{\partial \mathcal{C}(t)}{\partial t} = v\mathbf{n}, \quad (4)$$

where $\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|}$ is the unit vector in the outward normal direction of the curve \mathcal{C} and v indicates the velocity along the normal direction. We use the zero level set to implicitly represent the mask boundary, thus: $\phi(\mathcal{C}(t), t) = 0$. The chain rule gives us,

$$\frac{\partial \phi(\mathcal{C}(t), t)}{\partial t} = 0 \rightarrow \frac{\partial \phi}{\partial \mathcal{C}(t)} \frac{\partial \mathcal{C}(t)}{\partial t} + \frac{\partial \phi}{\partial t} = 0. \quad (5)$$

Consider all the points on the evolving front $\mathcal{C}(t)$, $\frac{\partial \phi}{\partial \mathbf{c}} = \nabla \phi$, combining the Equation (4) and Equation (5), the motion equation of LSF $\frac{\partial \phi}{\partial t}$ can be formally expressed by

$$\frac{\partial \phi}{\partial t} = -v|\nabla \phi|. \quad (6)$$

Equation (6) is a partial differential equation (PDE), once the level set ϕ and velocity v are defined, the first-order derivative in space and time of Equation (6) can be approximated using finite difference techniques. Evolution of LSF $\phi(x, y, t)$ can be performed iteratively. We use $\phi_i(x, y)$ to denote $\phi(x, y, t_i)$ for simplicity. For $i \in \{0, 1, 2 \dots T - 1\}$, the i^{th} -step update is

$$\phi_{i+1}(x, y) = \phi_i(x, y) + \Delta t \frac{\partial \phi_i}{\partial t}, \quad (7)$$

where Δt is the time step, $\phi_0(x, y)$ is the initial LSF, and the $\phi_T(x, y)$ is the corresponding output LSF after T evolution steps. As shown in Fig. 1(b), we can obtain the optimized mask after T steps by applying the Equation (3).

B. The Lithography Simulation Model

During the conventional lithography process, the input mask \mathbf{M} is transformed through an optical projection system into the aerial image. The distribution of aerial light intensity \mathbf{I} floating on the wafer forms the printed image \mathbf{Z} . The optical projection system can be expressed mathematically using Hopkins's diffraction model [16]. The sum of coherent systems (SOCS) can roughly estimate Hopkins's diffraction model by performing singular value decomposition, the optical projection process is then replaced by a set of coherent kernels. The intensity of aerial image \mathbf{I} can be represented by convolving the mask \mathbf{M} and a set of optical kernels \mathbf{H} ,

$$\mathbf{I}(x, y) = \sum_{i=1}^{N^2} \sigma_i |\mathbf{M}(x, y) \otimes h_i(x, y)|^2. \quad (8)$$

Here \otimes denotes the convolution operation, and h_i is the i^{th} kernel of the optical kernel set \mathbf{H} and σ_i is the corresponding weight of the coherent system. The N_k^{th} order approximation to the partially coherent system can be obtained by,

$$\mathbf{I}(x, y) \approx \sum_{i=1}^{N_k} \sigma_i |\mathbf{M}(x, y) \otimes h_i(x, y)|^2, \quad (9)$$

where $N_k = 24$ in our implementation. After optical simulation, the aerial image undergoes a resist model to estimate the final printed shape on wafer. For methodology verification and also for simplicity, we adopt the constant threshold resist model which is consistent with the ICCAD 2013 contest settings [17]. As depicted in Fig. 2(c), given the print threshold I_{th} , the printed wafer image can be expressed as:

$$\mathbf{Z} = \begin{cases} 1, & \text{if } \mathbf{I} \geq I_{th}, \\ 0, & \text{if } \mathbf{I} < I_{th}. \end{cases} \quad (10)$$

C. Mask Printability and Mask Manufacturability

Mask printability represents the quality of the printed patterns generated from the optimized mask. In this paper, we use squared L_2 error and process variation band (PVBand) as two typical metrics to evaluate mask printability. Moreover, the mask fracturing shot count proposed in Neural-ILT [8] is also applied in this work to evaluate mask complexity and manufacturability.

1) *Squared L_2 error*: Given the wafer image \mathbf{Z} and target image \mathbf{Z}_t , the squared L_2 error is calculated by: $\|\mathbf{Z} - \mathbf{Z}_t\|_2^2$.

2) *PVBand*: Process variation band (PVBand) is the bitwise-XOR region among all the printed patterns under different process conditions. In our work, for simplicity, we calculate the PVBand under two extreme conditions, one at nominal condition with +2% dose and the other one at defocus and -2% dose. A mask is more robust if its PVBand area is smaller.

3) *Mask Fracturing Shot Count*: Many conventional pixel-based ILT methods tend to optimize the mask only to improve mask printability. However, most of these optimized masks contain plenty of tiny irregular sub-features, which increase the difficulty for mask manufacture. In this work, we use shot count to evaluate the mask manufacturability. An evaluated mask \mathbf{M} can be fractured into a set of small rectangles which could replicate exactly the original mask. Mask fracturing shot count stands for the number of fractured rectangles.

