A GPU-Enabled Level-Set Method for Mask Optimization

Ziyang Yu, Guojin Chen, Yuzhe Ma, *Member, IEEE*, and Bei Yu, *Member, IEEE*

***Abstract*—As the feature size of advanced integrated circuits keeps shrinking, resolution enhancement techniques (RETs) are utilized to improve the printability in the lithography process. Optical proximity correction (OPC) is one of the most widely used RETs aiming at compensating the mask to generate a more precise wafer image. In this article, we put forward a level-set-based OPC approach with high mask optimization quality and fast convergence. In order to suppress the disturbance of the condition fluctuation in the lithography process, we propose a new process window-aware cost function. Then, a novel momentum-based evolution technique is adopted, which demonstrates substantial improvement. We also propose a self-adaptive conjugate gradient method that promises a higher optimization stability and less consuming time. Moreover, the graphics processing unit (GPU) is leveraged for accelerating the proposed algorithm. We take the output masks from a machine learning-based mask optimization flow as the input and work as the postprocess to refine the quasi-optimized masks. Experimental results on ICCAD 2013 benchmarks show that our algorithm outperforms all previous OPC algorithms in both solution quality and runtime overhead.**

***Index Terms*—Design for manufacturability (DFM), graphics processing unit (GPU), level set, mask optimization, optical proximity correction (OPC), process variation.**

I. Introduction

IN THE past decades, much progress has been made in optical lithography technology. In the lithography process, pixelated optical masks are shaped in design patterns and projected on the wafer images. However, the resolution of the lithography system is proportional to the wavelength of the lithographic source light, and it is inversely proportional to the size of the mask due to the diffraction effect [1]. Thus, it becomes more and more challenging to further downscale the transistor since the feature size is already much smaller than the light source wavelength (193 nm).

To further extend the resolution limit, several resolution enhancement techniques (RETs) are proposed for mask optimization. Optical proximity correction (OPC) as a major RET aims at compensating for the distortion of the printed image by predistorting the shape of the mask pattern. Generally speaking, OPC can be divided into three categories: 1) rule-based OPC [2]; 2) model-based OPC [3]–[5]; and 3) inverse lithography technique (ILT) [6]–[8]. The key to rule-based OPC is predetermining a set of empirical correction rules applying to different kinds of features on design patterns. This is easy to implement but could only improve the local fidelity. Model-based OPC adopts mathematical models to represent the lithography process and moves the edge segments on the mask gradually.

As a critical OPC method, ILT treats mask optimization as an inverse imaging problem that can be solved numerically. It aims at optimizing the carefully designed objective function and adjusting the pixelwise mask backward. A variety of attempts have been made in ILT to improve both the printed pattern fidelity and the process robustness [6], [7], [9]–[11].

Back in the early 1990s, Liu and Zakhor [12], [13] proposed a branch-and-bound algorithm together with a simulated annealing algorithm to systematically design predistorted masks. However, this method turns out to be very time consuming. Sherif *et al.* [14] introduced the linear objective function with unconditional constraints to solve the nonlinear mask optimization problem iteratively. Granik [15] discussed and compared the linear, quadratic, and nonlinear methods to solve the inverse mask optimization problem and presented a model-based algorithm with SRAF insertion [16]. Yu and Pan [17] proposed a topological invariant pixel-based OPC to balance the printed contour fidelity and the degree of mask manufacturing difficulty. Poonawala and Milanfar [6], [18] put forward the model-based OPC and employed the steepest descent algorithm to improve the efficiency of the optimization process. They then developed the regularization framework to enhance the quality of aerial image [9]. Shen *et al.* [19] performed 2-D discrete cosine transformation (DCT2) on the target image, which could keep its feature while reducing the mask complexity. Zhang *et al.* designed a new pixel-based cost function [20] which could reduce the computational complexity and lessen the dependence on the initial conditions. Jia *et al.* [10] incorporated the focus variation into the source-mask optimization cost function to improve both the printed pattern fidelity and the process robustness. Liu and Shi [11] used the homotopy continuation framework to accelerate the optimization and improve the mask manufacturability. Gao *et al.* [7] pushed this further by directly optimizing the edge placement error (EPE) and suppressing the process variation band (PV Band) simultaneously.

Manuscript received 19 October 2021; revised 28 February 2022; accepted 26 April 2022. Date of publication 18 May 2022; date of current version 20 January 2023. This work was supported in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14208021 and Project CUHK14209420. The preliminary version has been presented at the IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE) in 2021 [DOI: 10.23919/DATE51398.2021]. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Bei Yu.)*

Ziyang Yu, Guojin Chen, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Yuzhe Ma is with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China.

Digital Object Identifier 10.1109/TCAD.2022.3175939

1937-4151 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

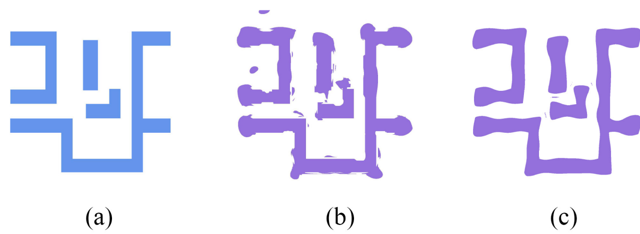

Fig. 1. Comparison of mask complexity. (a) Is the target pattern. (b) Is the mask generated by pixelwised OPC [7]. (c) Is mask generated by our proposed level-set method.

However, the mask optimized by the pixelwise ILT still contains unwanted tiny outliers and fractures, creating obstacles to the mass production. As an alternate ILT strategy, the level-set algorithm has been widely explored [21]–[24]. Different from regarding every pixel on the mask as an isolated unit, the level-set method tracks the evolution of the mask boundary to reduce the geometric deviation in the final printed image [25]. This improves the mask continuity and suppresses the degree of irregularity. Fig. 1 compares the mask generated by pixel-based ILT and level-set algorithm. Given the desired pattern in Fig. 1(a), the mask in Fig. 1(b) contains many unwanted outliers and fractures while the mask in Fig. 1(c) is much cleaner.

The mathematical expression of the level set is first proposed in [25], which described the technical details of the propagation of the mask fronts. In [26], the level-set method was adopted in the lithography processing problem. It was further used to solve inverse problems with constraints [27]. Shen *et al.* [21] applied the level-set method to the inverse lithography problem and developed the mask optimization framework. Besides, they further considered the process variations as random variables and incorporated them into the framework for robust design [22]. Lv *et al.* [23] enhanced the computational efficiency and pattern fidelity by adopting conjugate gradient (CG) descent and adjusting the time step. Geng *et al.* [24] modified the objective function taking the focus and variations into account and used the hybrid CG method to achieve stable convergence. However, the performance can still be improved. In conventional gradient descent optimization, the search direction could oscillate in different directions. Besides, when approaching the local optimal, the learning rate keeps decreasing. Those could lead to slow convergence and sometimes get stuck in local optimum. To overcome this, the momentum-based gradient method utilizes the gradient from past iterations to decide the search direction by introducing a momentum term. The search path could be smoother and the direction becomes more steadily toward global optimal using the momentum strategy.

In this article, we develop a comprehensive momentum-based level-set method to acquire better fidelity printed patterns. We propose a new process window-aware cost function that could not only suppress the influence of the process condition variation but also help reduce the EPE. To get better convergence and improve the optimization stability, we also develop a novel self-adaptive CG (SACG) method which could switch from different kinds of search velocities automatically. The compute unified device architecture (CUDA) toolkit is also used to perform graphics processing unit (GPU) acceleration which could substantially reduce the runtime. In recent deep learning-based mask optimization flow [28], the output masks from neural networks are quasi-optimized which need further refinement. Besides, those generated masks are in irregular shapes which are challenging for mask optimization flows. As a high-performance ILT flow, our method is also robust enough to handle rough initial masks, which makes it practical for our method to work as a post-refinement of other quasi-optimized mask optimization flows. We apply the mask filters with alternate sizes on the irregular input masks and balance between the complex local features with global properties. We test our method on the benchmarks released by IBM on ICCAD 2013 contest [29], and the results show our method could outperform the top winner of ICCAD 2013 and several previous methods. The main contributions of our methods are listed as follows.

1) We propose a novel process variation-based cost function, which could suppress the PV Band meanwhile reducing the EPE.
2) We develop a momentum-based SACG method to improve the convergence and enhance the optimization stability.
3) We adopt the GPU acceleration scheme and reduce the time of the optimization notably.
4) We take the roughly optimized masks as the input and adjust the resolution to improve the computational efficiency and mask quality.
5) We perform experiments on ICCAD 2013 contest benchmarks and the results turn out to be prominent among the top winner of the contest and some previous algorithms.

