Adaptive ILT via Multi-Level Lithography Simulation

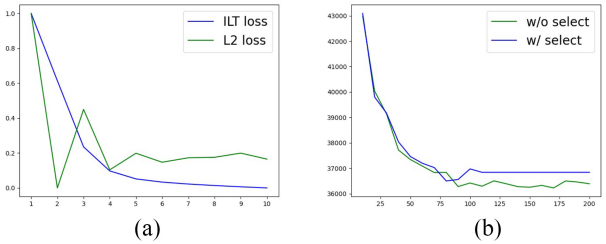
Shuyuan Sun, Fan Yang, *Member, IEEE*, Bei Yu, *Senior Member, IEEE*, Li Shang, *Member, IEEE*, and Xuan Zeng, *Senior Member, IEEE*

***Abstract*—Inverse lithography technology (ILT) is an advanced resolution enhancement technology (RET) approach that pushes the limits of current process conditions to achieve smaller feature sizes. This article presents an efficient ILT framework centered on the concept of solving mask optimization problem from approximation to precision. First, low-resolution lithography simulation is employed to guide ILT, generating a preliminary solution at a low cost. Subsequently, high-resolution lithography simulation is utilized for fine-tuning, and output the final correction results. Additionally, we introduce a continuous correction scheme and a hyperparameter selection strategy to enhance the adaptability and robustness of the ILT solver for full-chip scale layouts. For via layer patterns, we significantly improve the solver's performance by adopting a new high-resolution ILT approach combined with a restricted optimization region. Experimental results show that our method effectively reduces edge placement error (EPE) violations and improves efficiency for ILT on full-chip scale layouts. In the ICCAD 2013 benchmark, our method outperforms state-of-the-art approaches, reducing L_2 loss by at least 1.3%. For via layer patterns, our method achieves a 98% reduction in L2 loss and nearly triples the runtime efficiency compared to its earlier version. Furthermore, it delivers an 8% reduction in L2 loss and a 45% reduction in PVBand compared to other existing methods.**

***Index Terms*—Inverse lithography, optical proximity correction, multi-level lithography simulation.**

I. Introduction

As TECHNOLOGY nodes continue to shrink while the wavelength of the light source remains unchanged, the inconsistency between the printed shapes on the wafer and the designed patterns becomes increasingly pronounced, making it increasingly difficult to align them. Inverse lithography technology (ILT) is a promising approach of resolution enhancement technology (RET) that has the potential to extend the capabilities of the current technology node and achieve smaller feature sizes. ILT directly manipulates the layout pixels under the guidance of lithography simulation, giving it greater flexibility in mask correction compared to traditional model-based OPC. In [2], [3], and [4], extensive experiments are conducted to compare the correction capabilities of ILT and model-based OPC. Experimental results confirm that ILT can reduce hotspot patterns, greatly improve the process window, and provide a larger depth of focus (DoF). While the finer adjustment granularity of ILT offers many advantages, it also introduces three significant drawbacks that hinder its widespread adoption: high runtime overhead, poor mask manufacturability, and unstable correction quality.



(a) (b)

Fig. 1. (a) Shows the mismatch between the ILT loss and the L_2 loss in last 10 iterations of ILT. (b) Indicates that skipping result-checking to select the mask with the minimum L_2 loss during low-resolution ILT does not degrade the final outcome.

For high runtime overhead, recent methods have attempted to address this issue in two ways: 1) using GPU acceleration for optical simulations and 2) using deep learning to reduce the number of iterations. In [5], the computational process of forward optical simulation is modeled as a layer of a neural network. By using PyTorch [6] to leverage the powerful computational capabilities of GPUs, the optical simulation was greatly accelerated, thereby speeding up ILT. In [7], the level set-based method achieved more than a threefold acceleration on GPUs compared to CPUs.

Due to ILT's larger solution space, it requires more iterations to achieve a satisfactory result. Recent deep learning-based ILT approaches can be categorized into two types: 1) image generation-based methods, which use neural networks to predict an initial solution and thereby reduce the number of iterations [8], [9], [10] and 2) neural ODE-based methods, which employ neural networks to imitate the ILT process and simplify the computations involved [11], [12], [13].

Received 22 August 2024; revised 4 December 2024 and 1 March 2025; accepted 6 April 2025. Date of publication 15 April 2025; date of current version 22 October 2025. This work was supported in part by the National Key R&D Program of China under Grant 2020YFA0711900 and Grant 2020YFA0711903; in part by the National Natural Science Foundation of China (NSFC) Research Projects under Grant 92373207 and Grant 62090025; and in part by the Research Grants Council of Hong Kong SAR under Grant RFS2425-4S02. The preliminary version has been presented at the 60th ACM/IEEE Design Automation Conference (DAC) in 2023 [1] [DOI: 10.1109/DAC56929.2023.10247704]. This article was recommended by Associate Editor E. Young. *(Corresponding author: Fan Yang.)*

Shuyuan Sun, Fan Yang, and Xuan Zeng are with the State Key Laboratory of Integrated Circuits and Systems and the School of Microelectronics, Fudan University, Shanghai 200433, China (e-mail: yangfan@fudan.edu.cn).

Bei Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR.

Li Shang is with the State Key Laboratory of Integrated Circuits and Systems and the School of Computer Science, Fudan University, Shanghai 200433, China.

Digital Object Identifier 10.1109/TCAD.2025.3561233

For image generation-based methods, they first utilize generation models such as U-Net [14] to predict a good initial solution for ILT, and then use a small number of iterations to adjust the final output. In [8], a generative adversarial network (GAN) is introduced to enhance the robustness of image generation. In [5], a pretrained U-Net model is used as the backbone model and integrated with a ILT layer for layout correction. In [9], a deep model is proposed for mask prediction, and then the predicted mask is further optimized using the GPU-accelerated level set-based method [7]. These prior works have demonstrated runtime benefits by employing deep models for initial mask prediction. However, due to the inherent uncertainty of neural networks, the quality of the predicted masks cannot be guaranteed. Although the initial solution will be further optimized by subsequent rigorous ILT iterations, a poor initial solution may lead to a final mask quality significantly lower than that produced by the original ILT.

For neural ODE-based methods, they learn to optimize the iterative ILT solving process and significantly enhancing the correction efficiency. In [12], "learn-to-optimize" framework is proposed to unroll the iterative ILT process into a learnable neural network. In [11], an implicit learning framework is proposed to directly generate high-quality optimized masks using lithography-conditioned inputs. These methods greatly enhance the optimization efficiency of ILT, but they are heavily dependent on a good numerical ILT solver to produce a high-quality training dataset.

To improve the poor mask manufacturability resulting from ILT, many efforts have been made. In [5], a loss term to evaluate mask complexity is introduced to eliminate small shapes around major features. In [9], a curvature penalty term is introduced in the loss function, aiming to obtain smoother shape boundaries. In [15], a spatial attention map is introduced to avoid holes and outliers on the optimized layout. In [15] and [16], an average pooling operation is applied on neighboring pixels to eliminate small features. Although these methods mitigate unwanted features to a certain extent, unnatural curvatures and overly fine features still inevitably exist in the final mask.

Additionally, experimental results reveal that the correction quality of ILT is unstable during the optimization process, with this issue being more pronounced on via layer patterns. This instability arises because ILT employs approximations to map binary mask images and wafer images into continuous values between 0 and 1 using nonlinear functions. Therefore, the loss value used during ILT (referred to as ILT loss) does not fully reflect the difference between the wafer image and the target image (referred to as L_2 loss). And L_2 loss serves as a metric for assessing the printability of the mask. Fig. 1(a) shows an example of the discrepancy between the ILT loss and the L_2 loss in the last 10 iterations of ILT. Although the ILT loss continuously decreases, the L_2 loss is smallest at the second iteration. Therefore, like [8], [15], when a fixed number of iterations is specified for ILT and the final corrected result is used as the output, it may fail to satisfy the desired correction requirements. In [5], after updating the mask image in each ILT iteration, an optical simulation is conducted to check the quality of the mask. While this ensures the quality of the outputs, it significantly increases the runtime. In [11], each step is designed to improve upon the previous one, but this approach often becomes trapped in local optima.

In summary, achieving mask generation with high efficiency, high quality, and good manufacturability remains the central goal of ILT despite years of effort. To this end, we propose a multilevel correction strategy for ILT to tackle the aforementioned challenges.

Runtime Efficiency: We conduct multiresolution lithography simulations to guide ILT and optimizing the mask at different granularities, aiming to improve the runtime efficiency. The process begins with low-resolution ILT, guided by less precise lithography simulations, and then transitions to high-resolution ILT, which is directed by accurate simulations. Experimental results show that with a scale factor of 4, low-resolution ILT is about 18× faster than high-resolution ILT. This coarse-to-fine correction scheme can accelerate ILT in various situations without model pretraining. This multilevel ILT framework was introduced in [1] as the preliminary work. Additionally, we adopt the Adam optimizer to further enhance the efficiency of mask optimization for via layer patterns. For the full-chip scale experiment, we reduce the total runtime by selecting appropriate hyperparameter configurations. Although selecting the hyperparameters requires some time, the improved efficiency of the ILT solver with the optimal settings leads to an overall reduction in runtime.

Manufacturability: To enhance the manufacturability of the resulting masks, we employ an average pooling operation to merge neighboring pixels. Additionally, we employ a restricted optimization region to avoid generating too complex mask patterns in this article. And it is shown that the obtained masks is more regular and symmetrical than previous version.

Printability: In this article, we further advance the performance of multilevel ILT for via layer patterns to achieve higher printability. Given that via layer shapes are smaller and more sensitive to adjustments compared to metal layer patterns, even minor variations can cause significant discrepancies in the wafer image. To address this issue, we introduce a new high-resolution ILT, a continual correction scheme and a hyperparameter selection strategy. First, we design a new high-resolution ILT that incorporates result-checking in each iteration. Although the final output mask has minimal ILT loss, it may still fail to produce a satisfactory wafer image. This is because the discrepancy between the ILT loss and the actual L_2 loss is larger in via layer patterns, as shown in Fig. 1(a). In addition, the PVBand variation in the high-resolution ILT is also small due to the smaller shapes of via patterns. Therefore, we omit the PVBand calculation in the high-resolution ILT, saving two optical simulations per iteration. As a result, despite the new added result-checking step, the runtime of high-resolution ILT is reduced by 20% while maintaining the same number of iterations. Experimental results show that omitting result-checking during low-resolution ILT has minimal effect on the final outcome, as shown in Fig. 1(b).