III. DEEP NEURAL LEVEL SET ALGORITHMS

As depicted in Fig. 3, the proposed DevelSet framework consists of two parts, DevelSet-Optimizer (DSO) and DevelSet-Net (DSN). In this section, we will first introduce the improved level set-based ILT algorithm of DSO in Section III-A applying curvature term to improve the mask manufacturability, along with the full implementation in CUDA platform by combining the mechanism of GPU parallelism with the numerical setting of level set. Then a novel multi-branch neural network *i.e.*, DSN is proposed in Section III-B to: 1) provide a better initial LSF for DSO to reduce the total iterations, 2) predict a weighted matrix to selectively regularize the mask boundary and compensate for mask printability loss caused by the curvature term. Finally, we perform the end-to-end joint optimization for DevelSet in Section III-C to accomplish instant mask optimization with higher mask printability and lower mask complexity.

A. DevelSet-Optimizer (DSO)

The objective of the conventional ILT-based OPC method is to find an optimized mask $\mathbf{M}_{\text{opt}} = \mathcal{L}^{-1}(\mathbf{Z}_t; \mathbf{P}_{\text{nom}})$, where \mathbf{Z}_t is the design target and the $\mathcal{L}(\cdot; \mathbf{P}_{\text{nom}})$ denotes the forward lithography process under the nominal condition. The pixel-based ILT methods represent the intensity wafer image as the pixel-wise parameters and then update the mask pixel-by-pixel with the guidance of the inverse gradient from the lithography model. While the level set-based ILT methods regard the optimization process as the evolution of the level set continuum, the mask is formulated by the cross-section of zero

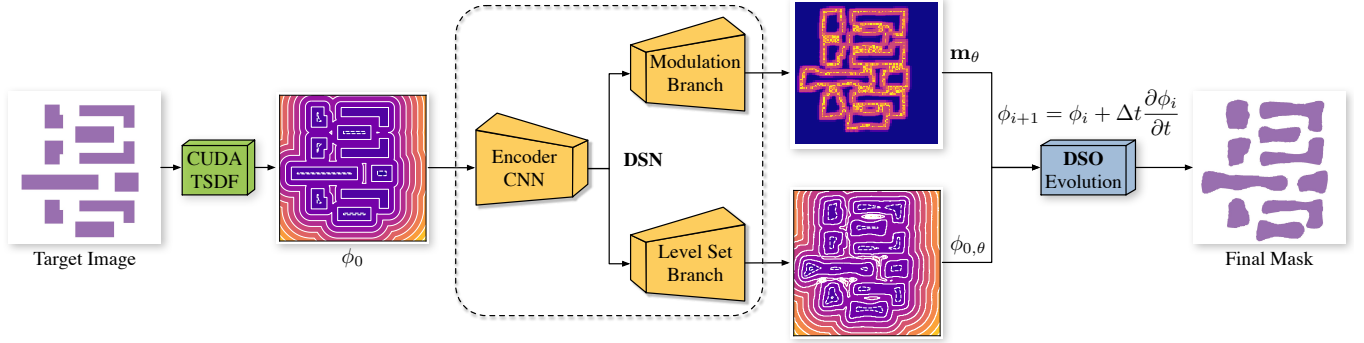


Fig. 3 Overview of DevelSet framework with the end-to-end joint optimization flow of DSN and DSO.

height plane and the level set continuum. Mathematically, the level set continuum is represented with LSF ϕ , the evolution procedure can be expressed by Equation (7). The SOTA pixel-based method Neural-ILT [8] brings the ILT to the on-neural-network training solution with a CUDA accelerated lithography simulator, achieving a breakthrough in runtime boosting. It applies a mask complexity refinement layer and domain knowledge re-training to eliminate the complex shape in masks thus reducing mask complexity. Nonetheless, Neural-ILT sacrifices mask printability that the sum of L_2 and PVBand is worse than the previous learning-based work PGAN-OPC [6]. The SOTA level set-based method GLS-ILT [11] leveraged GPU accelerated FFT to speed up the momentum-based algorithm. However, there is still room for acceleration since GLS-ILT only accelerates the FFT module and much runtime is wasted on CPU and GPU transfers.

DSO is a CUDA accelerated iterative mask optimizer applying our improved level set algorithm. By controlling the curvature term and transferring all the computations to the GPU, DSO obtains mask printability superiority and breakthrough in runtime performance simultaneously. As revealed in Equation (7), the key to leverage level set methods in mask optimization tasks is the definition of the LSF ϕ , and then integrate the ILT-based gradient as the velocity term v . Moreover, we introduce the curvature term to improve mask printability and reduce mask complexity.