The remainder of this article is organized as follows. Section II gives an introduction to the lithography process and formulates the inverse lithography problem, including two important evaluation metrics. Section III gives the detailed elaboration of the level-set framework with an efficient process variation-based cost function, the input mask resolution adjustment strategy, and a novel SACG method. Section IV details experimental results and comparisons, followed by conclusion in Section V.

II. PRELIMINARIES

In this section, we introduce the preliminaries on the lithography model and the mask optimization problem. Related variables and mathematical operators are listed in Table I.

A. Lithography Process

The lithography process is composed of an optical projection model and a photoresist model. The incident light passes through the mask and sends the spatial information of the mask pattern $\boldsymbol{M}(x, y)$ into the optical projection system. The input light intensity distribution is transformed to the aerial intensity distribution $\boldsymbol{I}(x, y)$ on the wafer plane. Due to the diffraction effect, the aerial image can be expressed based on Hopkins

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

TABLE I
VARIABLES AND OPERATORS USED IN THIS ARTICLE

$\boldsymbol{R}^*$	Target image
$\boldsymbol{R}$	Printed image
$\boldsymbol{M}$	Mask
$\boldsymbol{I}$	Aerial image
$\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_K\}$	Optical kernels
$\{\boldsymbol{h}_1^\dagger, \ldots, \boldsymbol{h}_K^\dagger\}$	The conjugate transpose of optical kernels
$\odot$	Element-wise matrix multiplication operator
$\otimes$	Convolution operator

diffraction theory [30]

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

In above equation, $\boldsymbol{h}_k(x, y)$ is the kth optical kernel function and μ_k is the corresponding weight. We apply the Kth order approximation to simplify the simulation, and $K = 24$ is the total number of optical kernels in accordance to the contest [29].

The aerial image $\boldsymbol{I}$ is then transformed into the wafer image $\boldsymbol{R}$ by comparing the aerial intensity to the photoresist intensity threshold. To simulate this process, we adopt the constant threshold model here. The mathematical expression is given as follows:

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

where I_{th} is the intensity threshold that controls the binary image on the wafer plane.

B. Inverse Lithography Technique

Due to the low-pass property of the band-limited lithography system, the printed wafer image $\boldsymbol{R}$ is typically a blurred version of the input mask $\boldsymbol{M}$ as is written in (1).

The objective is to synthesize a predistorted binary mask so that the corresponding printed image could be as close to the target image $\boldsymbol{R}^*$ as possible. Mathematically, this equals to minimizing the geometric distance between the nominal printed image with the target image

$$\boldsymbol{M}^* = \underset{\boldsymbol{M}}{\operatorname{argmin}} \left\| \boldsymbol{R} - \boldsymbol{R}^* \right\|^2. \tag{3}$$

In above equation, $\|\cdot\|$ is the Euclidean norm (L_2 norm). Once the image is printed on the wafer plane, several metrics are measured to evaluate the quality of the image. In the following parts, the introduction of two metrics called EPE and PV Band will be given.

C. Edge Placement Error

EPE is evaluated as the geometric distortion of the target image. As is shown in Fig. 2(a), probe points are set equidistantly on every horizontal and vertical edges. If the distance D from target to the printed image is larger than the EPE constraint th_{EPE}, we label it as an EPE violation

$$\text{EPE violation}(x, y) = \begin{cases} 1, & \text{if } D(x, y) \geq th_{\text{EPE}} \\ 0, & \text{if } D(x, y) < th_{\text{EPE}}. \end{cases} \tag{4}$$

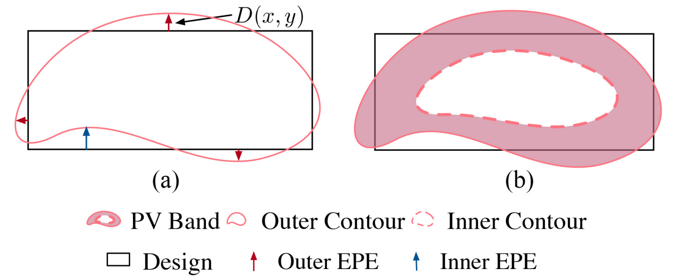

Fig. 2. Two considered metrics. (a) Measurement of EPE. (b) Measurement of PV Band.

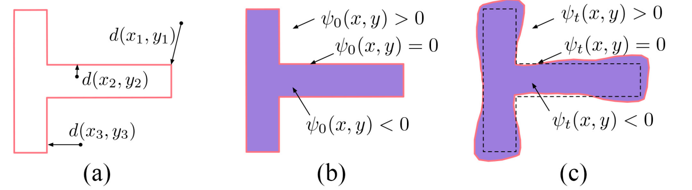

Fig. 3. Evolution process of the level-set method. (a) Perpendicular distance d from points to the mask contour. (b) Initial mask. (c) Mask pattern after t iterations.

D. Process Variation Band

In real applications, process variation could cause the deviation in the final printed image, leading to printed failure. It is essential to maintain the robustness of printed images. PV Band is utilized to evaluate such robustness, which is defined as the exclusive OR (XOR) region between the outermost and innermost contours under different process conditions, as is shown in Fig. 2(b). In this article, the depth of the focus (focus/defocus) and the intensity of incident light (dose) are considered as the process variables.

III. LEVEL-SET-BASED ILT FRAMEWORK

In the level-set framework, the mask optimization process is transformed into a contour evolution problem. Different from many ILT methods in which each pixel is taken as an optimized unit, the level-set method considers the mask boundary as a continuum. Fig. 3 gives an example of the boundary evolution process. To get the level-set function, first we need to obtain the perpendicular distance $d(x, y)$ from all points on the mask plane to the mask boundary. An example in initialization step is shown in Fig. 3(a). The initial position for the mask boundary is represented by red contour in Fig. 3(b), the contour divides the mask plane into three parts which could be characterized by the level-set function $\psi(x, y)$. The level-set function values outside and inside the mask contour are set to be positive and negative values of perpendicular distance $d(x, y)$, respectively, and the contour itself is zero boundary of the level-set function. The mathematical representation is given as follows:

$$\psi(x, y) = \begin{cases} -d(x, y), & \text{if } (x, y) \in C^- \\ 0, & \text{if } (x, y) \in \partial C \\ d(x, y), & \text{if } (x, y) \in C^+. \end{cases} \tag{5}$$

∂C is the boundary of the mask. C^- and C^+ are the inner and outer regions of the mask, respectively. Once the initial

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

mask contour is given by ψ_0, the wafer image could be simulated and evaluated. After that error information could be sent back to update the level-set function. The new mask is adjusted based on the changed function value. This is an iterative method and the mask could be updated to optimum, as is shown in Fig. 3.

A. Level-Set-Based ILT

For level-set-based ILT, we adopt the binary mask. Whether the pixel is transmitted or blocked is determined by the level-set function $\psi(x, y)$, the relation can be described as follows:

$$\boldsymbol{M}(x, y) = \begin{cases} \boldsymbol{m}_{\text{in}}, & \text{if} \quad \psi(x, y) \le 0 \\ \boldsymbol{m}_{\text{out}}, & \text{if} \quad \psi(x, y) > 0. \end{cases} \tag{6}$$

In the above equation, $\boldsymbol{m}_{\text{in}} = 1$ and $\boldsymbol{m}_{\text{out}} = 0$ means that the pixel on (x, y) is determined to be inner and outer of the mask. In general, this level-set-based ILT algorithm is developed to reduce the pattern distortion which is described in (3). The cost function can be expressed as follows:

$$L_{nom}(\boldsymbol{M}) = \left\| \boldsymbol{R} - \boldsymbol{R}^* \right\|_F^2. \tag{7}$$

To enable the backpropagation, we use the *sigmoid* function to approximate the step function in (2)

$$\boldsymbol{R} = \text{sig}(\boldsymbol{I}) = \frac{1}{1 + e^{-s(\boldsymbol{I} - I_{th})}} \tag{8}$$

where s is the steepness. By combining (1), (7), and (8), we could get the detailed expression of the nominal cost function

$$L_{nom}(\boldsymbol{M}) = \left\| \text{sig}\left[\sum_{k=1}^{K} \mu_k |\boldsymbol{h}_k(x, y) \otimes \boldsymbol{M}(x, y)|^2 \right] - \boldsymbol{R}^* \right\|^2. \tag{9}$$