Second, we introduce a continual correction scheme and a hyperparameter selection strategy to enhance the adaptability and robustness of the proposed ILT framework for large-scale datasets. A large-scale dataset consists of hundreds of tiles. While a fixed number of ILT iterations works well for some tiles, it may not be sufficient for others to meet the

desired specifications To address this, we introduce a continual correction phase, which performs additional ILT iterations for tiles that do not meet the correction requirements after the main correction phase, as shown in Fig. 2.

For hyperparameter selection, we design features to identify representative mask patterns from a large dataset and utilize Bayesian optimization (BO) to determine optimal hyperparameters based on these patterns. For smaller datasets, such as the ICCAD 2013 benchmarks with only ten test cases, we directly apply hyperparameter selection to achieve the best correction results for each individual test case using the proposed ILT solver.

In summary, the proposed ILT framework delivers high computational efficiency while producing masks with excellent manufacturability and printability. Unlike other machine learning-based ILT methods [11], [12], it does not require training data collection or model pretraining. Instead, it serves as a numerical ILT solver that can be utilized to generate training data for ML-based methods.

We apply ILT to a full-chip via layer layout, effectively reducing edge placement error (EPE) violations and L_2 loss. We also conduct an ablation study on a large-scale dataset of tiles cut from the same layout to demonstrate the effectiveness of our approach. In the ICCAD 2013 benchmark, our approach outperforms state-of-the-art methods, reducing the L_2 loss by at least 1.3%. For via layer patterns, the new method achieves a 98% reduction in L_2 loss and decreases the runtime by nearly threefold compared to the preliminary version of the proposed approach. Additionally, compared to other method, our approach reduces the L_2 loss by over 8% and the PVBand by 45%. This article primarily improves the adaptability and robustness of the original Multi-ILT method for large-scale datasets, with a focus on via layer patterns.

The remainder of this article is organized as follows. In Section II, we will present the background and evaluation metrics used in ILT. In Section III, we propose the multilevel ILT approach. In Section IV, we empirically study the multilevel ILT behaviors. In Section V, experimental results are presented. In Section VI, we conclude this article.

II. Background

A. Forward Lithography Model

In this section, we introduce the forward optical simulation process and the approximations made by ILT. The forward lithography model maps the designed mask image $\boldsymbol{M}$ to the wafer image $\boldsymbol{Z}$, incorporating both the lithography model and the photoresist model. The lithography model computes the aerial image $\boldsymbol{I}$ representing the distribution of lithography intensities. The photoresist model take the aerial image $\boldsymbol{I}$ as input, and output the final wafer image $\boldsymbol{Z}$. For simplicity, we use a compact photoresist model with a constant threshold I_{th}, as shown in the first line of (1). Because direct truncation prevents gradient backpropagation, in ILT we approximate it using the sigmoid function, as shown in the second line of (1). Following the settings in [17], we set $I_{\text{th}} = 0.225$

$$\boldsymbol{Z}(x,y) = \begin{cases} 1, & \boldsymbol{I} \geq I_{\text{th}} \\ 0, & \boldsymbol{I} < I_{\text{th}} \end{cases} \approx \frac{1}{1+\exp\left[-\alpha \times (\boldsymbol{I} - I_{\text{th}})\right]}. \tag{1}$$

Another approximation introduced by ILT involves mapping the binary mask to continuous values between 0 and 1, allowing for the calculation of gradients, as shown in (2). The approximations introduced in (1) and (2) are the primary reasons for the discrepancy between the ILT loss and the actual L2 loss. As a result, the mask obtained by ILT with the minimum loss may not be the optimal solution for the layout correction

$$\boldsymbol{M} \in \{0,1\}^{N\times N} \approx \frac{1}{1+\exp\left(-\beta \times \left(\boldsymbol{M}' - T_R\right)\right)}. \tag{2}$$

Lastly, we introduce the lithography model, which can be approximated by the Hopkin's model [18] as shown in (3), mapping the mask image $\boldsymbol{M}$ to the aerial image $\boldsymbol{I}$

$$\begin{aligned} \boldsymbol{I}(x,y) &\approx \sum_{k=1}^{N_k} w_k \|\boldsymbol{h}_k(x,y) \otimes \boldsymbol{M}(x,y)\|^2 \\ &= \sum_{k=1}^{N_k} w_k \left\| \mathcal{F}^{-1}(\boldsymbol{H}_k \odot \mathcal{F}(\boldsymbol{M})) \right\|^2 \end{aligned} \tag{3}$$

where $\boldsymbol{h}_k \in \mathbb{R}^{P\times P}$ represents the kth optical kernel, and w_k is the corresponding weight. N_k denotes the total number of kernels used for the simulation. $\otimes$ denotes the convolution operation, and $\odot$ denotes element-wise multiplication.

The first line of (3) computes the lithography intensity by convolving the optical kernel with the mask image in the spatial domain, with complexity $O(N_k \cdot P^2 \cdot N^2)$. It is well-known that convolution in the spatial domain is equivalent to multiplication in the frequency domain. Thus, computational efficiency can be enhanced by using fast Fourier transformation (FFT), as indicated in the second line of (3).

$\mathcal{F}$ and $\mathcal{F}^{-1}$ represent FFT and inverse FFT, respectively. $\boldsymbol{H}_k \in \mathbb{C}^{P\times P}$ is the frequency domain representation of $\boldsymbol{h}_k$, and $\odot$ denotes element-wise multiplication. We exclude the high-frequency components of $\mathcal{F}(\boldsymbol{M})$ to enable multiplication with $\boldsymbol{H}_k$. The complexity of $\boldsymbol{H}_k \odot \mathcal{F}(\boldsymbol{M})$ is $O(P^2)$. To restore the original dimensions, the inverse FFT has a complexity of $O(N^2 \log N^2)$. Thus, the overall complexity of lithography calculation in the frequency domain is $O((N_k+1)\cdot N^2 \log N^2 + N_k \cdot P^2)$. In our implementation, we use $P = 35$, $N = 2048$, and $N_k = 24$. Given that $P \ll N$, the bulk of the computation is dominated by the N_k inverse FFTs.

B. Evaluation Metrics and Problem Formation

Definition 1 (Squared L_2 Loss): Given the target image $\boldsymbol{Z}_t$ and the wafer image $\boldsymbol{Z}_{\text{norm}}$, the squared L_2 loss is calculated as $\|\boldsymbol{Z}_{\text{norm}} - \boldsymbol{Z}_t\|_2^2$. The wafer image $\boldsymbol{Z}_{\text{norm}}$ is obtained under nominal dose and focus conditions and will also be referred to as "L2".

Definition 2 (PVBand): The process variation band (PVB) represents the XOR area between two aerial images, $\boldsymbol{Z}_{\text{in}}$ and $\boldsymbol{Z}_{\text{out}}$, generated under different conditions.

We adopt the same settings as [17]: $\boldsymbol{Z}_{\text{in}}$ is generated under the defocus and -2% dose condition. $\boldsymbol{Z}_{\text{out}}$ is generated under nominal focus and $+2\%$ dose condition.

Definition 3 (EPE): EPE is a metric used to evaluate the distortion of the wafer image relative to the target image. Measurement points are evenly distributed along the horizontal and vertical edges of the target image. If the

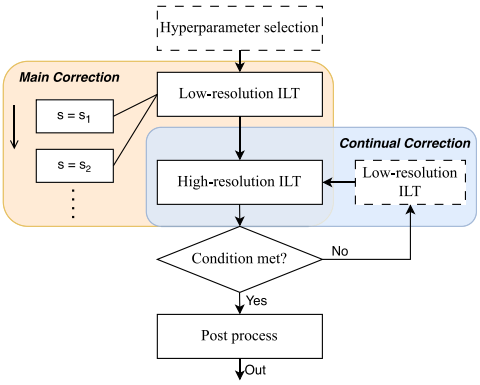


Fig. 2. Complete framework of our method.

vertical distance $D(x, y)$ from the printed contour to the target contour exceeds a threshold distance *thr*, it is considered an EPE violation. This metric will also be referred to as 'EPE'.

Equation (4) shows an example of EPE violation. We set the threshold distance *thr* to 15 nm, following the setting of [17]

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

Definition 4 Mask Fracturing Shot Count: The mask fracturing shot count represents the number of rectangles used to replicate the optimized curvilinear mask shapes [5]. This metric evaluates the complexity of mask patterns.

Problem 1 (Mask Optimization): Given a target layout $\boldsymbol{Z}_t$, our goal is to obtain an optimized mask $\boldsymbol{M}$ within an acceptable runtime, aiming to improve both the printability and manufacturability of the masks.

III. Algorithms

A. Enhanced Multi-ILT Framework

The overall framework of our method is illustrated in this section and shown in Fig. 2 and Algorithm 1. We first perform low-resolution ILT with different scale factors s_i from large to small to achieve computational efficiency. Then, high-resolution ILT is employed to fine-tune the layout and reduce mask complexity. The resulting output is then checked to determine if it meets the required standards. If it does not, the process enters a continual correction phase (lines 3 in Algorithm 1). We detail the low- and high-resolution ILT in Section III-B. Different loss functions are adopted according to their different accuracy requirements.