1) *The Improved Level Set-Based ILT: Truncated signed distance function.* The theoretical framework of the level set method is independent of which particular LSF ϕ is used. A popular type of LSF is the signed distance function (SDF) in Equation (2). SDF is Lipschitz continuous and from Rademacher's theorem, it is almost everywhere differentiable. However, instead of using SDF, we adopt truncated signed distance function (TSDF) as our LSF with an upper bound D_u and a lower bound D_l ,

$$\phi_{\text{TSDF}} = \begin{cases} D_u, & \text{if } \phi_{\text{SDF}} > D_u, \\ \phi_{\text{SDF}}, & \text{if } D_l \leq \phi_{\text{SDF}} \leq D_u, \\ D_l, & \text{if } \phi_{\text{SDF}} < D_l. \end{cases} \quad (11)$$

In our work, we set D_u to 900 and D_l to -100 according to the design rules of the benchmark. The TSDF improves the

stability of optimization procedure by reducing the variance of the dataset. More importantly, the TSDF ensures the fast convergence of the DevelSet-Net to enable the end-to-end joint optimization of the whole mask optimization framework.

Motion term. Motion term $\frac{\partial \phi}{\partial t}$ is the core component in level set methods determining the evolution process. According to Equation (6), it is dominated by the velocity term v . In our work, the velocity is the gradient back-propagated from the forward lithography simulation process of the partial coherent imaging system. All the techniques used in the previous pixel-based framework can be seamlessly migrated to the DevelSet framework, for DSO provides a proxy that allows us to perfectly control the gradient near the mask boundary. The objective function of DSO consists of the commonly used ILT loss and the PVBand loss,

$$L_{\text{DSO}} = \alpha L_{\text{ilt}} + \beta L_{\text{pvb}}. \quad (12)$$

To minimize the image difference of target image and nominal image, the ILT loss is given by

$$L_{\text{ilt}} = \sum_{x=1}^N \sum_{y=1}^N (\mathbf{Z}(x, y) - \mathbf{Z}_t(x, y))^2, \quad (13)$$

where the \mathbf{Z}_t is the target image; \mathbf{Z} is the wafer image after the lithography under the nominal condition; N is the width of the target image. To enable the evolution process differentiable, the step function in Equation (10) is approximated as

$$\mathbf{Z} = \frac{1}{1 + \exp(-\sigma_z \times (\mathbf{I} - I_{th}))}, \quad (14)$$

where σ_z is the steepness of the sigmoid function. Then the gradient of ILT loss can be expressed as

$$\begin{aligned} \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} &= 2 \times (\mathbf{Z} - \mathbf{Z}_t) \odot \frac{\partial \mathbf{Z}}{\partial \mathbf{M}} \\ &= 2\sigma_z \times \{ \mathbf{H}' \otimes [(\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M} \otimes \mathbf{H}^*)] \\ &\quad + (\mathbf{H}')^* \otimes [(\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M} \otimes \mathbf{H})] \}, \end{aligned} \quad (15)$$

where the \mathbf{H}' is the 180° rotation of the optical kernel set \mathbf{H} , and the \mathbf{H}^* is the conjugate of \mathbf{H} .

To minimize the area of PVBand, we expect the innermost/outermost wafer under min/max process conditions as

close to the target image as possible. The PVBand loss is given by,

$$L_{\text{pvb}} = (\mathbf{Z}_{\text{in}} - \mathbf{Z}_t)^2 + (\mathbf{Z}_{\text{out}} - \mathbf{Z}_t)^2. \quad (16)$$

The gradient of PVBand loss can be represented as

$$\begin{aligned} \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}} &= 2 \times (\mathbf{Z}_{\text{in}} - \mathbf{Z}_t) \odot \frac{\partial \mathbf{Z}_{\text{in}}}{\partial \mathbf{M}} \\ &+ 2 \times (\mathbf{Z}_{\text{out}} - \mathbf{Z}_t) \odot \frac{\partial \mathbf{Z}_{\text{out}}}{\partial \mathbf{M}}. \end{aligned} \quad (17)$$

The detailed derivation of Equation (17) is similar to Equation (15). Now the velocity v is

$$v = \alpha \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}}. \quad (18)$$

And the motion equation is finally derived as

$$\frac{\partial \phi_i}{\partial t} = -(\alpha \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}}) |\nabla \phi_i|. \quad (19)$$

Curvature term. As revealed in Equation (7), the evolution manner of level set is defined by several updating terms, which can be roughly divided into two categories: (1) external terms that attract the curve to the desired location-based on the data evidence, such as the inverse lithography gradient or the optimization methods, and (2) internal regularization terms on the curve shape, *e.g.* curvature and the length of the curvature. Previous level set-based methods focus on the improvements of the external terms since the internal term such as curvature requires extensive calculations to get the second-order derivatives. However, with GPU acceleration, DSO takes the maximum advantage of the effective feature in the implicit representation to obtain the curvature of the boundaries, which is practical to control the smoothness of the front and eliminate the noise points on the mask pattern. The curvature term is formally defined as

$$\kappa = \lambda \mathbf{m}_\theta |\nabla \phi_i| \operatorname{div} \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right), \quad (20)$$

where λ is the curvature weight. However, in OPC tasks, there may exist sharp corners in some parts of the masks. Directly apply curvature term on the level set evolution process may harm the lithography results. Thus, we add a weighted matrix \mathbf{m}_θ to control the curvature term, and the subscript θ denotes the \mathbf{m}_θ is predicted by the modulation branch parameters of DSN. We will introduce the modulation branch in Section III-B1 detailedly. Then the level set evolution of DSO can be described as the sum of the motion term and curvature term,

$$\begin{aligned} \frac{\partial \phi_i}{\partial t} &= -(\alpha \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}}) |\nabla \phi_i| \\ &+ \lambda \mathbf{m}_\theta |\nabla \phi_i| \operatorname{div} \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right). \end{aligned} \quad (21)$$