Given the cost function, the velocity $\boldsymbol{v}(x, y)$ can be deduced based on [27]

$$\boldsymbol{v}(x, y) = -\frac{\partial L_{nom}(\boldsymbol{M})}{\partial \boldsymbol{M}} |\nabla \psi(x, y)| \tag{10}$$

where $|\nabla \psi(x, y)|$ is the gradient of the level-set function. We need to calculate the Jacobian of the cost function $L_{nom}(M)$ at $\boldsymbol{M}$

$$\begin{aligned} \frac{\partial L_{nom}(\boldsymbol{M})}{\partial \boldsymbol{M}} &= G(\boldsymbol{M}) = \frac{\partial \left\| \boldsymbol{R} - \boldsymbol{R}^* \right\|^2}{\partial \boldsymbol{M}} = 2\big(\boldsymbol{R} - \boldsymbol{R}^*\big) \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{M}} \\ &= 2\big(\boldsymbol{R} - \boldsymbol{R}^*\big)\boldsymbol{R}(1 - \boldsymbol{R}) \frac{\partial}{\partial \boldsymbol{M}} \\ &\quad \times \sum_{k=1}^{K} \Big(\mu_k |\boldsymbol{h}_k(x, y) \otimes \boldsymbol{M}(x, y)|^2 \Big) \\ &= 2\big(\boldsymbol{R} - \boldsymbol{R}^*\big)\boldsymbol{R}(1 - \boldsymbol{R}) \odot \frac{\partial}{\partial \boldsymbol{M}} \sum_{k=1}^{K} \mu_k \\ &\quad \times (\boldsymbol{h}_k(x, y) \otimes \boldsymbol{M}(x, y)) \odot \Big(\boldsymbol{h}_k^{\dagger}(x, y) \otimes \boldsymbol{M}(x, y) \Big) \\ &= 2 \cdot \Big\{ \boldsymbol{h}_k \otimes \Big[\big(\boldsymbol{R}^* - \boldsymbol{R}\big) \odot \boldsymbol{R} \odot (1 - \boldsymbol{R}) \otimes \Big(\boldsymbol{h}_k^{\dagger} \otimes \boldsymbol{M} \Big) \Big] \\ &\qquad + \boldsymbol{h}_k^{\dagger} \otimes \big[\big(\boldsymbol{R}^* - \boldsymbol{R}\big) \odot \boldsymbol{R} \odot (1 - \boldsymbol{R})(\boldsymbol{h}_k \otimes \boldsymbol{M}) \big] \Big\}. \end{aligned} \tag{11}$$

B. Process Variation-Aware Cost Function

To keep the optimization algorithm robust to different process conditions, a process variation-aware cost function should be taken into consideration. We propose a new cost function that could help minimize the PV band area and, meanwhile, improve the quality of printed images without aggravating increasing computational burden

$$L_{pvb}(\boldsymbol{M}) = \left\| \boldsymbol{R}_{\text{in}}(\boldsymbol{M}) - \boldsymbol{R}^* \right\|^2 + \left\| \boldsymbol{R}_{\text{out}}(\boldsymbol{M}) - \boldsymbol{R}^* \right\|^2. \tag{12}$$

To achieve a low cost value in (12), $\boldsymbol{R}_{\text{in}}(\boldsymbol{M})$ and $\boldsymbol{R}_{\text{out}}(\boldsymbol{M})$ should be both close to the target image, which is stricter than minimizing the distance between $\boldsymbol{R}_{\text{in}}(\boldsymbol{M})$ and $\boldsymbol{R}_{\text{out}}(\boldsymbol{M})$. This adjusted cost function could guide the optimization process to find the desired mask, which could generate the wafer image with a tolerable PV band and less EPE number.

The total cost function is a linear combination of (7) and (12)

$$L = L_{nom}(\boldsymbol{M}) + w_{pvb} L_{pvb}(\boldsymbol{M}). \tag{13}$$

The gradient of the total cost function can also be expressed in a separate way

$$G(\boldsymbol{M}) = G_{nom}(\boldsymbol{M}) + w_{pvb} G_{pvb}(\boldsymbol{M}) \tag{14}$$

where w_{pvb} is the weight of the PV band cost function. The calculation of $G_{pvb}(\boldsymbol{M})$ is similar to (11).

C. Self-Adaptive Conjugate Gradient Method

CG methods are proved to be efficient when solving optimization problems in large-scale systems. In our work, the CG method is applied to help evolve from the initial level-set function to the final result. Denote the velocity in the ith iteration as $\boldsymbol{v}_i$

$$\boldsymbol{v}_i = \begin{cases} -G(\boldsymbol{M})|\nabla \psi_0|, & \text{if} \quad i = 0 \\ -G(\boldsymbol{M})|\nabla \psi_i| + \lambda_i \boldsymbol{v}_{i-1}, & \text{if} \quad i > 0 \end{cases} \tag{15}$$

where λ_i is the CG coefficient characterized by the CG method. In previous work, we adopt the Polak–Ribière–Polyak (PRP) CG method proposed in [31], which is one of the widely applied formulas for λ_i. In the PRP method, the expression of CG coefficient is as follows:

$$\lambda_i^{\text{PRP}} = \frac{\|G_i(\boldsymbol{M})|\nabla \psi_i|\|^2 - G_i(\boldsymbol{M})|\nabla \psi_i| \cdot G_{i-1}(\boldsymbol{M})|\nabla \psi_{i-1}|}{\|G_{i-1}(\boldsymbol{M})|\nabla \psi_{i-1}\|^2}. \tag{16}$$

However, we observe from the experimental results that under some test cases, the evolution speed with PRP CG could be unstable, and obtained solutions can be further improved. In order to achieve better convergence, in this article, we adopt a modified SACG method to achieve better results in aspects of printed pattern fidelity and runtime.

The Fletcher–Reeves (FR) conjugate method proposed in [32] offers another choice for velocity update

$$\lambda_i^{FR} = \frac{\|G_i(\boldsymbol{M})|\nabla \psi_i|\|^2}{\|G_{i-1}(\boldsymbol{M})|\nabla \psi_{i-1}\|^2}. \tag{17}$$

It has been widely researched that the FR CG method is globally convergent on general functions, while its drawback

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

is sometimes it could cause a jam [33], [34], and for some test targets, the optimization process using only the FR CG method would take serval iterations without making significant progress to the better masks..

To overcome these problems, we combine the PRP CG with FR CG to make full use of the superiority of both methods. The CG coefficient could be adjusted in a self-adaptive manner

$$\lambda_i = \begin{cases} -\lambda_i^{FR}, & \text{if } \lambda_i^{FR} < -\lambda_i^{\text{PRP}} \\ \lambda_i^{\text{PRP}}, & \text{if } \lambda_i^{FR} \geq \left|\lambda_i^{\text{PRP}}\right| \\ \lambda_i^{FR}, & \text{if } \lambda_i^{FR} < \lambda_i^{\text{PRP}}. \end{cases} \tag{18}$$

D. Momentum-Based Updating Strategy

In our level-set mask optimization framework, the input masks could be in irregular shapes which are generated from other quasi-optimized mask optimization flow. Besides, the target patterns themselves for some test cases are sophisticated. Therefore, it is necessary to further avoid the stuck and suppress the oscillations during the update of level-set function $\psi(x, y)$. In our work, the momentum term containing the history information of velocity is utilized to guide the search direction

$$\psi_{i+1}(x, y) = \psi_i(x, y) + (\boldsymbol{v}_i(x, y) + \alpha_{\boldsymbol{v}} \boldsymbol{v}_{i-1}(x, y))\Delta t. \tag{19}$$

In the above equation, $\alpha_{\boldsymbol{v}}$ is a hyperparameter controlling the weight of velocity from the last iteration. It can be tuned from case to case. This momentum-based term could help optimize masks within fewer iterations on several benchmarks without reducing the quality.

E. Mask Filters for Irregular Input

As a robust high-performance mask optimization method, we could accept the complex masks as initial masks and refine them to get desired wafer patterns. This capability shows great practical value because our proposed mask optimization could be concatenated to other quasi-mask optimization flows and work as a postprocess. The framework is displayed in Fig. 4. We display an example of an irregular initial mask and the magnified view of a local area with and without mask filters, as can be seen in Fig. 5. It can be observed that the output mask from the previous optimization flow contains many nonrectangular complex local features. The curvilinear edges composed of a large number of short segments bring a heavy computational burden when calculating the level-set distance function. Besides, the tiny local features sometimes lead to overfitting and degrade the mask quality. In this article, we apply filters with different grid sizes on the input masks and decrease the resolution. Fig. 5 displays two filtered results using 2×2 filters and 4×4 filters, respectively, and the orange boxes are filters in different sizes. For a filter with size (f, f), we check the transmissivity in every nonoverlapping (f, f) square region. If there exists at least one transmitted pixel in an (f, f) region, then the whole region is transmitted, otherwise, it is totally blocked. In this way, we maintain the major optimized properties and abandon the minor local topological features. The calculation of the distance to the mask boundary can be significantly accelerated.