B. Multilevel ILT

First, we define two loss functions L used in ILT, as shown in (5) and (6). The loss function (5) consists of two parts: the first term calculates the L_2 loss, and the second term calculates the PVBand. $\boldsymbol{Z}_{\text{norm}}$ is replaced by $\boldsymbol{Z}_{\text{out}}$ in the calculation of the L_2 loss to save computation time and GPU memory during the optimization process. For mental layer patterns, we use (5) for both low- and high-resolution ILT. Because the shapes in the metal layer are relatively large, PVBand still needs to be considered even in high-resolution ILT

$$L_a = \|\boldsymbol{Z}_{\text{out}} - \boldsymbol{Z}_t\|_2^2 + \|\boldsymbol{Z}_{\text{in}} - \boldsymbol{Z}_{\text{out}}\|_2^2 \tag{5}$$

$$L_b = \|\boldsymbol{Z}_{\text{norm}} - \boldsymbol{Z}_t\|_2^2. \tag{6}$$

In (6), we omit the calculation of PVBand. This equation is used as the loss function in the high-resolution ILT when correcting via patterns. Because via layer patterns are smaller, PVBand does not change much, and SRAFs have already been generated by the low-resolution ILT. Additionally, corrections on via patterns are more sensitive to changes compared to metal layer shapes, where even slight variations can cause significant differences in the wafer image. Therefore, when correcting via patterns, the high-resolution ILT focuses on fine-tuning the main features to minimize the L_2 loss as much as possible. Experimental results show it will not significantly increase PVBand on via patterns. The ILT algorithm for metal layer patterns is the same as in [1]. In Algorithm 2, we show the ILT algorithm specifically designed for via layer patterns.

We then integrate low-resolution and high-resolution ILT in the proposed framework to achieve computational efficiency and improve mask manufacturability. High-resolution ILT maintains the same computational complexity as conventional

Algorithm 1 Two-Level Adaptive ILT

Input: iteration numbers in main correction $\{n_1, n_2\}$; iteration number of low-resolution ILT in continual correction n_3; maximum number of epochs in the continual correction phase N_1.

1: low-resolution ILT(iters $= n_1, \ldots$); ▷ Main correction
2: high-resolution ILT(iters $= n_2, \ldots$);
3: **if** result_check() == *False* **then** ▷ Continual correction
4: **repeat**
5: low-resolution ILT(iters $= n_3, \ldots$);
6: high-resolution ILT(iters $= n_2, \ldots$);
7: **if** result_check() == *True* **then**
8: *break*;
9: **else**
10: compare_and_save_solution();
11: **end if**
12: $i = i + 1$;
13: **until** $i \geq N_1$;
14: **end if**

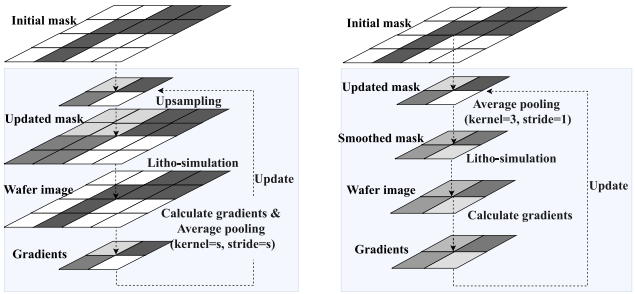


Fig. 3. Illustration of (a) high-resolution ILT and (b) low-resolution ILT.

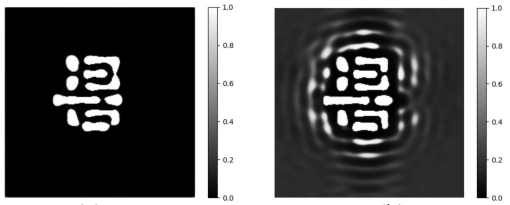
Algorithm 2 Multi-ILT on via Patterns

1: **Input:** Target image $\boldsymbol{Z}_t \in \mathbb{R}^{N\times N}$; scale factor s; learning rate lr; Optimizer Optim$(\cdot)$; The maximum number of consecutive iterations allowed in high-resolution ILT without achieving a better solution N_t;
2: $\boldsymbol{Z}_{t,s} \leftarrow$ AvgPool$(\boldsymbol{Z}_t$, kernel_size $= s$, stride $= s)$;
3: $\boldsymbol{M'}_s \leftarrow \boldsymbol{Z}_{t,s}$;
4: $tol = 0$;
5: **repeat**
6: $\quad \boldsymbol{M}_s = f_{\text{binary}}(\boldsymbol{M'}_s)$;
7: $\quad$ **if** $flag == 1$ **then** ▷ high-resolution ILT
8: $\quad\quad \boldsymbol{M}(x, y) = Upsample(\boldsymbol{M}_s)$;
9: $\quad\quad \boldsymbol{Z}_{\text{norm}}(x, y) =$ Litho$(\boldsymbol{M})$;
10: $\quad\quad \boldsymbol{Z}_{norm,s} =$ AvgPool$(\boldsymbol{Z}$, kernel_size $= s$, stride $= s)$;
11: $\quad\quad Loss = \left\|\boldsymbol{Z}_{norm,s} - \boldsymbol{Z}_{t,s}\right\|_2^2$;
12: $\quad\quad$ result_check$(\boldsymbol{M'}_s, \boldsymbol{Z}_t)$;
13: $\quad\quad$ compare_and_save_solution();
14: $\quad$ **else** ▷ low-resolution ILT
15: $\quad\quad \boldsymbol{M}_s =$ AvgPool$(\boldsymbol{M}_s$, kernel_size $= 3$, stride $= 1)$;
16: $\quad\quad \boldsymbol{Z}_{out,s}(x, y) =$ Litho$(\boldsymbol{M}_s)$;
17: $\quad\quad \boldsymbol{Z}_{in,s}(x, y) =$ Litho$(\boldsymbol{M}_s)$;
18: $\quad\quad Loss = \left\|\boldsymbol{Z}_{out,s} - \boldsymbol{Z}_{t,s}\right\|_2^2 + \left\|\boldsymbol{Z}_{in,s} - \boldsymbol{Z}_{out,s}\right\|_2^2$;
19: $\quad$ **end if**
20: $\quad \boldsymbol{G}_s =$ Optim$\left(\frac{\partial Loss}{\partial \boldsymbol{M'}_s}\right)$;
21: $\quad \boldsymbol{M'}_s = \boldsymbol{M'}_s - lr \cdot \boldsymbol{G}_s$;
22: **until** exit conditions are met.

ILT but generates simpler mask patterns. It performs average pooling to merge pixels and simplify the mask patterns. The downscaled mask and wafer images are then fed into the loss function (lines 10 in Algorithm 2), and gradients are computed to update the mask image. $\boldsymbol{M'}_s$ and $\boldsymbol{Z}_s$ represent the reduced mask and wafer images, respectively, with a scale factor s. Afterward, the mask $\boldsymbol{M}_s$ is restored to its original size to perform an accurate lithography simulation (line 7 in Algorithm 2). The computation flow of high-resolution ILT is illustrated in Fig. 3(a).

Since ILT employs nonlinear functions to map binary values to floating-point values for gradient calculation, the loss value does not directly indicate mask quality, as demonstrated in Fig. 1(a). This discrepancy is especially significant for via layer patterns. To address this issue, we incorporate an additional result verification step in the high-resolution ILT specifically for via patterns, despite adding one more lithography simulation (line 11–12 in Algorithm 2). This step is crucial since the output of high-resolution ILT serves as the final correction result.

For the low-resolution ILT, lithography simulations are performed every s pixel in both X and Y directions. It reduces the dimensionality of the simulation, significantly saving time while accepting a small loss in accuracy. Both the simulation and optimization processes are performed at this reduced size (lines 15, 16 and 20 in Algorithm 2), eliminating the need for upsampling and downsampling steps in low-resolution ILT. Litho$(\cdot)$ denotes the lithography simulation functions. The low-resolution simulation is outlined in (7), where $\mathcal{F}_N$ represents the FFT with a transform dimension of N



(a) (b)

Fig. 4. Binarized masks $\boldsymbol{M}$ using different T_R. The L_2 loss and PVBand of the mask obtained from (a) are 50626 and 51465. The L_2 loss and PVBand of the mask obtained from (b) are 43452 and 46361.

$$\begin{aligned}\boldsymbol{I}(sx, sy) &= \sum_{k=1}^{N_k} w_k \cdot \boldsymbol{I}_k \\ &= \sum_{k=1}^{N_k} w_k \left\| \mathcal{F}_N^{-1}(\boldsymbol{H}_k \odot \mathcal{F}_N(\boldsymbol{M}))(sx, sy) \right\|^2. \end{aligned} \tag{7}$$

In (7), much of the computation is wasted in the scaled-down aerial image. According to the mathematical theorem, reducing the spatial domain is equivalent to expanding the frequency domain. Therefore, to avoid redundant calculation, $\boldsymbol{I}_k$ in (7) can be computed using (8).

It is important to note that the transform dimension for the inverse FFT in (8) is $s\times$ smaller than in (7). Using (8) to obtain the aerial images significantly enhances computational efficiency, as the majority of the computation time is spent on the N_k inverse FFT operations

$$\begin{aligned}\boldsymbol{I}_k(sx, sy) &= \mathcal{F}_N^{-1}\left(\frac{1}{s^2}(\boldsymbol{H}_k \odot \mathcal{F}_N(\boldsymbol{M}))\left(\frac{x}{s}, \frac{y}{s}\right)\right) \\ &= \mathcal{F}_{N/s}^{-1}\left(\frac{1}{s^2}(\boldsymbol{H}_k \odot \mathcal{F}_N(\boldsymbol{M}))\right). \end{aligned} \tag{8}$$

In our implementation, we use (9) to approximate (8), and experimental results confirm the validity of this approximation. The runtime for 200 forward lithography simulations using (3), (8), and (9) are 8.173, 0.767, and 0.466 s, respectively. The high- and low-resolution ILT processes are illustrated in Fig. 3. In Algorithm 2, the binary function f_{binary} (line 5) and the smoothing pooling (line 14) will be detailed in Sections III-C and III-D respectively

$$\boldsymbol{I}_k \approx \mathcal{F}_{N/s}^{-1}\left(\boldsymbol{H}_k \odot \mathcal{F}_{N/s}(\boldsymbol{M}_s)\right). \tag{9}$$

C. Binary Function

As discussed in Section II-A, it is essential to constrain the pixel values of the mask between 0 and 1. Various binary functions $f_{\text{binary}}(\cdot)$ have been studied. For example, [19] uses the cosine transformation as its binary function, as shown in

$$\boldsymbol{M} = \frac{1 + \cos \boldsymbol{M'}}{2} \tag{10}$$

where $\boldsymbol{M'}$ represents the mask to be optimized and $\boldsymbol{M}$ denotes the incomplete binarized mask.