2) *The CUDA Implementation of DSO:* Conventional ILT-based mask optimization methods suffer from severe computational overhead, and the situation grows worse in level set-based methods. When the new terms are leveraged to improve mask printability, the new computational cost is also

introduced to the already burdensome computation system. Hence, the major challenge for DSO is to overcome the drawback of high computational effort. By implementing the entire DSO framework on the CUDA platform, we find a way to balance efficiency and performance. Next, we will detail the CUDA implementation of each term in level set algorithm, as well as engineering tricks to make our DSO framework significantly faster.

Numerical settings. The level set-based mask optimization methods focus on 2D situation with an image as the input. The space is discretized by a Cartesian grid with steps $\Delta x, \Delta y$, where the coordinates (x, y) represent the x^{th}, y^{th} pixel in the image. The first-order derivatives in space and time of Equation (21) can be approximated using finite difference techniques. We apply weighted essential nonoscillatory (WENO) [18] numerical polynomial interpolation method that uses the smoothest possible polynomial interpolation to find ϕ . And the first-order and second-order spatial derivatives of ϕ can be represented with central differences as

$$\begin{aligned} \nabla \phi_x &= \frac{1}{2} (\phi(x+1, y) - \phi(x-1, y)), \\ \nabla \phi_y &= \frac{1}{2} (\phi(x, y+1) - \phi(x, y-1)), \\ \nabla \phi_{xx} &= \phi(x+1, y) + \phi(x-1, y) - 2 \times \phi(x, y), \\ \nabla \phi_{yy} &= \phi(x, y+1) + \phi(x, y-1) - 2 \times \phi(x, y), \\ \nabla \phi_{xy} &= \frac{1}{4} [(\phi(x+1, y+1) - \phi(x-1, y+1)) \\ &\quad - (\phi(x+1, y-1) - \phi(x-1, y-1))], \end{aligned} \quad (22)$$

and the curvature term is then computed numerically with

$$\begin{aligned} \kappa &= \lambda \mathbf{m}_\theta |\nabla \phi_i| \operatorname{div} \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right) \\ &= \lambda \mathbf{m}_\theta \frac{\nabla \phi_{xx} \nabla \phi_y^2 - 2 \nabla \phi_y \nabla \phi_x \nabla \phi_{xy} + \nabla \phi_{yy} \nabla \phi_x^2}{\nabla \phi_x^2 + \nabla \phi_y^2}. \end{aligned} \quad (23)$$

CUDA-based TSDF. The first tremendous challenge is to calculate the truncated signed distance function (TSDF) on a given target image (2048×2048), in an extremely short period. The most celebrated method to calculate signed distance function is the Fast Marching Method introduced by [19]. Instead of using Fast Marching Method, we have specially designed the TSDF algorithm based on the characteristics of CUDA parallelism. In DSO, we use the target pattern as the initial mask. The first step focuses on extracting the boundary segments and calculating the distance towards the boundary using the `CUDA_TSDF` function in Algorithm 1. We apply pixel-wise Shift and XOR operation to obtain the mask boundary lines b_h, b_v (line 2-5). Then for all pixels p on mask plates we calculate the distance towards all boundary lines and select the minimum distance for each point in parallel. Finally, we apply the Equation (11) to generate the truncated signed distance function (line 6-10). For a complicated mask generated from the neural network, experimental result shows the `CUDA_TSDF` can achieve more than 98% reduction in TSDF calculation time.

CUDA-based geometry gradient and curvature term. As demonstrated in Equation (22), the numerical settings are well compatible with CUDA parallelism. The spatial derivatives of ϕ are calculated in function `CUDA_geometry_gradient` of Algorithm 1. And the curvature term can be calculated with the function `CUDA_curvature` with GPU acceleration. All the operations in Algorithm 1 such as `shift` and `XOR` are pixel-wise independent and can be parallelly performed per pixel per thread, which not only reduces the total runtime of the DSO but also makes it possible to integrate the level set evolution into neural network.

CUDA accelerated lithography simulation. According to the previous experimental analysis, lithography simulation is the most time-consuming part of the mask optimization flow, since it involves plenty of convolution operations between different kernels and the mask images. Inspired by Neural-ILT [8], we implement our CUDA accelerated lithography simulator and integrate the forward and backward functionalities into the popular machine learning framework `PyTorch`, with some engineering improvements. First, the optical kernels and corresponding weights are loaded and pinned in GPU memory throughout the optimization process, all the computations are performed on GPU to reduce the data transfer time from CPU to GPU. Second, the runtime bottleneck of the CUDA lithography simulator lies on the `CUDA_FFT` and `CUDA_IFFT` operators. Our improved `CUDA_FFT` operator runs faster than the commonly used `cuFFT` and the `torch.fft` libraries.