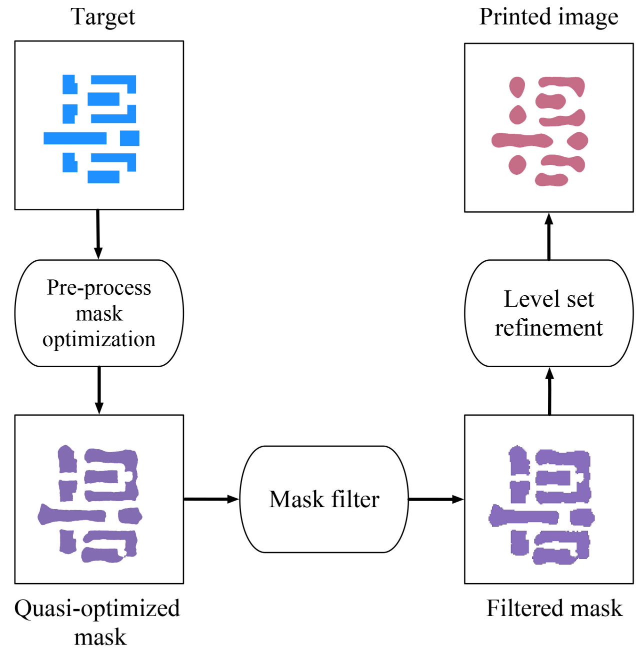

Fig. 4. Overview of Level-set mask optimization as postprocess of quasi-optimization flow.

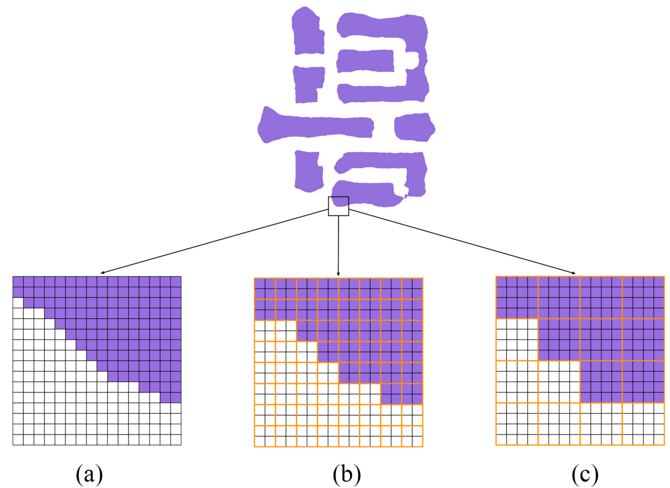

Fig. 5. Complex initial mask and the magnified details corresponding to black-box area. (a) Raw magnified mask without using filters. (b) Magnified mask area after 2×2 filter. (c) Magnified mask area after 4×4 filter.

F. Level-Set-Based ILT Mask Update Algorithm

Based on the techniques introduced above, we could build a complete iterative algorithm to generate the optimal mask using the level-set function. In the initialization stage, the initial mask $\boldsymbol{M}_0$ can either be shaped as the same with target image $\boldsymbol{R}^*$ or be generated from other quasi-optimization flows. Then, level-set function $\psi_0(x, y)$ for the initial mask can be calculated with (5) (line 1). The mask is sent into forward lithography simulator and printed on wafer plane described by (1) and (8). Once the wafer image $\boldsymbol{R}_0$ is generated, gradient of total cost function $G_0(\boldsymbol{M})$ is calculated by (11) (line 2). From (15), the starting velocity $\boldsymbol{v}_0$ is set as the negative initial gradient (line 3).

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

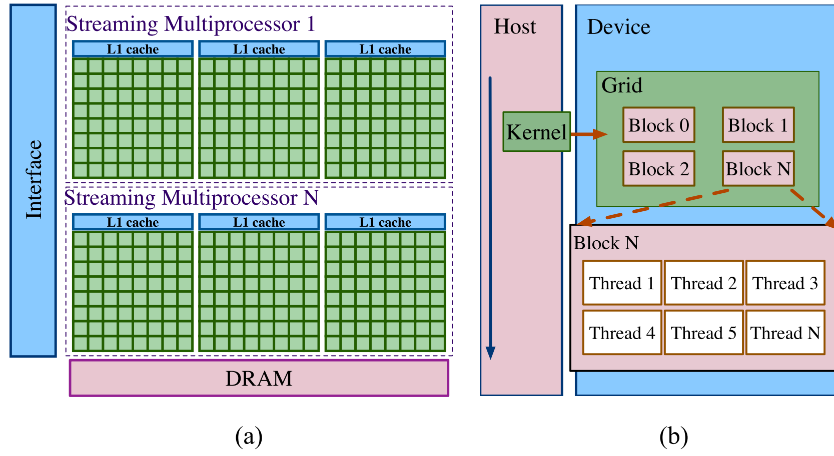

Fig. 6. Overview of GPU. (a) GPU architecture. (b) GPU programming model and thread hierarchy.

In every iteration, the optimization process starts from choosing a proper time step Δt_i (line 5). In order to keep the stability, we suppress the value of Δt_i with regards to the maximum value of evolution velocity

$$\Delta t_i = \frac{\lambda_t}{\max(|\boldsymbol{v}_i(x, y)|)}. \tag{20}$$

In the above equation, λ_t is a constant time-step coefficient. We set it to 2.5 in our experiment. Then, the change of level-set function could be calculated using the momentum strategy (line 6). The level-set function in the next iteration could be updated based on (19) (line 7), leading to the correction of the mask pattern (line 8). After that the wafer image is simulated and evaluated (line 9), the gradient for the next iteration could be computed (line 10), followed by the calculation of new velocity (line 12). The loop will continue until the preset iteration number N is achieved or the maximum velocity is less than the tolerance ϵ, which means the optimization process has found a stable result. The whole process is described in Algorithm 1.

The updated mask in the last iteration loop is chosen as the optimized mask $\boldsymbol{M}^*$.

Algorithm 1 Level-Set-Based ILT Method Flow

Require: : Target image $\boldsymbol{R}^*$, initial masks $\boldsymbol{M}_0$, optical kernels $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$, resistant model steepness s, intensity threshold I_{th}, max iteration number N, velocity tolerance ϵ.

Ensure: Optimized mask $\boldsymbol{M}^*$.

1: Initialize: $\psi_0 \leftarrow \boldsymbol{M}_0$;
2: $G_0(\boldsymbol{M}) \leftarrow \frac{\partial L_0(\boldsymbol{M})}{\partial \boldsymbol{M}}$;
3: $\boldsymbol{v}_0 = -G_0(\boldsymbol{M})|\nabla\psi(x, y)|$;
4: **repeat**
5: Time step: $\Delta t_i \leftarrow \frac{\lambda_t}{\max(|\boldsymbol{v}_i(x,y)|)}$;
6: Change: $\Delta\psi_i \leftarrow (\boldsymbol{v}_i(x, y) + \lambda_{\boldsymbol{v}}\boldsymbol{v}_{i-1}(x, y))\Delta t_i$;
7: Level-set function: $\psi_{i+1} \leftarrow \psi_i + \Delta\psi_i$;
8: Mask : $\boldsymbol{M}_{i+1}(x, y) = \begin{cases} \boldsymbol{m}_{in}, & \text{if } \psi(x, y) \leqslant 0, \\ \boldsymbol{m}_{out}, & \text{if } \psi(x, y) > 0, \end{cases}$;
9: Evaluate the printed image with Score function: $\boldsymbol{R}_{i+1}, (\boldsymbol{R}_{i+1})_{in}, (\boldsymbol{R}_{i+1})_{out}$;
10: Gradient of cost function: $G_{i+1}(\boldsymbol{M}) \leftarrow \frac{\partial L_{i+1}(\boldsymbol{M})}{\partial \boldsymbol{M}}$;
11: $\lambda_{i+1} \leftarrow$ SACG method;
12: $\boldsymbol{v}_{i+1} = -G_{i+1}(\boldsymbol{M})|\nabla\psi(x, y)| + \lambda_{i+1}^{PRP} \cdot \boldsymbol{v}_i$;
13: **until** $i \geq N$ or $|\boldsymbol{v}|_{max} \leq \epsilon$

G. GPU-Enabled Acceleration

A GPU is formed by multiple units named streaming multiprocessors (SM) as shown in Fig. 6(a). Each SM can execute many threads concurrently. CUDA is a parallel computing platform and a programming model developed by NVIDIA for its GPU. The NVIDIA CUDA Toolkit provides a development environment for creating high-performance GPU-accelerated applications. The parallel portions of a GPU application are executed on the device as kernels. A kernel is executed as a grid of thread blocks which is a batch of threads that can cooperate with each other [see Fig. 6(b)].