Due to the periodicity of (10), the initial learning rate for ILT using (10) should be chosen carefully, otherwise the optimization will have difficulty converging. Therefore, [20]

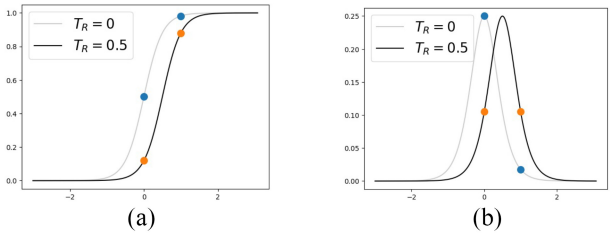


Fig. 5. In (a), the blue and orange dots represent the values in $\boldsymbol{M}$ in the first ILT iteration. They both use the sigmoid transformation but with $T_R = 0$ and $T_R = 0.5$, respectively. b) Is their corresponding gradients.

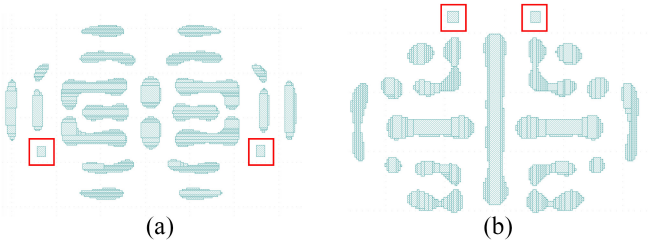


Fig. 6. Visualization of mask fracturing results for the obtained masks in case2 and case4 of the ICCAD 2013 benchmarks. The shapes within the red boxes are replaced by rectangular bounding boxes during the post-processing step to enhance manufacturability. (a) CASE 2. (b) CASE 4.

propose a monotonic sigmoid transformation as the binary function, as shown in

$$\boldsymbol{M} = \frac{1}{1 + \exp\left(-\beta \times \left(\boldsymbol{M}' - T_R\right)\right)} \tag{11}$$

where β and T_R are hyberparameters. Most pixel-based ILTs use the sigmoid transformation as its binary function and take $\beta = 4$ and $T_R = 0$.

In the initial iteration of ILT, the values in the binarized mask $\boldsymbol{M}$ are $\{0.5, \approx 1\}$, while the values in the original mask are $\{0, 1\}$. Due to this discrepancy, ILT assigns a large negative value to the opaque areas of the optimized mask $\boldsymbol{M}'$ after the first iteration. This makes the generation of SRAFs on the opaque areas challenging because the sigmoid function tends to fall into local optima. As shown in Fig. 4(a), the binarized mask $\boldsymbol{M}$ with $T_R = 0$ after 40 ILT iterations results in areas outside the main features being almost completely opaque with no SRAFs. Although SRAFs may eventually appear after more iterations, the process is very slow.

Therefore, we set $T_R = 0.5$, making the initial binarized mask $\boldsymbol{M}$ values $\{\approx 0.1, \approx 0.9\}$, which reduces the difference from the original mask and facilitates the generation of SRAFs. In Fig. 5, we plot the sigmoid transformation and its gradient. In Fig. 5(a), the initial values of the binarized mask M using $T_R = 0$ and $T_R = 0.5$ are represented in blue and orange, respectively. In the first iteration, for the sigmoid transformation using $T_R = 0$, $M(x, y)$ corresponding to $M'(x, y) = 0$ not only deviates significantly from the initial mask but also has the largest gradient value.

Fig. 4(b) illustrates the binarized mask using $T_R = 0.5$ with the same settings as Fig. 4(a). The results indicate that this improved binary function facilitates the generation of SRAFs more effectively. Due to downsampling and pooling operations, the SRAFs we generate are simpler in shape, as demonstrated in Fig. 6. These simpler SRAFs enhance mask printability without compromising manufacturability.

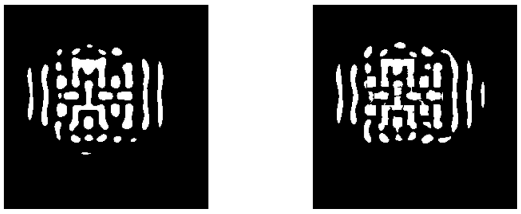


Fig. 7. L_2 loss and PVBand of mask (a) is 70308 and 69069. The L_2 loss and PVBand of mask (b) is 69043 and 70762.

Since the values in the intermediate mask $\boldsymbol{M} \in (0, 1)$ are continuous, a final binarization step is required to produce the binary mask $\boldsymbol{M}_{\text{out}}$. Equation (12) defines this final binarization process, which is commonly used in many ILT methods [5], [8], [9], [21]

$$\boldsymbol{M}_{\text{out}} = \begin{cases} 1, \boldsymbol{M} \geq t_m \\ 0, \boldsymbol{M} < t_m \end{cases} \tag{12}$$

where t_m is the threshold value and set to 0.5.

The binary function (11) is applied during ILT iterations, generating a mask with continuous pixel values in the range $(0, 1)$. While the binarization step (12) is performed after the ILT iterations to convert the intermediate mask into discrete binary values (0 or 1), producing the final mask.

In Fig. 4(b), the pixels corresponding to the main features are very close to 1, which means they remain largely unchanged in subsequent iterations. Our observations indicate that adjustments to SRAFs dominate the later iterations of ILT. However, due to the small gradients at each step, the number of SRAFs appearing in the final mask $\boldsymbol{M}_{\text{out}}$ is limited.

To generate more SRAFs, we use a smaller T_R for the final output. Specifically, we use $T_R = 0.5$ during optimization and $T_R = 0.4$ for the final binarization. The L_2 loss and PVBand results presented in Fig. 4 are based on these settings.

D. Simplify Shapes via Average Pooling

Numerous studies have shown that there are many benefits to adopting curvilinear mask shapes, which are superior to rectilinear mask shapes in many measures of mask quality, especially in terms of process variation band (PVBand) and DoF. Reference [22] shows that smoothly changing the shapes of contours can improve the PVBand of the optimized mask, which also applies to ILT.

In this section, we introduce an additional pooling operation designed to smooth shape contours in the low-resolution ILT. This average pooling is performed on $n \times n$ adjacent pixels, but with a stride size that is not equal to n. For instance, with $n = 3$ and a stride size of 1, the input and output sizes remain the same. This differs from the average pooling used in high-resolution ILT described in Section III-B, which is specifically intended for downsampling.

TABLE I
EFFECT OF LEARNING RATE ON L2 LOSS FOR LOW-RESOLUTION ILT

learning rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Metal	72558	63830	61254	60240	62560	60320	62557
Via	2044	4131	2752	4571	2978	5096	4491

By incorporating this operation, each mask pixel receives averaged attention from its neighboring pixels, allowing for more coherent adjustments based on their values. This approach ensures that changes to each pixel are more gradual and continuous, enhancing the overall smoothness of the mask. In our method, where we introduce additional SRAFs into the optimized mask, this smoothing operation becomes crucial for simplifying the mask's graphics.

In the low-resolution ILT process, we apply this pooling operation (line 15 of Algorithm 2) after the binarization step (line 6 of Algorithm 2), as illustrated in Fig. 3(b). Experimental results demonstrate that this pooling effectively prevents holes and fractures in the resulting masks.

Fig. 7(a) shows the mask image with pooling, while Fig. 7(b) depicts the mask without pooling. Although the mask without pooling exhibits a smaller squared L_2 loss, it also presents greater shape complexity. For optional post-processing, we refine the final mask images by handling excessively small and medium-sized shapes. For both metal and via layers, shapes with a pixel area smaller than 1000 are removed. Additionally, for metal layers, medium-sized irregular SRAFs with a pixel area between 1000 and 5000 are simplified by replacing them with their corresponding rectangular bounding boxes. The post-processing step further reduce the shot count, as shown in Fig. 6.

Additionally, the post-processing step does not significantly degrade the quality of the correction results. For Fig. 6(a), before post-processing, the L_2 loss and PVBand are 27 231 and 37 388, respectively. After post-processing, they are 27 191 and 37 414. For Fig. 6(b), before post-processing, the L_2 loss and PVBand are 9130 and 22 383, respectively. After post-processing, they are 9208 and 22 386.

IV. EXTENDED ALGORITHMS

The selection of hyperparameters plays a crucial role in determining ILT's performance. In the previous version, hyperparameters were chosen manually. In this article, we implement BO to automatically determine these values, thereby enhancing the method's robustness and adaptability. We also find that replacing the SGD optimizer with the Adam optimizer as the ILT engine significantly boosts correction efficiency, particularly for via layers. In the previous work, contour smoothing on via layer patterns sometimes causes the disappearance of design features. To counter this, we introduce a "positive region" to preserve these features. Furthermore, when using the Adam optimizer, we restrict the optimization region to improve the manufacturability of the resulting mask.

A. Hyperparameter Selection

The choice of hyperparameters plays a crucial role in determining the correction effectiveness of ILT. In Table I, we demonstrate how learning rates impact ILT performance, as measured by the L_2 loss. Suitable learning rates vary across different layout types, and an inappropriate learning rate can significantly degrade performance by increasing the L_2 loss. In the table, red boxes highlight poor learning rates, while green boxes indicate acceptable ones. The first row, labeled "Metal," corresponds to case2 from the ICCAD 2013 benchmark [17], while the second row, labeled "Via," represents an example from the via dataset.

To optimize hyperparameters effectively, we employ BO [23]. For smaller layouts, BO can be used to find the optimal solution achievable by ILT. For larger layouts, which are divided into multiple overlapping tiles, a subset of representative tiles is selected for BO. This approach ensures that the hyperparameters are well-suited for consistent correction across all tiles, balancing efficiency and accuracy.