B. DevelSet-Net (DSN)

Although the CUDA accelerated DSO framework has achieved a remarkable speedup, there is still much room for improvement. Given the recent advance of deep learning on OPC, we propose a novel neural network with level set embeddings to improve efficiency and mask printability.

1) *Network Architecture and Training:* As illustrated in Fig. 3, the DSN is a mulit-branch neural network adopting the simple UNet [20] as the backbone. Our key contribution is the integration of level set embeddings with the conventional OPC networks.

Multi-branch pre-training. To utilize the advance of the mulit-branch neural networks, two types of losses are optimized simultaneously,

$$L_{\text{DSN}}(\theta) = L_0(\theta) + L_m(\theta). \quad (24)$$

Level set branch supervision. As illustrated in Fig. 3, different from the typical OPC networks, the level set branch predicts the initial LSF $\phi_{0,\theta}$ for DSO, instead of the pixel-wise mask. The mean square error is employed as the objective function,

$$L_0(\theta) = \sum_{(x,y)} (\phi_{0,\theta}(x,y) - \phi_{\text{gt}}(x,y))^2, \quad (25)$$

where $\phi_{0,\theta}$ is the predicted LSF with network parameters θ . The ϕ_{gt} is the ground truth LSF generate by DSO.

Algorithm 1 CUDA Level Set Algorithms

Input : Target image \mathbf{Z}_t

- 1: **function** `CUDA_TSDF`(\mathbf{Z}_t)
- 2: $\mathbf{Z}_{tu}, \mathbf{Z}_{td} \leftarrow$ Shift \mathbf{Z}_t upwards, downwards by 1 pixel;
- 3: $\mathbf{Z}_{tl}, \mathbf{Z}_{tr} \leftarrow$ Shift \mathbf{Z}_t leftwards, rightwards by 1 pixel;
- 4: $b_h \leftarrow (\mathbf{Z}_t \text{ XOR } \mathbf{Z}_{tu}) + (\mathbf{Z}_t \text{ XOR } \mathbf{Z}_{td})$;
- 5: $b_v \leftarrow (\mathbf{Z}_t \text{ XOR } \mathbf{Z}_{tl}) + (\mathbf{Z}_t \text{ XOR } \mathbf{Z}_{tr})$;
- 6: **for all** pixels on target image \mathbf{Z}_t **do**
- 7: $d_{ij} \leftarrow$ Distance from pixel p_i to boundary b_j ;
- 8: $d_i \leftarrow$ Minimum distance of point p_i in all d_{ij} ;
- 9: $\phi_{\text{SDF}} \leftarrow$ SDF matrix from all d_i ;
- 10: $\phi_{\text{TSDF}} \leftarrow$ TSDF matrix using Equation (11);
- 11: **return** ϕ_{TSDF} ;

Output : Truncated Signed Distance Function ϕ_{TSDF} ;

Input : TSDF matrix ϕ_{TSDF} ;

- 12: **function** `CUDA_geometry_gradient`(ϕ)
- 13: $\phi_u, \phi_d \leftarrow$ Shift ϕ upwards, downwards by 1 pixel;
- 14: $\phi_l, \phi_r \leftarrow$ Shift ϕ leftwards, rightwards by 1 pixel;
- 15: $\nabla\phi_x \leftarrow (\phi_r - \phi_l)/2$; $\nabla\phi_y \leftarrow (\phi_u - \phi_d)/2$;
- 16: **return** $\nabla\phi_x, \nabla\phi_y$;

Output : Geometry gradient $\nabla\phi_x, \nabla\phi_y$;

Input : TSDF ϕ_{TSDF} , geometry gradient $\nabla\phi_x, \nabla\phi_y$;

- 17: **function** `CUDA_curvature`($\phi, \nabla\phi_x, \nabla\phi_y$)
- 18: $\nabla\phi_{xx} \leftarrow$ `CUDA_geometry_gradient`($\nabla\phi_x$);
- 19: $\nabla\phi_{yy} \leftarrow$ `CUDA_geometry_gradient`($\nabla\phi_y$);
- 20: $\phi_{ul}, \phi_{ur}, \phi_{dl}, \phi_{dr} \leftarrow$ Shift ϕ to 4 diagonal directions;
- 21: $\nabla\phi_{xy} \leftarrow ((\phi_{ur} - \phi_{ul}) - (\phi_{dr} - \phi_{dl}))/4$;
- 22: $\kappa \leftarrow$ Curvature term using Equation (23);
- 23: **return** κ ;

Output : Curvature term κ ;