Previous work [35] built a learning-based independent mask printability evaluation framework, in which the author implemented matrix-based concentric circle sampling (MCCS) with CUDA to accelerate the feature extraction from layout pattern, and this served as a preparation step for machine learning model training. This can be regarded as an indirect CUDA-based speedup method in the mask optimization process. On the contrary, we implement the lithography simulation process with CUDA and accelerate the mask optimization process directly. As is expressed in (1), the level-set method requires massive calls of forward lithography simulation which brings a large amount of computational efforts from convolution operations. Based on the properties of convolution, we can transform the calculation of aerial image intensity into the following expression:

$$\begin{aligned} \boldsymbol{M} \otimes \boldsymbol{H} &= \sum_{k=1}^{K} \mu_k \cdot (\boldsymbol{M} \otimes \boldsymbol{h}_k) = \sum_{k=1}^{K} \boldsymbol{M} \otimes (\mu_k \cdot \boldsymbol{h}_k) \\ &= \boldsymbol{M} \otimes \sum_{k=1}^{K} \mu_k \cdot \boldsymbol{h}_k \end{aligned} \tag{21}$$

where $\boldsymbol{H}$ is the general kernel function defined as the weighted sum of the optical kernel functions. With this transformation,

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

TABLE II
BENCHMARK STATISTICS

ID	Pattern area
B1	215344
B2	169280
B3	213504
B4	82560
B5	281958
B6	286234
B7	229149
B8	128544
B9	317581
B10	102400

the general kernel function could be precomputed in a multiprocessing way. This could reduce convolution operations by K times and greatly improve the efficiency of our approach.

We apply the fast Fourier transform (FFT) algorithm to accelerate convolution operations. The FFT is a divide-and-conquer algorithm for efficiently computing discrete Fourier transforms (DFT) of complex or real-valued data sets. By efficiently converting convolution operations between point-value representation and coefficient representation, the FFT algorithm reduces the convolution computations from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log(N))$. However, by profiling the runtime of the level-set method, the runtime of the CPU-based FFT algorithm takes nearly ninety percent of the total runtime. We implement the FFT algorithm using the CUDA Toolkit for GPU acceleration. Our GPU-based FFT algorithm provides a simple interface to compute FFTs by leveraging the parallelism of the GPU, which reduces the total runtime to a large extent. Moreover, (21) can be computed with GPUs in parallel to further reduce the total runtime. To the best of our knowledge, we are the first to implement the CUDA-based acceleration on the lithography simulation process. This hardware-based acceleration is proved efficiently and can reduce the runtime of one iteration significantly in Section IV. However, it is also a general acceleration strategy and can be applied smoothly in many other OPC frameworks, only if it is an ILT-based method.

IV. EXPERIMENTAL RESULTS

Our level-set ILT algorithm is implemented in C/C++. We adopt the 193 nm wavelength lithography system with a defocus range of ±25 nm and a dose range of ±2%, which is provided in the ICCAD 2013 contest [29]. We adopt 24 optical kernel functions in the optical model, and the threshold intensity in the photoresist model is 0.225. The PV band metric system and EPE checker module are also provided in the contest. For all following experiments, the targets are ten benchmarks composed of rectangles and polygons in different shapes released by IBM [29]. Each benchmark layout is a 32 nm 1× metal layer. The size of the image is 2048 nm × 2048 nm with the resolution of 1 nm^2 per pixel. The pattern area of each benchmark is listed in Table II. The outermost final printed pattern is generated at nominal focus and +2% dose while the innermost printed pattern is generated at defocus and −2% dose. The EPE violation threshold th_{EPE} is set to 15 nm. EPE is measured on the sample points located on the pattern edges every 40 nm. In our experiment, target patterns B1 and B3 are the most complex, and the primary objective is to optimize the masks under nominal condition. The weights of the PV band cost function w_{pvb} in (14) for them are 0.2. For other targets, the w_{pvb} are set to be 2.5. The weights of velocity from the last iteration in (19) for different targets are set differently, ranging from 0.2 to 0.5.

To evaluate the effectiveness of our SA-Level-set mask optimization method quantitatively, we adopt four metrics described in [29], which are the number of EPE (#EPE), PV Band area (PVB), runtime (RT), and the number of shape violations. The score function is the linear combination of those metrics

$$\text{Score} = \text{RT} + 4 \times \text{PVBand} + 5000 \times \#\text{EPE} + 10000 \times \text{ShapeViol}. \quad (22)$$

In the above score function, ShapeViol is visually checked from the final printed image.

A. Comparison With Different Level-Set Methods

In the first experiment, we compare our mask optimization methods with recent other level-set-based methods. We implement our mask optimization flow on a single Nvidia Tesla P100 GPU accelerator. The initial masks are shaped as the same with target patterns. The lithography model is the same in [29], and the metrics are calculated using (22). It is worth mentioning that Shen *et al.* [21], [22] and Lv *et al.* [23] used different benchmarks and metrics. For fair comparisons, we build the flow described in these papers on our own and test them on an Intel Xeon E5-2690 V4 CPU with 2.6 GHz and 32-GB RAM to obtain the converged results. The detailed performance results are listed in Table III. The "Level set" represents the primary level-set-based optimization flow proposed in [21]. The "Robust-Level set" in [22] considers the influence of aberration conditions and aims to minimize the expectation-orient objective function under different conditions. The "CG-Level set" in [23] includes the PRP CG method and Euler time step. The "FLSB-ILT" in [24] adds the PV Band-based objection function to nominal cost. The "M-Level set" in [36] represents our momentum-based level-set function. The "SA-Level set" improves the M-Level set by adopting the SACG method. Compared with previous level-set optimization methods, our momentum-based level-set mask optimization tactfully assembles the process variation-aware cost function, CG method, and momentum-based updating strategy, and it successfully gets better scores than previous level-set-based methods.

In the last three columns of Table III, we apply the SACG method and test the performance on the same test cases. The SA-Level-set method could reduce the PVB value slightly compared to the momentum-based level-set method for most test cases without increasing the EPEs and, thus, achieves 0.7% improvements for the total scores. It can be seen the significant contribution of the SACG method is accelerating the convergence and reducing the runtime. Since the weight of the runtime term in score function (22) is small compared

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

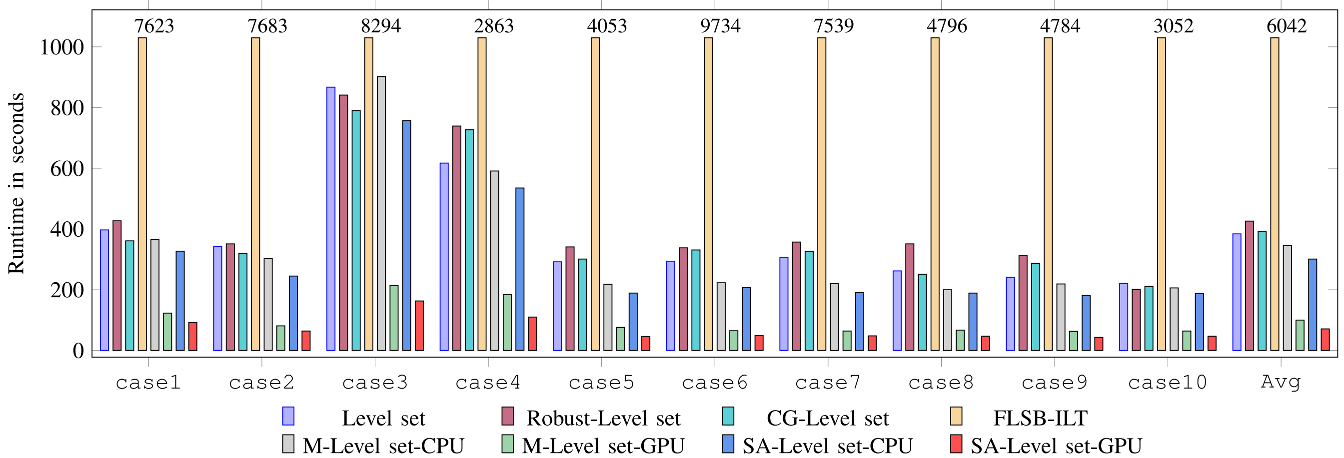

Fig. 7. Runtime comparison with different level-set-based methods. For better display, we truncate the values above 1000 s, for those cases, the exact run time is marked on the top.

TABLE III
COMPARISON BETWEEN DIFFERENT LEVEL-SET-BASED METHODS

ID	Level set [21]			Robust-level set [22]			CG-level set [23]			FLSB-ILT [24]			M-Level set [36]			SA-Level set		
	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score
B1	7	60672	278085	7	59934	275163	8	62094	288737	15	53576	296927	4	62693	270895	4	60656	262716
B2	1	51518	211415	3	50576	217655	3	50518	217392	6	43414	211339	1	50724	207977	1	49914	204720
B3	45	103597	690255	41	100704	658657	48	90718	653662	49	81593	579666	29	100945	598994	29	100945	598943
B4	1	29868	125089	1	28267	118807	1	29437	123475	5	29505	145883	0	29831	119508	0	29524	118206
B5	1	58338	238644	1	56626	231845	1	58585	239641	2	54155	230673	1	56510	231116	1	56023	229138
B6	1	52825	216594	1	51454	211154	1	52927	217039	1	51841	222098	1	51204	209881	1	50897	208637
B7	0	46746	187291	0	45536	182501	0	46827	187634	2	43934	193275	0	45056	180288	0	44921	179732
B8	1	22957	97090	1	22835	96691	1	22709	96087	3	23554	114012	1	22757	96095	1	22508	95079
B9	1	64636	263785	0	63391	253876	0	64588	258639	4	61479	270700	0	64597	258466	0	62177	248751
B10	0	19109	76657	0	19082	76529	0	19082	76539	6	19258	110084	0	18769	75140	0	18780	75167
Avg.			238491			232288			235885			237466			223722			**222109**
Ratio			1.074			1.046			1.062			1.069			1.007			**1.000**

Pattern area / PVB unit: nm^2

with other terms, this improvement is considerable. The results show the SACG method could enhance the level-set updating algorithm and help to find a better resolution.