Below, we explain our method for selecting representative patterns to avoid homogeneity. First, we use domain knowledge to manually design the following features to represent mask patterns.

1) *mean_area:* average area of the shapes in the pattern;
2) *min_area:* minimum area of the shapes in the pattern;
3) *thin_area:* area of the shapes with small width;
4) *num_shapes:* number of shapes in the pattern;
5) *pixel_density:* pixel density of the pattern.

After obtaining the feature vectors, we apply KMeans clustering. The pattern corresponding to the feature vector closest to each cluster center is selected as a representative pattern. For the selected tiles, BO generates 3 hyperparameter points per iteration and runs for a total of 5 iterations. In each iteration, the scores of all selected tiles are aggregated, and the hyperparameters that yield the best score are applied to the remaining tiles.

This hyperparameter selection strategy allowing our ILT algorithm to adapt to various datasets and enhance its robustness. Experimental results demonstrate that an appropriate learning rate not only reduces the number of correction failures but also improves correction efficiency. By decreasing the number of patterns entering the continual correction phase, it ultimately reduces the total runtime of ILT on the dataset.

B. Continual Correction

For large layouts, the chosen iteration counts are well-suited for most tiles and are used during the main correction phase (lines 1 and 2 in Algorithm 1). However, some tiles may still fail to meet the correction requirements after this phase. To address this, we introduce a continual correction phase (lines 3–13 in Algorithm 1) to further refine these tiles.

In the continual correction phase, the low-resolution ILT is applied with a single scale factor and significantly fewer iterations than in the main correction stage. The primary purpose of the low-resolution ILT in this phase is to provide a new starting point for high-resolution ILT, allowing it to escape local optima. Additionally, we set a maximum number of attempt epochs N_1. If none of the attempts produce satisfactory correction results, we retain the best result obtained (line 10

Fig. 8. L_2 loss and PVBand of mask (a) is 37335 and 48581. The L_2 loss and PVBand of mask (b) is 34351 and 50785.

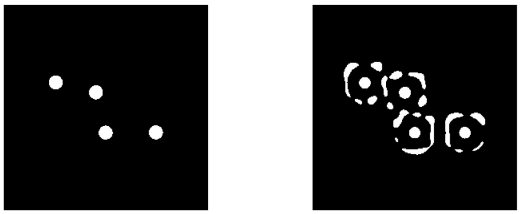


Fig. 9. L_2 loss and PVBand of mask (a) is 1921 and 11169. The L_2 loss and PVBand of mask (b) is 2044 and 7631.

TABLE II
COMPARISON OF SGD AND ADAM OPTIMIZER

	SGD			Adam		
	L2	PVB	EPE	L2	PVB	EPE
ICCAD 2013	26488.6	**41238.7**	**2.2**	**25533.5**	42446.8	2.4
Via dataset	4523.2	10183.8	1.6	**2654.6**	**8853**	**0**

in Algorithm 1). This adaptive approach ensures that all tiles achieve the required correction quality.

C. Optimizers

The ILT process is similar to training a neural network, where the pixel values on the layout can be seen as the weights in a neural network. Therefore, we can use optimizers from machine learning to calculate the update values in ILT. In Table II, we compare the performance of SGD and Adam [24] optimizer on ICCAD 2013 benchmark and a Via dataset. Both datasets contain ten masks, and the numbers in the table are the average metrics on these masks. The "Score" is the sum of L2 and PVB, and L2, PVB and EPE are introduced in Section II. For comparing the performance of ILT using SGD and Adam optimizers, we employed the method outlined in Section IV-A to select the suitable hyperparameters for each optimizer to achieve optimal performance.

Based on the results presented in Table II, Adam outperforms the SGD optimizer in both datasets. However, a major drawback of ILT using the Adam optimizer is that the output masks tend to be more complex, posing challenges to mask manufacturability. The correction results of ILT on the metal layer using Adam and SGD optimizers are compared in Fig. 8. Therefore, it is necessary to define correction regions to avoid generating too many SRAFs. The via patterns exhibit relatively regular shapes, making it easier to specify the correction regions, and thus more suitable for using the Adam optimizer. Besides, ILT with the SGD optimizer exhibits inferior performance on via patterns, as shown in Fig. 9. In some cases, it fails to generate sufficient SRAFs to enhance the PVBand, resulting in a significantly larger PVBand compared to the results obtained with the Adam optimizer. Although ILT with the SGD optimizer can eventually achieve good correction results with more iterations [1], it comes at the cost of longer runtime.

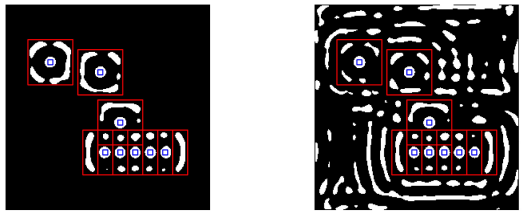


Fig. 10. Red boxed areas represent the "optimization region," and the blue boxed areas represent the positive region. Restricting the optimization region when using the Adam optimizer can significantly simplify the mask image, while defining the positive region with the SGD optimizer can prevent the disappearance of target shapes. (a) With opt. region. (b) W/o opt. region.

TABLE III
COMPARISON OF ILT RESULTS WITH DIFFERENT CORRECTION RADII

	r=300				r=200			
	L2	PVB	L2+PVB	#shots	L2	PVB	L2+PVB	#shots
Average	5255	**12874.1**	**18129.5**	457.7	**4973**	13391.8	18364.8	**330.4**
Ratio	1.057	**0.961**	**0.987**	1.385	**1**	1	1	**1**

TABLE IV
EFFECT OF OPTIMIZATION REGION

opt. region	pos. region	L2	PVB	EPE
✗	✗	3619	7928.1	0
✗	✓	4899.8	**7451.7**	0.5
✓	✗	**2524.1**	8716.7	0
✓	✓	2662.2	8857.8	0

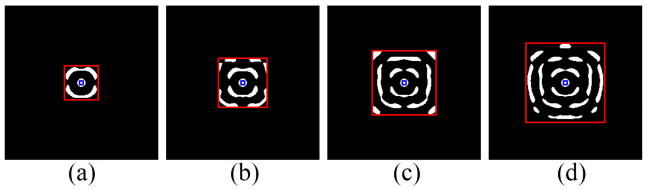


Fig. 11. Results for the same layout with different optimization region radii r. (a)–(d) Correspond to the following values: L_2 losses of 530, 522, 781, and 779, and PVBand values of 1870, 1696, 1577, and 1575, respectively. While $r = 300$ yields a slightly lower combined L2 and PVBand value, $r = 200$ leads to a much simpler correction outcome.

D. Optimization Region

As discussed in Section IV-C, assigning an optimization region to ILT is crucial for managing the complexity of mask patterns, and helps to improve the manufacturability of the masks, as shown in Fig. 10. Additionally, specifying the optimization region can further enhance the correction ability of ILT [15], particularly for via patterns.

For via patterns, we define the optimization region as extending 200 nm outward from the edges of the shapes. We conduct comparative experiments on the same layout by applying ILT with optimization region radii r of 200, 300,

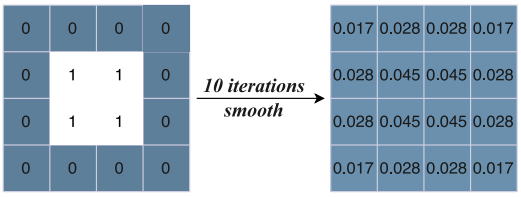


Fig. 12. Performing just 10 smoothing operations on the original pattern can still result in the loss of target features on via patterns. This explains why using the SGD optimizer may cause the disappearance of the target shapes.

400, and 500 nm. For a single via pattern, their ILT results are shown in Fig. 11. For patterns with multiple interacting shapes, we test the method on 10 cases, and the results are presented in Table III. The experimental results shown that as the optimization region expands, PVBand decreases, the complexity of masks (#*shots*) increases, and L2 does not necessarily decrease. Considering L2, PVBand, and #*shots* together, we choose $r = 200$ as the correction radius for via patterns.

In the first and third rows of Table IV, we compare the optimization results on the via dataset with and without specifying the optimization region. We tested ILT with the Adam optimizer on ten via patterns, and the data in Table IV is the average result. It shows that assigning an optimization region can greatly improve the printability of the output masks.

On the via dataset, when using the SGD optimizer along with the smoothing operation described in Section III-D, it is crucial to keep the original target pixels unchanged. This area of unchanged pixels is called the positive region, which is the area within the blue line in Fig. 10.

Since the gradient values updated by SGD are relatively small on via patterns, applying the SGD optimizer along with smoothing operations can lead to the disappearance of target features, as illustrated in Fig. 12. Without assigning positive regions, ILT using SGD fails to deliver good correction results on all ten testing via patterns. However, when using the Adam optimizer, specifying the positive region is not essential. It sometimes yields better results, but not always.

In conclusion, if SGD is used as the ILT optimizer, it is recommended to assign positive regions. However, if Adam is used as the ILT optimizer, specifying positive regions is unnecessary. When it comes to the optimization region, using it consistently enhances ILT capability and improves the manufacturability of the resulting masks, regardless of the optimizer employed.

V. Experimental Results

The adaptive ILT is developed on Pytorch. All experiments are performed on a Linux workstation with a 2.6GHz Intel Xeon CPU and a single Nvidia GeForce RTX 3090 GPU. The litho-simulation tool is obtained from ICCAD 2013 competition [17] and we reimplemented it using *torch.fft* to take advantage of GPU. For simplicity, we use "L2," "PVB," "EPE," "TAT," and "#shots" to denote squared L_2 loss, PVB, EPE, turnaround time and mask fracturing shots count, respectively. Experimental results demonstrate the efficiency and effectiveness of our method. The proposed method is validated on designs that cropped from M1 layer and via layer.