Modulation branch supervision. During the training process, the modulation branch aims to find the best \mathbf{m}_θ in Equation (21) for curvature term evolution in DSO, which is a boundary-aware model for detecting the curvature-sensitive areas. The idea is carried out by shifting the ground-truth TSDF ϕ_{gt} with a set of distance Δh ,

$$\begin{aligned} \tilde{\phi}_m(x,y) &= \phi_{\text{gt}}(x,y) + \Delta h, \\ \tilde{m} &= H(\tilde{\phi}_m(x,y)), \end{aligned} \quad (26)$$

where Δh is uniformly sampled from $[-20, 20]$, \tilde{m} is a set of \mathbf{m}_θ . $H(\phi)$ is the Heaviside function,

$$H(z) = \begin{cases} 1, & z \geq 0, \\ 0, & z < 0. \end{cases} \quad (27)$$

For every target image Z_t , the ground-truth of modulation branch is

$$m_{\text{gt}} = \underset{\tilde{m}}{\operatorname{argmin}} L_{\text{DSO}}. \quad (28)$$

During the training, the modulation branch learns to simulate an optimized \mathbf{m}_θ . As suggested in [21], the simple Heaviside function in Equation (27) acts on zero level set, which may get

TABLE I Mask Printability, Complexity Comparison with SOTA.

Bench	Area(nm^2)	ILT [4]			GLS-ILT [11]			PGAN-OPC [6]			Neural-ILT [8]			DevelSet		
		L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots
case1	215344	49893	65534	2478	46032	62693	1476	52570	56267	931	50795	63695	743	49142	59607	969
case2	169280	50369	48230	704	36177	50642	861	42253	50822	692	36969	60232	571	34489	52012	743
case3	213504	81007	108608	2319	71178	100945	2811	83663	94498	1048	94447	85358	791	93498	76558	889
case4	82560	20044	28285	1165	16345	29831	432	19965	28957	386	17420	32287	209	18682	29047	376
case5	281958	44656	58835	1836	47103	56328	963	44733	59328	950	42337	65536	631	44256	58085	902
case6	286234	57375	48739	993	46205	51033	942	46062	52845	836	39601	59247	745	41730	53410	774
case7	229149	37221	43490	577	28609	44953	548	26438	47981	515	25424	50109	354	25797	46606	527
case8	128544	19782	22846	504	19477	22541	439	17690	23564	286	15588	25826	467	15460	24836	493
case9	317581	55399	66331	2045	52613	62568	881	56125	65417	1087	52304	68650	653	50834	64950	932
case10	102400	24381	18097	380	22415	18769	333	9990	19893	338	10153	22443	423	10140	21619	393
Average		44012.7	50899.5	1300.1	38615.4	50030.3	968.6	39948.9	49957.2	706.9	38503.8	53338.3	558.7	38402.8	48673.0	699.8
Ratio		1.146	1.046	1.858	1.006	1.028	1.384	1.040	1.026	1.010	1.003	1.096	0.798	1.000	1.000	1.000

[†] L_2 and PVB unit: nm^2 .

stuck in the local minima. To tackle this, we replace it with the Approximated Heaviside Function (AHF) with a parameter ε ,

$$H_\varepsilon(\phi) = \frac{1}{2} \left(1 + \frac{2}{\pi} \arctan \left(\frac{\phi}{\varepsilon} \right) \right). \quad (29)$$

Thus, the objective function is

$$L_m(\theta) = \sum_{(x,y)} (H_\varepsilon(\phi_{m,\theta}(x,y)) - m_{gt}(x,y))^2, \quad (30)$$

where $H_\varepsilon(\phi_{m,\theta})$ is the output of modulation branch.

C. DevelSet (DSN+DSO) End to End Joint Optimization

As illustrated in Fig. 3, we apply the CUDA_TSDF function to facilitate the fast transform from pixel-wise target image to LSF ϕ_0 . After pre-training of the two branches of DSN, we fix all the parameters of DSN then directly feed the output of level set branch $\phi_{0,\theta}$ and modulation branch m_θ into the evolution process of DSO to generate the final mask. We choose the conjugate gradient (CG) method [13] for optimization in DSO, and follow CFL condition [13] to set the time step $\Delta t = \eta/\max(|v|)$, where v is evolution velocity in Equation (18), and η is CFL condition number.

IV. EXPERIMENTAL RESULTS

The DevelSet framework is developed with the popular deep learning framework PyTorch and CUDA platform. All the tests are performed on Linux system with 2.2GHz CPU and a single Nvidia Titan Xp GPU. The lithography engine is from ICCAD 2013 CAD Contest [17], which also provides the ten industrial M1 designs on 32nm design node as evaluation dataset. The scripts for shot count evaluation are obtained from the authors of Neural-ILT [8] to guarantee comparable results. The training set of DevelSet-Net is obtained from the author of GAN-OPC [6]. We pick $\sigma_z = 50$, $N_h = 24$, $\alpha = 1$, $\beta = 7.5$, $\lambda = 0.9$, $\varepsilon = 0.03$, and $\eta = 0.85$ for DevelSet optimization.