We separately present the runtime comparison of our methods with previous level-set-based methods. The runtime is measured from the input of the target pattern to the production of the optimized mask. For all methods listed in Table III except for FLSB-ILT, the iterative optimization process stops either when the maximum velocity of mask contour is smaller than the threshold $\epsilon = 0.001$, or the preset maximum iteration number $N = 50$ is reached, as described in Section III-F. For FLSB-ILT in [24], the optimization method stops when the root mean square (RMS) of cost function (13) is smaller than the tolerance value 0.1 or the maximum iteration number is met. The results are presented in Fig. 7. It is worth mentioning that for the method that consumes time larger than 1000 s for one test case, we truncate the bar to 1000 s and add the exact runtime on top of it. As introduced in Section III-G, the FFT part consumes more than 90% of the total runtime on our CPU machine. This FFT algorithm could be implemented in a multiprocessing way with GPU acceleration and achieves a speedup of more than 5×. The overall efficiency could thus be largely improved. By adopting GPU acceleration, we significantly reduce the computational time in all cases without degenerating the fidelity or robustness of the masks. The SA-Level-set process with GPU has almost 5.4×, 6.0×, and 5.5× speedups compared with level set [21], Robust-Level set [22], and CG-Level set [23], respectively. For FLSB-ILT [24], they use the same litho model on the same test cases. FLSB-ILT consumes more than 17× more average time than the M-Level set. Compared with the SA-Level set with CPU only, it could achieve more than 4.2× speed up. Combined with the SACG method, our SA-Level-set method could obtain 1.4× and 4.8× speedup compared with GPU and CPU versions of the M-Level set, respectively, and achieve the best among all compared methods.

B. Comparison With Different SOTA Methods

In the next experiment, we compare our methods with other state-of-the-art process window-aware pixel-based OPC methods: MOSAIC_fast, MOSAIC_exact in [7], robust OPC in [4], and PVOPC in [5]. The initial masks are shaped as the same with the target pattern, which is consistent with other compared methods. In this section, our methods are also tested on a single Nvidia Tesla P100 GPU accelerator. Since all compared methods use identical metrics and benchmarks and the lithography system is also the same, the comparison can be made directly. The detailed data are listed in Table IV.

It is worth mentioning Kuang *et al.* [4] offered their detailed data to us, based on which we are able to compute the score values using (22). Although the tradeoff between EPE counts

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
COMPARISON WITH STATE-OF-THE-ART OPC ALGORITHMS

ID	MOSAIC_fast [7]			MOSAIC_exact [7]			Robust OPC [4]			PVOPC [5]			M-Level set [36]			SA-Level set		
	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score	#EPE	PVB	Score
B1	6	58232	263246	9	56890	274267	0	66218	265150	2	58269	243240	4	62693	270895	4	60656	262716
B2	10	47139	238812	4	48312	214493	0	53434	213878	0	52674	210826	1	50724	207977	1	49914	204720
B3	59	82195	624101	52	84608	600955	18	146776	677256	47	81541	561367	29	100945	598994	29	100945	598943
B4	1	28244	118298	3	24723	115161	0	33266	133371	0	26960	108030	0	29831	119508	0	29524	118206
B5	6	56253	255327	2	56299	237363	1	65631	267713	4	61820	267342	1	56510	231116	1	56023	229138
B6	1	50981	209238	1	49285	204224	0	62068	248625	0	55090	220414	1	51204	209881	1	50897	208637
B7	0	46309	185475	0	46280	186761	0	51069	204495	0	51977	207982	0	45056	180288	0	44921	179732
B8	2	22482	100186	2	22342	100031	0	25898	103691	0	22869	91541	1	22757	96095	1	22508	95079
B9	6	65331	291646	3	62529	268138	1	75387	306667	0	70713	282907	0	64597	258466	0	62177	248751
B10	0	18868	75703	0	18141	73276	0	18536	74205	0	17846	71425	0	18769	75140	0	18780	75167
Avg.			236203			227467			249505			226507			223722			**222109**
Ratio			1.063			1.024			1.123			1.020			1.007			**1.000**

Pattern area / PVB unit: nm^2

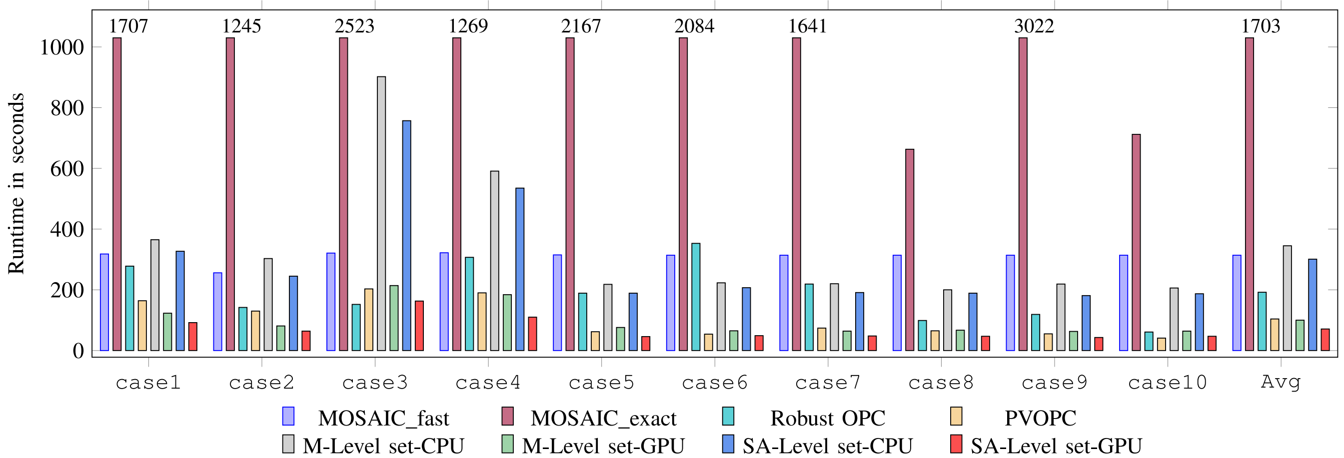

Fig. 8. Runtime comparison with different OPC methods. For better display, we truncate the values above 1000 s, for those cases, the exact run time is marked on the top.

and PV band area is still challenging to handle, our methods successfully attain the best general performance with the lowest score among all the compared methods. The performance using the momentum-based level-set method without SACG is 5.6% better than the fast version of MOSAIC work (MOSAIC_fast). For the high-quality version MOSAIC_exact, our result is still 1.6% better. Compared with the rule-based robust OPC and PVOPC, our result has a general improvement of 11.5% and 1.2%, respectively.

The results show our method could generate robust and high fidelity pattern images on these benchmarks, and it reveals the potential for achieving high-quality masks for other desired patterns.

The runtime comparison is presented in Fig. 8. For MOSAIC_fast and MOSAIC_exact in [7], the iterative optimization processes stop when the RMS of gradients of all pixels is smaller than tolerance value 0.015, or the iteration number is already 20. In robust OPC [4] and PVOPC [5], the mask is updated iteratively until the total number of EPE violations is less than a threshold or the maximum number of iterations is achieved. MOSAIC_fast adopts an alternate gradient method to reduce computational time. It consumes 4.1× more runtime than the SA-Level set. MOSAIC_exact sacrifices the computational efficiency to pursue a high-quality mask, the SA-Level-set method achieves almost 24× speedup on this. In an ILT flow, the simulations consume the most time in the optimization process. As is explained in [4], in Robust OPC, they only run the simulators in two process conditions for each iteration and estimate the results in the third process condition using the experimental data. Compared with it, the SA-Level set is 2.7× faster. The rule-based PVOPC also consumes 1.46× more time than the SA-Level set.

Fig. 9 displays the optimized masks of ten test cases together with corresponding printed patterns and PV Bands. The mask images are in good accordance with the target images. Outliers and fractures are reduced, which makes the masks more manufacturing friendly.