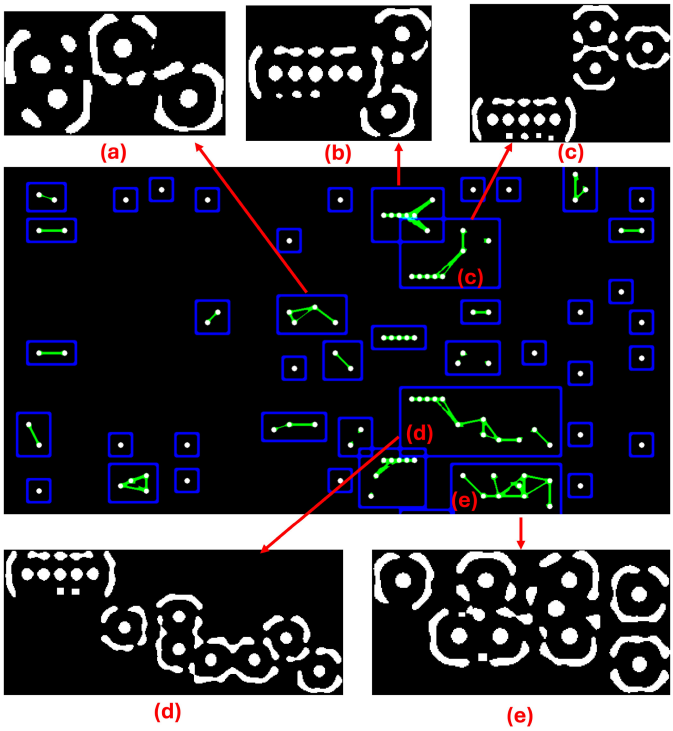


Fig. 13. We extract unique patterns from the layout by treating each via as a node in a graph, connecting vias within a distance of *hB* with an edge. We then identify and extract the unique connected components. The blue boxes highlight these components, many of which appear in multiple locations on the layout. Snippets (a)–(e) shows the corresponding ILT results for these connected components.

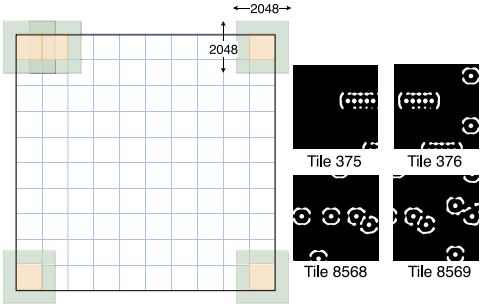


Fig. 14. Large layout is divided into smaller overlapped tiles.

A. Results on Full Chip

In this experiment, we evaluate the proposed method on a full-chip via layer layout with a size of 146 μm $\times$ 148 μm, corresponding to layer 40 of `ispd19_test1` [25]. Since the via shapes are all 50 nm $\times$ 50 nm squares with varying placement, we can model the layout as a graph and extract the unique placement relationships. This approach significantly reduces stitching issues when correcting the via layer layout. However, this method cannot be applied to the full-chip metal layer layout, and we plan to explore this in future research.

We model the full-chip via layout as a graph, where each via is represented as a node. An edge is added between two

vias if their distance is within hB. Given that the light source wavelength is 193 nm, we set $hB = 500$. Next, we identify all connected components in the graph and extract 1895 unique ones. The blue boxes in Fig. 13 highlight these connected components. Each connected component is expanded with a margin to form a sublayout for ILT. Among these sublayouts, 1495 have dimensions within $2\,\mu\text{m} \times 2\,\mu\text{m}$, and can be processed directly by ILT. These tiles exhibit varying shape densities, with some containing only one or two vias, while others contain up to 13 vias. We compile these 1495 sublayouts into a large-scale dataset to validate the effectiveness of our method.

First, we establish a correction criterion to determine whether a tile has been successfully corrected. Since the L_2 loss varies with the total area of the designed shapes, it is not practical to set a fixed threshold for the L_2 loss to assess correction quality.

Definition 5 (L_2 Ratio): We define the L_2 ratio as the L_2 loss normalized by the total area of the target image, expressed as $(L_2/\|\boldsymbol{Z}_t\|_2^2)$.

L_2 Ratio provides a more effective metric for evaluating the quality of layout optimization. In this experiment, we set L_2 Ratio ≤ 0.22 as the correction criterion.

The default ILT setting consists of 80 iterations of low-resolution ILT with a scale factor of 4, followed by 10 iterations of high-resolution ILT with a scale factor of 8. The performance of ILT with the default setting is shown in the "default" row of Table V. The results show that, while some tiles are successfully corrected, 641 tiles still have EPE violations, and 1469 tiles fail to meet the correction criterion after the ILT process. In Table V, the "Average L2" and "Average PVB" represent the average values of "L2" and "PVB" across all 1495 tiles, while "TAT" refers to the total turnaround time.

By incorporating only the continual correction phase into the default setting, the number of tiles with EPE violations is significantly reduced from 641 to 3, as shown in Table V. Additionally, the average L_2 loss decreases by more than 1.69 times. However, this improvement comes with tradeoffs: the total turnaround time (TAT) increases by 3.4 times compared to the default setting, and 1350 tiles still fail to meet the correction criterion.

With hyperparameter selection alone, the number of tiles with EPE violations is reduced to 54, with only a slight increase in runtime compared to the default setting. Additionally, when both continual correction and hyperparameter selection are applied, L2 and EPE are further reduced.

Finally, by incorporating all proposed techniques, we successfully eliminated all EPE violations across the tiles, as shown in the last row of Table V. The total runtime includes 7982.844 s for ILT and 187.29 s for hyperparameter selection. Compared to the default setting, this approach increases the runtime by 0.55 times while using only 46% of the runtime required when combining the default setting with the continual correction phase. Furthermore, it achieves the smallest average L_2 loss, indicating superior correction performance.

For the selected tiles, BO generates 3 hyperparameter points per iteration and runs for a total of 5 iterations. In each iteration, the scores of all selected tiles are aggregated, and the hyperparameters that yield the best score are applied to the remaining tiles.

For via shapes, due to their small size, they may fail to develop adequately on the wafer, resulting in the smallest PVB. The default setting achieves the smallest PVB, which is attributed to its poor correction performance. Upon examining the wafer images, it is evident that via shapes on many tiles are poorly developed or fail to appear altogether.

After processing the 1495 connected components, there remain 400 that cannot be handled directly by ILT, as their dimensions exceed $2\,\mu\text{m} \times 2\,\mu\text{m}$. The ILT results for these sublayouts may experience stitching issues. In this article, we apply several engineering tricks to mitigate these problems, though they are not fully eliminated.

After performing ILT on each connected component, we place the corrected mask patterns back into the original full-chip layout. Note that many connected components appear in multiple locations on the full chip. Once all connected components have been processed, the entire chip layout is effectively corrected. We then divide the layout into multiple 2048×2048 nm^2 tiles, as shown in Fig. 14, run optical simulations, and measure the correction results. This partitioning yields 20 023 tiles in total, of which 11 092 contain shapes in the tile core region.

We performed optical simulations on these 11 092 tiles, considering only the measurement results from each tile's core region. This approach provides accurate full-chip measurements, as shown in the second row of Table VI. The first row, labeled "Estimate," is derived by summing the measurement result of each connected component, multiplied by its number of occurrences in the layout. Note that the 13 EPEs in the first row appear in the 400 larger connected components. Experimental results show that the accurate measurements are slightly worse than the estimates and introduce one additional EPE. Considering that there are 11 092 tiles, this demonstrates that our method for extracting unique patterns from the via layer and correcting them is effective. This not only significantly reduces the correction time but also mitigates stitching issues.

TABLE VI
ESTIMATE AND ACCURATE FULL-CHIP ILT RESULTS

	L2	PVB	EPE
Estimate	10,389,051	38,821,910	13
Accurate	11,056,324	40,824,465	14

B. Results on M1 Designs

In Tables VII and VIII, we compare our approach with state-of-the-art methods on ten M1 designs of 32 nm technology nodes provided by ICCAD 2013 benchmarks [17]. The columns labeled "A2-ILT [15]," "L2O-ILT [12]," and "DevelSet [9]" show the results of selected baseline ILT methods. Column "Multi-ILT" is the previous version of our proposed method [1]. The "Ours" column represents the

TABLE V
COMPARISON ON ICCAD 2013 BENCHMARKS (PART I)

	Average L2 (nm^2)	Average PVB (nm^2)	EPE $\neq 0$	L_2 Ratio > 0.22	TAT (s)
default	6160.656	**6710.744**	641	1469	**5255.287**
default + BO	4317.704	7926.918	54	1476	6322.271
default + continual correction	3631.539	8130.874	3	1350	17848.203
default + continual correction + BO	2269.466	8177	0	65	9803.064
default + continual correction + BO + new_high	**2245.188**	8228.882	**0**	**40**	8170.134

*:*For via layer patterns, the PVBand is minimal when no features are printed on the wafer image. Therefore, the comparison of PVBand is only meaningful when the patterns are correctly developed, such as when EPE = 0.*

TABLE VII
COMPARISON ON ICCAD 2013 BENCHMARKS (PART II)

Benchmarks		A2-ILT [15]				L2O-ILT [12]				Multi-ILT [1]				Ours			
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots
case1	215344	45824	59136	7	242	39636	46905	3	-	38403	47226	3	380	38169	45578	3	420
case2	169280	33976	52054	3	211	29108	37099	0	-	28174	37561	0	285	27191	37414	0	259
case3	213504	94634	82661	62	282	67263	69115	21	-	65825	69384	21	326	65063	71667	20	340
case4	82560	20405	29435	2	103	10807	20694	1	-	10446	21723	0	241	9130	22383	0	258
case5	281958	37038	62068	1	319	31909	48797	0	-	28393	50180	0	390	27996	49837	0	439
case6	286234	40701	54842	2	244	31474	45453	0	-	30147	44267	0	403	30794	43942	0	428
case7	229149	21840	48474	0	206	16942	35942	0	-	14691	36688	0	370	14444	35809	0	394
case8	128544	14912	24598	0	156	12236	19496	0	-	11172	19735	0	278	10816	19889	0	279
case9	317581	47489	68056	2	248	34849	56706	0	-	34888	56108	0	482	35041	55027	0	483
case10	102400	9399	20243	0	126	7203	15976	0	-	7171	16611	0	164	7180	16797	0	150
Average		36621.8	50156.7	7.9	**213.7**	28142.7	**39618.3**	2.5	-	26931	39948.3	2.4	331.9	**26582.4**	39834.4	**2.3**	344.5
Ratio		1.378	1.259	3.435	0.620	1.059	0.995	1.087	-	1.013	1.003	1.043	0.963	1.000	1.000	1.000	1.000