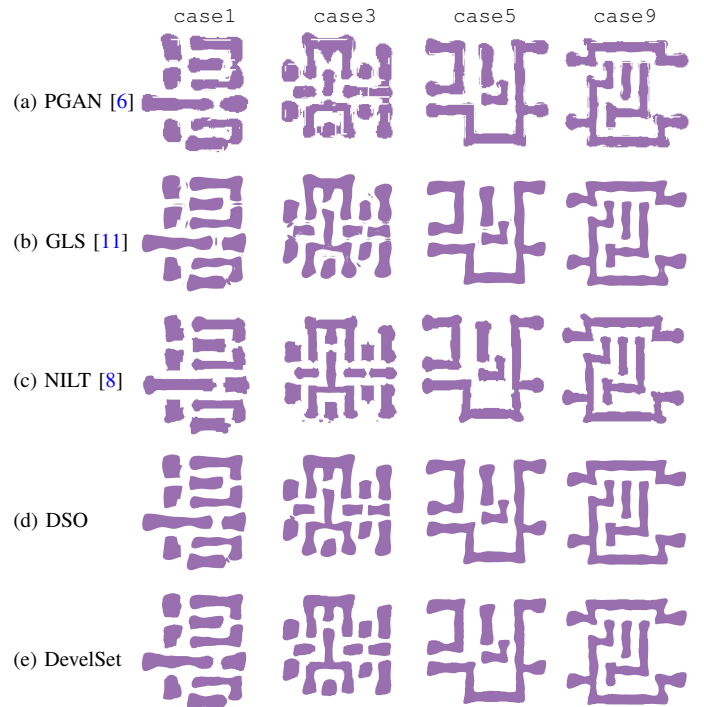


Fig. 4 Mask visualizations of: (a) PGAN-OPC [6], (b) GLS-ILT [11], (c) Neural-ILT [8], (d) DSO, and (e) DevelSet framework (DSN + DSO).

A. Comparison with State-of-the-art.

1) *Mask Printability and Complexity*: We compare the performance of the proposed DevelSet with other SOTA mask optimization methods, and the detailed values are listed in TABLE I. Compared with the conventional ILT, the L_2 and PVB are reduced by 14.6% and 4.6% respectively. Compared with the conventional level set mask optimization GLS-ILT, the L_2 and PVB could reduce by 0.6% and 2.8%. Our framework also displays superiority when compared with PGAN-OPC and Neural-ILT, which are two high-performance machine learning-based mask optimization frameworks. The

TABLE II Runtime comparison with SOTA.

Bench	ILT [4]	GLS [11]	PGAN [6]	NILT [8]	DSO	DevelSet
	TAT (s)	TAT (s)	TAT (s)	TAT (s)	TAT (s)	TAT (s)
case1	1280	123	358	13.57	3.39	1.5
case2	381	81	368	14.37	2.84	1.4
case3	1123	214	368	9.72	3.59	1.29
case4	1271	184	377	10.4	4.1	1.65
case5	1120	76	369	10.04	2.68	0.91
case6	391	65	364	11.11	2.57	0.84
case7	406	64	377	9.67	2.32	0.76
case8	388	67	383	11.81	2.67	1.14
case9	1138	63	383	9.68	2.86	1.21
case10	387	64	366	11.46	2.27	0.42
Average	788.5	100.1	371.3	11.18	2.93	1.11
Ratio	710.360	90.180	334.505	10.072	2.640	1.000

performance of L_2 could achieve 4.0% and 0.3% improvements and PVB could obtain 2.6% and 9.6% improvements, respectively. As shown in Fig. 4, these results prove the high quality of DevelSet generated masks.

Among the above-mentioned methods, the shot numbers of DevelSet reduced by 85.8%, 38.4% and 1.0% compared with ILT, GLS-OPC and PGAN-OPC. For Neural-ILT which also considers mask complexity, DevelSet generated masks contain 20.2% more shots. As depicted in Fig. 4, although the masks of DSO and DevelSet contain fewer stains, more shots are needed to keep the boundaries smooth. And the quality and simplicity of masks are a trade-off, we are more concerned about the mask printability, this performance is acceptable.

2) *Runtime Comparison*: To prove the efficiency of our DevelSet mask optimization framework quantitatively, we evaluate the turn around time (TAT) of different methods, as is shown in TABLE II. Compared with above-mentioned four methods, DevelSet could achieve significant speedup from $10\times$ to $710\times$. With DSN as pre-processing, DevelSet can achieve $2.64\times$ speedup compared with DSO only, this strongly proves the runtime performance improvement brought by DSN.

B. Ablation Study.

We conduct a set of ablation studies to evaluate the influence of each module in DevelSet. As shown in TABLE III, we list the results of DSO with curvature term and without curvature term, and the influence of modulation branch for end-to-end joint optimization of DevelSet (the DSN+DSO column). The L_2 , PVB, #shots represent the square L_2 error, the area of PVBand, and the number of shots respectively. We use the score to evaluate mask printability and mask complexity comprehensively, where the score is calculated as the sum of the L_2 , PVB, and $10\times$ #shots. The result is better if the score is smaller.