The wafer images of cases B5 and B6 in first ten iterations are shown in Fig. 10 as the visualization of the optimization process. For the first several iterations, the results are unstable and shots and cuts could appear in the printed images. However, this could be corrected in the next few steps. In our experiments, for most cases, the optimal masks could be generated in 15 iterations.

C. SA-Level Set as Robust Postprocess

To evaluate the robustness of our SA-Level set working as a postprocess, we conduct further experiments to refine the complex output masks generated from a quasi-mask optimization

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

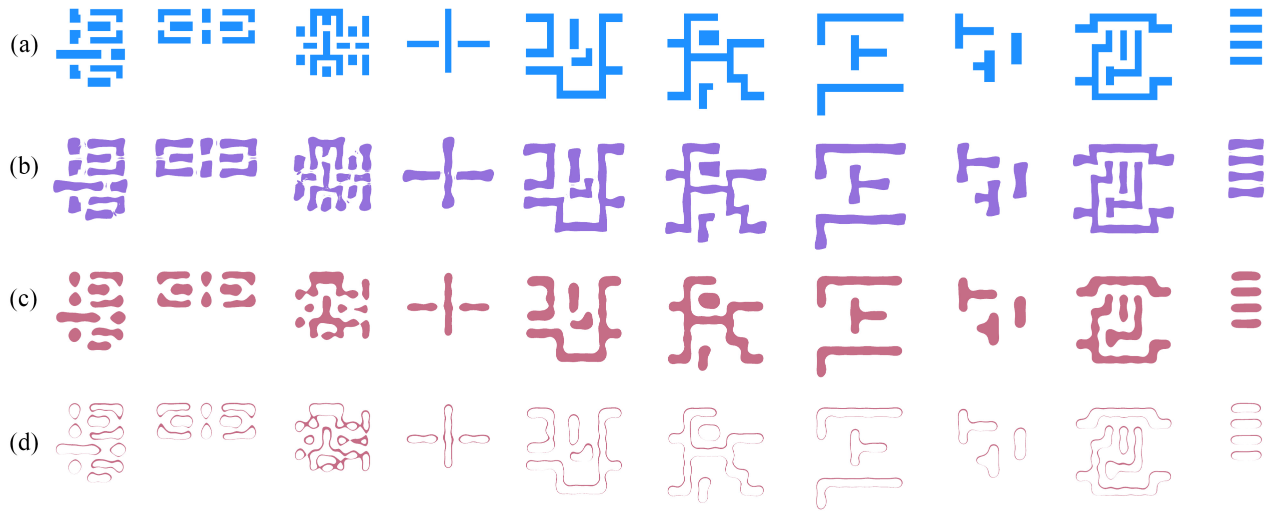

Fig. 9. Level-set inverse lithography results. Columns correspond to ten test cases. Rows from top to bottom are: (a) target patterns; (b) optimized masks; (c) printed wafer images; and (d) PV band.

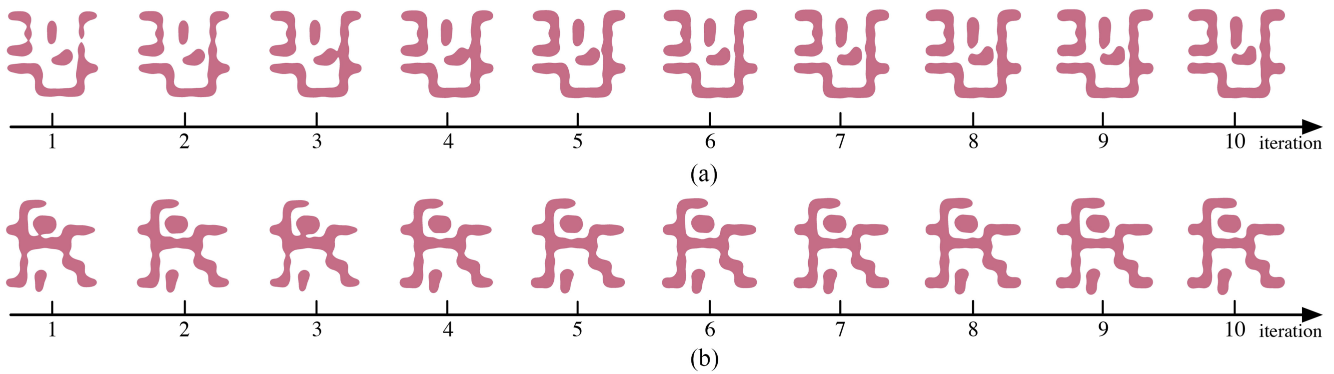

Fig. 10. Visualization of the optimization convergence in first ten iterations. The object pattern of rows from top to bottom is cases B5 and B6, respectively.

TABLE V
COMPARISON OF DIFFERENT POSTPROCESSES

Test case	MOSAIC [28]				M-Level set [36]				Filtered-Level set				FSA-Level set			
ID	#EPE	PVB	RT	Score	#EPE	PVB	RT	Score	#EPE	PVB	RT	Score	#EPE	PVB	RT	Score
B1	8	58043	448	272620	4	62019	377	268453	4	61491	357	266321	3	60696	299	259175
B2	11	53020	458	267538	1	51226	421	210325	1	50550	264	207464	1	50200	230	206030
B3	49	75644	441	628017	29	98214	396	558252	29	98214	393	558249	29	98214	394	558250
B4	5	26401	442	131046	0	31132	813	125341	0	30878	812	124324	0	30747	796	123784
B5	4	59765	469	259529	0	58675	146	234846	0	58570	187	234467	0	58502	162	234170
B6	0	54878	437	219949	0	53095	139	212519	0	52411	138	209782	0	51935	138	207878
B7	2	49156	447	207071	0	46400	115	185715	0	45974	115	184011	0	45712	139	182987
B8	4	24441	442	118206	0	23542	141	94309	0	23335	116	93456	0	23268	137	93209
B9	5	66492	452	291420	1	65996	302	269286	1	65713	260	268112	1	65327	228	266536
B10	1	21338	442	90794	0	20005	69	80089	0	19883	70	79602	0	19881	69	79593
Avg.			448	248619			292	223914			271	222579			259	221161
Ratio			1.730	1.124			1.127	1.012			1.046	1.006			**1.000**	**1.000**

Pattern area / PVB unit: nm^2

flow, the enhanced GAN (EGAN) flow in [28]. The targets are the same as those in Section IV-B. We denote the test case ID "EGAN-i" as the ith quasi-optimized input masks corresponding to the ith target. In order to observe and compare the general performance of masks using different optimization methods. We still adopt the score function in (22) as metric. It is worth mentioning that we get the executable files from Yang *et al.* [28], in order to keep the experimental conditions consistent and get comparable run time data, We run all the compared ILT refinement methods in this section on a macOS machine with a 1.4-GHz Intel i5 CPU. The filtered masks are set as initial masks in SA-Level-set optimization flow and replace the pixel-based ILT refinement engine as described in [28]. The detailed results are displayed in Table V.

The "MOSAIC" here represents MOSAIC_fast mask refinement adopted in EGAN-OPC flow, as is explained in [28].

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
MOMENTUM EFFECT COMPARISON

	#EPE	PVB	Runtime	Score
w/o. momentum	4.0	49736	81.4	224027
w. momentum	3.7	49635	70.9	221879

The M-Level set represents the EGAN generated masks with the momentum-based level-set algorithm but without using the SACG method and mask filtering. The "Filtered-Level set" represents the optimization of the filtered mask using a momentum-based level set, which the SACG method excluded. To gain a better balance between local features and global properties, and save memory storage, we use filters with the size of (2, 2) and (4, 4) to reshape the quasi-optimized masks. The "FSA-Level set" means the filtered masks optimized with the self-adaptive momentum-based level-set algorithm.

The FSA-Level-set result has a great 12.4% improvement compared with MOSAIC, which proves our complete postoptimization process is able to generate state-of-the-art masks compared with the traditional pixel-based ILT method. Compared with the Level set, the FSA-Level set could achieve 1.2% improvement, this demonstrates the significant advantages of the mask filter strategy. Besides, the mask filter strategy could adjust the complexity of the input mask and has strong robustness. We can concatenate our Level-set flow to other mask quasi-optimization flow and work as a postprocess. The FSA-Level set also exhibits 0.6% lower score value compared with the Filtered-Level set, together with the results in Section IV-B strongly proves the effectiveness of our SACG method.

We also list the detailed runtime value and compare the speed of different methods, as is shown in Table V. Compared with MOSAIC, our FSA-Level set has a great 1.73× speed up. For most cases, we could refine the same quasi-optimized masks within fewer iterations. Compared with the level set without using the mask filter strategy and the SACG method, the FSA-Level set is 1.13× faster. We further analyze the effect of the mask filter strategy and the SACG method. The Filtered-Level set could accelerate the optimization process compared with the level set, which validates the effectiveness of the mask filter strategy. But it takes 1.046× longer time compared with the FSA-Level-set result, which proves the SACG method could guide the searching algorithm to better masks with less time. This runtime superiority together with the general performance improvement could significantly prove the robustness of our FSA-Level-set method.