TABLE VIII
COMPARISON ON ICCAD 2013 BENCHMARKS

Bench	DevelSet [9]				L2O-ILT [12]				Multi-ILT [1]				Ours			
	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots	L2 (nm^2)	PVB (nm^2)	EPE	#shots
case1	48687	57612	8	917	39742	50432	3	-	40223	50214	3	288	39049	45661	3	418
case2	34062	49647	1	731	31550	42620	0	-	34361	44157	2	183	27353	37691	0	239
case3	92025	76744	59	847	67612	73850	22	-	68433	69974	23	313	66507	68931	21	340
case4	18194	27318	2	333	12550	20306	1	-	10868	21595	0	233	9409	22137	0	254
case5	42158	57123	1	891	34056	50982	0	-	29566	52168	0	340	28096	49940	0	414
case6	41611	51096	2	754	31830	47237	0	-	30430	45629	0	370	30172	43924	0	430
case7	24265	43032	0	493	20443	37207	0	-	16988	41240	0	288	14892	35773	0	387
case9	15301	22131	0	436	13429	19702	0	-	11730	20614	0	239	11427	19762	0	283
case9	49843	62139	0	887	39652	58708	0	-	36369	59127	0	414	36315	55291	0	440
case10	10341	20104	0	398	8363	17561	0	-	9262	18621	0	111	6692	16620	0	153
Average	37648.7	46694.6	7.3	668.7	29922.7	41860.5	2.6	-	28823.0	42333.9	2.8	**277.9**	**26991.2**	**39573**	**2.4**	335.8
Ratio	1.395	1.180	3.042	1.991	1.109	1.058	1.083	-	1.068	1.070	1.167	0.828	1.000	1.000	1.000	1.000

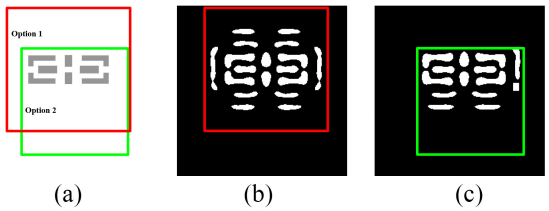


Fig. 15. (a) Two options for optimizing regions. (b) Mask obtained using option 1. (c) Mask obtained using option 2.

updated version of our method, which incorporates hyperparameter tuning tailored to each test case. Since the ICCAD 2013 benchmarks consist of only 10 test cases, our primary focus is on achieving the highest quality correction results for each individual case. To optimize performance, we use BO, generating 5 hyperparameter candidates per iteration and running for a total of 10 iterations. It takes approximately 200 s per case, which is longer than other methods. However, it produces the best correction results. In practice, when working with a small number of test layouts, ILT engineers typically manually fine-tune parameters to achieve optimal results. Our method automates this process, streamlining the workflow.

Both A2-ILT and DevelSet constrain mask optimization to a predefined region, but they each define this region differently. As shown in Fig. 15(a), A2-ILT uses option 1 while DevelSet uses option 2 as the optimizing region. For a fair comparison, we follow their respective settings in Tables VII and VIII.

In Table VII, our approach surpasses A2-ILT with a reduction of 37.8% in squared L_2 loss and a 25.9% reduction in PVBand. Despite having a higher "#shots," our mask shapes exhibit greater regularity compared to A2-ILT, thanks to the pixel merging in high-resolution ILT, the smoothing operation and the post-processing step. Compared to DevelSet, our method achieves a reduction of 39.5% in squared L_2 loss and a 18.0% reduction in PVBand, along with a 99.1% reduction in "#shots."

TABLE IX
COMPARISON OF RANDOM SEARCH AND BO

	Random			BO		
	L2	PVBand	EPE	L2	PVBand	EPE
trial 1	26438.7	40016.7	2.3	25980.4	40282.7	2.1
trial 2	26701	39416.7	2.4	26399.5	39723	2.3
trial 3	26642.9	39445.2	2.3	26404.5	39821.3	2.2
Average	26594.2	**39626.2**	2.333	**26261.5**	39942.3	**2.2**
Ratio	1.013	**0.992**	1.061	**1.000**	1.000	**1.000**

TABLE X
COMPARISON OF RESULTS WITH AND WITHOUT CONTINUAL CORRECTION

	Multi-ILT				With continual correction			
	L2	PVBand	EPE	TAT	L2	PVBand	EPE	TAT
case 1	38403	47226	3	**3.45**	**37033**	**46877**	3	11.364
case 3	65825	**69384**	21	**3.44**	**65111**	70509	**18**	10.983

Compared to L2O-ILT, our method results in a 0.5% increase in PVBand, but achieves a 5.9% reduction in squared L_2 loss, as shown in Table VII. In Table VIII, our method demonstrates a 10.9% reduction in squared L_2 loss and a 5.8% reduction in PVBand relative to L2O-ILT.

Compared to Multi-ILT [1], our method achieves reductions of 1.3% and 6.8% in L_2 loss, and 0.3% and 7% in PVBand, as shown in Tables VII and VIII, respectively. Furthermore, in both tables, our method achieves the fewest EPE violations, demonstrating its superior correction quality.

In Table IX, we compare the performance of BO and random search, both use the same ILT solver. The results are averaged over three independent experiments. We used the same dataset as in Table VII. For each case, both random search and BO sampled 50 points. The objective function for BO is as follows:

$$\text{Score} = L2 + \text{PVB} + \text{EPE} \times 100000. \tag{13}$$

Based on the experimental results, BO achieves better L2 and EPE performance compared to random search. Both random search and BO outperform the original ILT solver with the default settings, albeit at the cost of increased runtime. This demonstrates that optimizing hyperparameter selection for the ILT solver can further enhance its correction quality.

In Table X, we compare the experimental results of the original Multi-ILT with its version incorporating continual correction. Continual correction is applied only when the output lithographic image contains EPE. As shown in Table VII, only case 1 and case 3 exhibit EPE in the Multi-ILT correction results. Therefore, Table X presents results for case 1 and case 3 only. When selecting the final correction result, we prioritize solutions with lower EPE. If EPE remains the same, we then consider L2 and PVBand. For case 3, continual correction effectively reduces EPE, while for case 1, although EPE remains unchanged, both L2 and PVBand are improved.

C. Results on via Designs

The first dataset, referred to as "dataset-1" in Table XII, consists of ten via patterns, each of size 2048 × 2048, randomly selected and cropped from the ISPD 2019 large layout.

Fig. 16. Optimized mask output. (a) GLS-ILT [7]. (b) Adaptive-ILT.

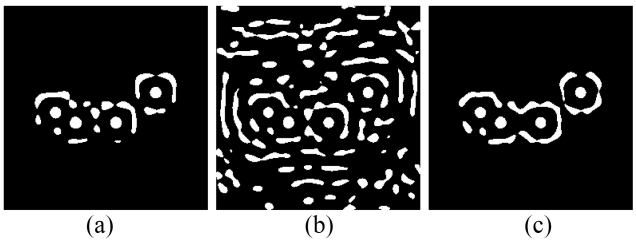


Fig. 17. Here are the output mask images corresponding to the ILT results from rows 1–3 in the ablation study shown in Table XI. The outputs from rows 4–5 are similar to row 3 and are not shown here. (a) Corresponds to row 1, (b) to row 2, and (c) to row 3 in Table XI. As seen in (b), without constraining the optimization region, directly using the Adam optimizer can lead to overly complex mask patterns, making manufacturing difficult.

The second via dataset, referred to as "dataset-2" in Table XII, is the same as the one used in [1] and contains fifteen via patterns from the dataset provided by [26]. Experiments reveal that dataset-1 is more challenging for Multi-ILT compared to dataset-2. Using the original Multi-ILT settings, the patterns in dataset-2 can be successfully transferred onto the wafer, but the majority of patterns in dataset-1 fail to be printed.

On these two datasets, we compare our approach with GLS-ILT [7], which is one of the state-of-art numerical ILT solvers. Our method incorporates the proposed improvements to Multi-ILT, including using the Adam optimizer, constraining the optimization region, and applying high-resolution ILT designed for via patterns. We perform 200 iterations of low-resolution ILT with a scale factor of 4 and 5 iterations of high-resolution ILT with a scale factor of 8. We adopt the hyperparameter selection from Section V-A. The results in Table XII confirm that these hyperparameters are also effective for this dataset. Since via shapes are relatively simple and small, the optimization space is limited. Therefore, unlike in the M1 experiment, we do not conduct separate hyperparameter tuning for each case.

In Table XII, our method outperforms GLS-ILT with an 8% reduction in squared L_2 loss and a 45% reduction in PVBand. This significant reduction in PVBand is primarily due to our method generating more SRAFs, as shown in Fig. 16. The smoothing operation mentioned in Section III-D also contributes to further reducing PVBand and regularizing the mask shapes. Additionally, our method achieves a 25% reduction in mask fracturing shot count (#shots) compared to GLS-ILT. This improvement is mainly due to the downsampling operation in the high-resolution ILT, which removes overly fine features from the resulting mask images. Furthermore, our method reduces runtime by 29% compared to GLS-ILT.