1) *The Effectiveness of Curvature Term*: We analyze the influence of the curvature term on mask complexity and printability. As the data listed in the DSO column in TABLE III, The DSO gets 805 #shots without curvature term but reduces to 726 when with the guidance of the curvature. In Fig. 5,

TABLE III Ablation study.

	DSO		DSN+DSO	
	w/o. curv.	w. curv.	w/o. mod.	w. mod.
L_2	38253.0	38454.0	39259.8	38402.8
PVB	49243.0	49398.0	48384.0	48673.0
#shots	805.0	726.0	712.8	699.8
score [†]	95546.0	95112.0	94771.8	94073.8

[†]score = $L_2 + PVB + 10 \times \text{\#shots}$.

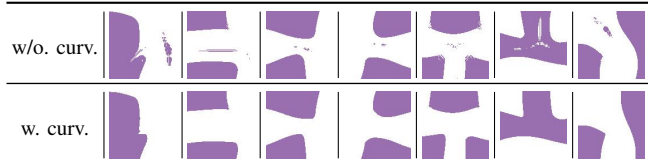


Fig. 5 Visualizations for ablation study of the curvature term.

we compare several parts of different masks to illustrate the influence of the curvature term. As shown in figures, the curvature term makes the mask boundaries more smooth and eliminates the isolated stains and edge glitches. Although the L_2 and PVB all become somewhat worse, which confirms our assumption that the curvature will harm mask printability a bit, the total score drops indicating the loss caused by the curvature term is acceptable.

2) *The Necessity of DSN and Modulation Branch*: Comparing the column DSO and the DSN+DSO in TABLE III, we find that the DSN boosts the overall performance of DevelSet by the end-to-end joint optimization. The DSN provides better initial LSF which help the DSO overcome the local minima and obtain better masks. The #shots number also drops because the upsample functions of the neural network make the mask generated by DSN more regular.

Further, we apply the modulation branch with curvature term to improve the overall score, the result in DSN+DSO with mod. column reveals that the modulation branch improves mask printability and reduces complexity. With the modulation branch, we have maximized the benefits of the curvature term while minimizing its adverse impacts.

V. CONCLUSION

In this paper, we present DevelSet, a CUDA and DNN accelerated end-to-end level set OPC framework that can perform instant mask optimization in around 1 second. By introducing the curvature term into the level set algorithm, we extend the applicable scenarios of level set-based ILT methodology for mask manufacturability improvement. Moreover, a novel multi-branch neural network with level set embeddings is proposed to boost the fast convergence of DevelSet. We believe the improved level set algorithm with CUDA/DNN accelerated joint optimization paradigm will have a real impact on the industrial mask optimization solutions.

ACKNOWLEDGMENT

This work is partially supported by Research Grants Council of Hong Kong SAR CUHK14209420, CUHK14208021.

REFERENCES

- [1] 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 μm ASIC," in *IEEE International Symposium on Quality Electronic Design (ISQED)*, 2000, pp. 81–85.
- [2] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process variation and pattern fidelity aware mask optimization algorithm," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 238–245.
- [3] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 35, no. 8, pp. 1345–1357, 2016.
- [4] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 52:1–52:6.
- [5] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 81–88.
- [6] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.
- [7] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.
- [8] 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 *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.
- [9] J. A. Sethian and D. Adalsteinsson, "An overview of level set methods for etching, deposition, and lithography development," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 10, no. 1, pp. 167–184, 1997.
- [10] Y. Shen, N. Wong, and E. Y. Lam, "Level-set-based inverse lithography for photomask synthesis," *Optics Express*, vol. 17, no. 26, pp. 23 690–23 701, Dec 2009.
- [11] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A GPU-enabled level set method for mask optimization," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2020.
- [12] Y. Shen, N. Jia, N. Wong, and E. Y. Lam, "Robust level-set-based inverse lithography," *Optics Express*, vol. 19, no. 6, pp. 5511–5521, 2011.
- [13] W. Lv, S. Liu, Q. Xia, X. Wu, Y. Shen, and E. Y. Lam, "Level-set-based inverse lithography for mask synthesis using the conjugate gradient and an optimal time step," *Journal of Vacuum Science & Technology B*, vol. 31, p. 041605, 2013.
- [14] Z. Geng, Z. Shi, X.-L. Yan, K.-S. Luo, and W.-W. Pan, "Fast level-set-based inverse lithography algorithm for process robustness improvement and its application," *Journal of Computer Science and Technology*, vol. 30, no. 3, pp. 629–638, 2015.
- [15] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.
- [16] H. Hopkins, "The concept of partial coherence in optics," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 208, no. 1093. The Royal Society, 1951, pp. 263–277.
- [17] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 271–274.
- [18] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, "Uniformly high order accurate essentially non-oscillatory schemes, iii," *Journal of Computational Physics*, vol. 71, no. 2, pp. 231–303, 1987. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0021999187900313>
- [19] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.
- [20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [21] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.