D. Ablation Study on Momentum Term of SA-Level Set

An ablation study is performed on the SA-Level-set optimization process with GPU to investigate the influence of the momentum strategy. The results are given in Table VI, where "w/o. momentum" represents the update of the level-set function is momentum excluded and only uses the latest velocity, while "w. momentum" refers to the momentum included the optimization process. We compare the average values in all four metrical aspects. The results show that the mask generated with momentum added could produce a more robust pattern, and the average number of EPE violations could be reduced with a smaller PV Band. Besides, the momentum strategy could help accelerate the convergence by reaching the best results in less time. Generally speaking, the momentum strategy could bring comprehensive development and improve the total score.

V. CONCLUSION

In this article, we develop a level-set inverse lithography mask optimization method with CUDA speedup to generate mask patterns with high fidelity and robustness in a very short time. We formulate the new process variation-based cost function to minimize the PV band and pattern displacement effectively. A momentum-based CG algorithm is adopted to help improve the convergence. To further enhance the mask quality and accelerate the optimization process, we propose the SACG method. The mask filters are adopted to improve the robustness for irregular input, which makes our method practical to work as a postmask optimization process. We also use GPU to accelerate the optimization process. Numerical experimental results show that our method could produce better masks in a short time, and it is robust to different process variations and complex input masks.

REFERENCES

[1] L. Rayleigh, "XXXI. Investigations in optics, with special reference to the spectroscope," *London Edinburgh Dublin Philos. Mag. J. Sci.*, vol. 8, no. 49, pp. 261–274, 1879.

[2] J.-S. Park *et al.*, "An efficient rule-based OPC approach using a DRC tool for 0.18 μm ASIC," in *Proc. IEEE Int. Symp. Qual. Electron. Design (ISQED)*, 2000, pp. 81–85.

[3] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process variation and pattern fidelity aware mask optimization algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 238–245.

[4] J. Kuang, W.-K. Chow, and E. F. Y. Young, "A robust approach for process variation aware mask optimization," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, 2015, pp. 1591–1594.

[5] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1345–1357, Aug. 2016.

[6] A. Poonawala and P. Milanfar, "Mask design for optical microlithography—An inverse imaging problem," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 774–788, Mar. 2007.

[7] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2014, pp. 1–52.

[8] Y. Ma *et al.*, "A unified framework for simultaneous layout decomposition and mask optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 5069–5082, Dec. 2020.

[9] A. Poonawala and P. Milanfar, "A pixel-based regularization approach to inverse lithography," *Microelectron. Eng.*, vol. 84, no. 12, pp. 2837–2852, 2007.

[10] N. Jia, A. K. Wong, and E. Y. Lam, "Robust mask design with defocus variation using inverse synthesis," in *Proc. Lithography Asia*, vol. 7140, 2008, Art. no. 71401W.

[11] F. Liu and X. Shi, "An efficient mask optimization method based on homotopy continuation technique," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, 2011, pp. 1–6.

[12] Y. Liu and A. Zakhor, "Optimal binary image design for optical lithography," in *Proc. Opt. Laser Microlithography III*, vol. 1264, 1990, pp. 401–412.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.

[13] Y. Liu and A. Zakhor, “Binary and phase shifting mask design for optical lithography,” *IEEE Trans. Semicond. Manuf.*, vol. 5, no. 2, pp. 138–152, May 1992.

[14] S. Sherif, B. Saleh, and R. De Leone, “Binary image synthesis using mixed linear integer programming,” *IEEE Trans. Image Process.*, vol. 4, no. 9, pp. 1252–1257, Sep. 1995.

[15] Y. Granik, “Solving inverse problems of optical microlithography,” in *Proc. Opt. Microlithography XVIII*, vol. 5754, 2005, pp. 506–526.

[16] Y. Granik, K. Sakajiri, and S. Shang, “On objectives and algorithms of inverse methods in microlithography,” in *Proc. Photomask Technol.*, vol. 6349, 2006, Art. no. 63494R.

[17] P. Yu and D. Z. Pan, “TIP-OPC: A new topological invariant paradigm for pixel based optical proximity correction,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2007, pp. 847–853.

[18] A. Poonawala and P. Milanfar, “Prewarping techniques in imaging: Applications in nanotechnology and biotechnology,” in *Proc. Comput. Imag. III*, vol. 5674, 2005, pp. 114–127.

[19] S. Shen, P. Yu, and D. Z. Pan, “Enhanced DCT2-based inverse mask synthesis with initial SRAF insertion,” in *Proc. Photomask Technol.*, vol. 7122, 2008, Art. no. 712241.

[20] J. Zhang, W. Xiong, Y. Wang, Z. Yu, and M.-C. Tsai, “A highly efficient optimization algorithm for pixel manipulation in inverse lithography technique,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2008, pp. 480–487.

[21] Y. Shen, N. Wong, and E. Y. Lam, “Level-set-based inverse lithography for photomask synthesis,” ’ *Opt. Exp.*, vol. 17, no. 26, pp. 23690–23701, Dec. 2009.

[22] Y. Shen, N. Jia, N. Wong, and E. Y. Lam, “Robust level-set-based inverse lithography,” *Opt. Exp.*, vol. 19, no. 6, pp. 5511–5521, 2011.

[23] 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,” *J. Vac. Sci. Technol. B*, vol. 31, no. 4, 2013, Art. no. 041605.

[24] 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,” *J. Comput. Sci. Technol.*, vol. 30, no. 3, pp. 629–638, 2015.

[25] S. Osher and J. A. Sethian, “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations,” *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988.

[26] J. A. Sethian and D. Adalsteinsson, “An overview of level set methods for etching, deposition, and lithography development,” *IEEE Trans. Semicond. Manuf.*, vol. 10, no. 1, pp. 167–184, Feb. 1997.

[27] F. Santosa, “A level-set approach for inverse problems involving obstacles fadil santosa,” *ESAIM Control Optim. Calc. Variations*, vol. 1, pp. 17–33, 1996.

[28] 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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2822–2834, Oct. 2020.

[29] S. Banerjee, Z. Li, and S. R. Nassif, “ICCAD-2013 CAD contest in mask optimization and benchmark suite,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2013, pp. 271–274.

[30] H. Hopkins, “The concept of partial coherence in optics,” *Proc. Roy. Soc. London A Math. Phys. Eng. Sci.*, vol. 208, no. 1093, pp. 263–277, 1951.

[31] B. T. Polyak, “The conjugate gradient method in extremal problems,” *USSR Comput. Math. Math. Phys.*, vol. 9, no. 4, pp. 94–112, 1969.

[32] R. Fletcher and C. M. Reeves, “Function minimization by conjugate gradients,” *Comput. J.*, vol. 7, no. 2, pp. 149–154, 1964.

[33] M. J. D. Powell, “Restart procedures for the conjugate gradient method,” *Math. Program.*, vol. 12, no. 1, pp. 241–254, 1977.

[34] W. W. Hager and H. Zhang, “A survey of nonlinear conjugate gradient methods,” *Pac. J. Optim.*, vol. 2, no. 1, pp. 35–58, 2006.

[35] B. Jiang, H. Zhang, J. Yang, and E. F. Young, “A fast machine learning-based mask printability predictor for OPC acceleration,” in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2019, pp. 412–419.

[36] Z. Yu, G. Chen, Y. Ma, and B. Yu, “A GPU-enabled level set method for mask optimization,” in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, 2021, pp. 1835–1838.

Ziyang Yu received the B.S. degree from the Department of Physics, University of Science and Technology of China, Hefei, China, in 2018, and the M.Phil. degree from the Department of Physics, The University of Hong Kong, Hong Kong, in 2020. He is currently pursuing the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His current research interests include design space exploration in electronic design automation and machine learning on chips.

Guojin Chen received the B.Eng. degree in software engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His current research interests include deep learning in VLSI design for manufacturability, high-performance computing, and computer vision.

Yuzhe Ma (Member, IEEE) received the B.E. degree from the Department of Microelectronics, Sun Yat-sen University, Guangzhou, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2020.

He is currently an Assistant Professor with Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include agile VLSI design methodologies, machine-learning-aided VLSI design, and hardware-friendly machine learning.

Dr. Ma received the Best Paper Awards from ICCAD 2021, ASPDAC 2021, and ICTAI 2019, and the Best Paper Award Nomination from ASPDAC 2019.

Bei Yu (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received nine Best Paper Awards from DATE 2022, ICCAD 2021 and 2013, ASPDAC 2021 and 2012, ICTAI 2019, *Integration, the VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD contest awards. He is an Editor of IEEE TCCPS NEWSLETTER. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on December 20,2024 at 09:28:49 UTC from IEEE Xplore. Restrictions apply.