TABLE XI
ABLATION STUDY ON VIA PATTERNS

Adam	opt. region	new-high	continual correction	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)
-	-	-	-	4563.40	8485.60	0	**210.10**	11.80
✓	-	-	-	3860.40	**6903.60**	0	1099.00	3.80
✓	✓	-	-	3180.00	8349.70	0	304.40	3.88
✓	✓	✓	-	2301.40	8355.40	0	306.30	**3.61**
✓	✓	✓	✓	**2279.40**	8353.20	0	306.30	4.03

TABLE XII
COMPARISON ON VIA PATTERNS

	GLS-ILT [7]					Ours				
	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)
dataset-1	2383.20	12319.00	0.00	345.20	4.65	2301.40	8355.40	0.00	306.30	3.61
dataset-2	5413.29	14766.12	0.00	413.00	4.68	4909.59	10266.82	0.00	301.65	3.62
Average	3898.25	13542.56	0.00	379.10	4.67	**3605.49**	**9311.11**	0.00	**303.97**	**3.62**
Ratio	1.08	1.45	1.00	1.25	1.29	**1.00**	**1.00**	1.00	**1.00**	**1.00**

In Table XI, we conduct ablation experiments on the four newly proposed measures on dataset-1. The four measures are: using the Adam optimizer, specifying the optimization region, implementing the high-resolution ILT designed for via patterns, and including a continual correction phase. In the table, they are denoted as "Adam" "opt. region," "new-high," and "continual correction," respectively. If none of these four measures are applied, it corresponds to the original Multi-ILT, which is represented in the first row of the table.

The original Multi-ILT uses the SGD optimizer and converges slower than the Adam optimizer, as mentioned in Section IV-C. It first performs 100 iterations of low-resolution ILTs with a scale factor of 4 using learning rates of 2 and 1, respectively. Then, it performs 150 iterations of low-resolution ILTs with a scale factor of 2 and a learning rate of 2. Finally, it performs 10 high-resolution ILTs with a scale factor of 8 to produce the final result. Consequently, the original Multi-ILT requires a much longer runtime to deliver a satisfactory result.

The next four rows in Table XI all use the Adam optimizer, performing 200 iterations of low-resolution ILT and 5 iterations of high-resolution ILT. Using the Adam optimizer significantly reduces runtime and improves the quality of the optimized masks, as shown in Table XI. However, without constraining the optimization region, using the Adam optimizer for ILT directly generates an excessive number of SRAFs, resulting in overly complex layout patterns, as shown in Fig. 17(b). In Table XI, constraining the optimization region reduced the number of shots by 3.6× and also decreased the L_2 loss.

Furthermore, using the new high-resolution ILT reduces the L_2 loss by an additional 38.2%, as shown in row 4 of Table XI. If we set stricter correction requirements, we enter the continual correction phase to further optimize the results, albeit at the cost of longer runtime, as shown in the last row of Table XI.

D. Effect of Downsampling

To validate the effectiveness of the downsampling operation, we perform 100 iterations of low-resolution ILT, high-resolution ILT, and ILT without downsampling, respectively. Their comparison results on case1 of the ICCAD 2013 benchmarks [17] are listed in Table XIII. We set the scale factor for both low-resolution and high-resolution downsampling to 4, with a learning rate of 1, without constraining the optimization region.

TABLE XIII
RESULTS OF USING DIFFERENT DOWNSAMPLING METHODS ON CASE1 OF ICCAD 2013 BENCHMARKS

	L2 (nm^2)	PVB (nm^2)	TAT (s)	#Shots
w/o. downsample	35349	47667	21.086	3048
high-resolution	40525	45875	20.976	882
low-resolution	40562	45906	**1.134**	889

Although the mask produced by ILT without downsampling has the smallest L_2 loss, its complexity is unacceptable. The high-resolution ILT consumes about the same time as the ILT without downsampling, but its produced mask has much smaller "#shots." The low-resolution ILT achieves more than 18× speedup over high-resolution ILT, which proves its good computational efficiency.

VI. CONCLUSION

In this article, we propose an ILT method based on multilevel lithography simulation, enhancing its adaptability and robustness by incorporating hyperparameter optimization and a continual correction phase. For via layer patterns, we design a new high-resolution ILT and employ a optimization region, greatly improving the correction capabilities of ILT compared to the previous version. Experimental results show that the proposed method outperforms SOTA methods in mask printability and runtime performance for both via layer patterns and metal layer patterns. Furthermore, we conduct comparative experiments on the choice of optimizer and find that the Adam optimizer is more effective for via layer patterns. We also perform a full-chip experiment on via layer layout to showcase the scalability of the proposed method. This experiment also stimulate us to solve the stitching issues for metal layer layouts in future work.

REFERENCES

[1] S. Sun, F. Yang, B. Yu, L. Shang, and X. Zeng, "Efficient ILT via multi-level lithography simulation," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.

[2] L. Pang, "Inverse lithography technology: 30 years from concept to practical, full-chip reality," *J. Micro/Nanopatterning, Mater., Metrol.*, vol. 20, no. 3, 2021, Art. no. 30901.

[3] K. Hooker, B. Kuechler, A. Kazarian, G. Xiao, and K. Lucas, "ILT optimization of EUV masks for sub-7nm lithography," in *Proc. 33rd Eur. Mask Lithogr. Conf.*, 2017, pp. 9–20.

[4] R. Pearman et al., "How curvilinear mask patterning will enhance the EUV process window: A study using rigorous wafer+ mask dual simulation," in *Proc. 26th Symp. Photomask Next-Gener. Lithogr. Mask Technol.*, 2019, pp. 59–67.

[5] B. Jiang, L. Liu, Y. Ma, B. Yu, and E. F. Y. Young, "Neural-ILT 2.0: Migrating ILT to domain-specific and multitask-enabled neural network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2671–2684, Aug. 2022.

[6] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[7] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A GPU-enabled level-set method for mask optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 2, pp. 594–605, Feb. 2023.

[8] H. Yang, S. Li, 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, Sep. 2019.

[9] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "DevelSet: Deep neural level set for instant mask optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 12, pp. 5020–5033, Dec. 2023.

[10] W. Ciou et al., "Sub-resolution assist feature placement with generative adversarial network," *J. Micro/Nanopatterning, Mater., Metrol.*, vol. 20, no. 4, 2021, Art. no. 41209.

[11] H. Yang and H. Ren, "ILILT: Implicit learning of inverse lithography technologies," 2024, *arXiv:2405.03574*.

[12] B. Zhu et al., "L2O-ILT: Learning to optimize inverse lithography techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 3, pp. 944–955, Mar. 2024.

[13] H. Yang et al., "Generic lithography modeling with dual-band optics-inspired neural networks," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 973–978.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assist. Interv. (MICCAI)*, 2015, pp. 234–241.

[15] Q. Wang, B. Jiang, M. D. F. Wong, and E. F. Y. Young, "A2-ILT: GPU accelerated ILT with spatial attention mechanism," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 967–972.

[16] X. Ma, Z. Wang, Y. Li, G. R. Arce, L. Dong, and J. Garcia-Frias, "Fast optical proximity correction method based on nonlinear compressive sensing," *Opt. Express*, vol. 26, no. 11, pp. 14479–14498, 2018.

[17] 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.

[18] H. H. Hopkins, "The concept of partial coherence in optics," *Proc. Royal Soc. London. Ser. A, Math. Phys. Sci.*, vol. 208, no. 1093, pp. 263–277, 1951.

[19] A. Poonawala and P. Milanfar, "Mask design for optical microlithography—An inverse imaging problem," *IEEE Trans. Image Process.*, vol. 16, pp. 774–788, 2007.

[20] X. Zhao and C. Chu, "Line search-based inverse lithography technique for mask design," *VLSI Design*, vol. 2012, no. 1, 2012, Art. no. 589128.

[21] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.

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

[23] A. I. Cowen-Rivers et al., "An empirical study of assumptions in Bayesian optimisation," 2020, *arXiv:2012.03826*.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.

[25] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proc. Int. Symp. Phys. Design*, 2019, pp. 147–151.

[26] H. Geng et al., "Hotspot detection via attention-based deep layout metric learning," in *Proc. 39th ICCAD*, 2020, pp. 1–8.

Shuyuan Sun received the B.E. degree in microelectronics science and engineering from Xidian University, Xi'an, Shaanxi, China, in 2020. She is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Chips and System, Microelectronics Department, Fudan University, Shanghai, China.

Her current research interests include hotspot detection, and optical proximity correction.

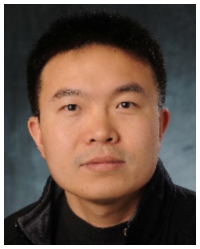
Fan Yang (Member, IEEE) received the B.S. degree from Xi'an Jiaotong University, Xi'an, China, in 2003, and the Ph.D. degree from Fudan University, Shanghai, China, in 2008.

He is currently a Full Professor with the Microelectronics Department, Fudan University. His research interests include model order reduction, circuit simulation, high-level synthesis, acceleration of artificial neural networks, yield analysis, and design for manufacturability.

Bei Yu (Senior 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 11 Best Paper Awards from the ICCAD 2024, 2021, and 2013, the IEEE TSM 2022, the DATE 2022, the ASPDAC 2021 and 2012, the ICTAI 2019, the Integration, the VLSI Journal in 2018, the ISPD 2017, the SPIE Advanced Lithography Conference 2016, the six ICCAD/ISPD contest awards, the IEEE CEDA Ernest S. Kuh Early Career Award in 2021, the DAC Under-40 Innovator Award in 2024, and the Hong Kong RGC Research Fellowship Scheme (RFS) Award in 2024. He has served as TPC Chair for ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.



Li Shang (Member, IEEE) received the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 2004.

He was the Deputy Director and the Chief Architect with Intel Labs China, Shanghai, China, and an Associate Professor of University of Colorado Boulder, Boulder, CO, USA. He is a Professor with the School of Computer Science, Fudan University, Shanghai, China. His research interests include computer systems, human-centered computing, machine learning, VLSI and EDA, with over 100 publications, multiple best paper awards and nominations.

Prof. Shang was a recipient of the NSF Career Award.

Xuan Zeng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Fudan University, Shanghai, China, in 1991 and 1997, respectively.

She is a Full Professor with the Department of Microelectronics, Fudan University. She was the Director with the State Key Laboratory of Integrated Chips & System, from 2008 to 2012. Her current research interests include analog circuit modeling and synthesis, design for manufacturability, high-speed interconnect analysis and optimization, and circuit simulation.