L2O-ILT: Learning to Optimize Inverse Lithography Techniques

Binwu Zhu, Su Zheng, Ziyang Yu, Guojin Chen, Yuzhe Ma, *Member, IEEE*, Fan Yang, *Member, IEEE*, Bei Yu, *Senior Member, IEEE*, and Martin D. F. Wong

***Abstract*—Inverse lithography technique (ILT) is one of the most widely used resolution enhancement techniques (RETs) to compensate for the diffraction effect in the lithography process. However, ILT suffers from runtime overhead issues with the shrinking size of technology nodes. In this article, our proposed L2O-ILT framework unrolls the iterative ILT optimization algorithm into a learnable neural network with high interpretability, which can generate a high-quality initial mask for fast refinement. Experimental results demonstrate that our method achieves better performance on both mask printability and runtime than the previous methods.**

***Index Terms*—Design for manufacture, mask optimization, learning to optimize.**

I. Introduction

WITH the continuous scaling-down of technology nodes, the proximity effect and optical diffraction are becoming non-neglectable, which seriously affects the yield of integrated circuits. Resolution enhancement techniques (RETs) are developed to reduce printing errors during the lithography process. Optical proximity correction (OPC) is one of the widely used RETs to compensate for lithography proximity effects by correcting mask pattern shapes and inserting assist features.

Typical OPC methodologies include model-based approaches [1], [2], [3] and inverse lithography technology (ILT)-based methods [4], [5], [6], [7], [8], [9]. For model-based OPC, the edges of polygons in the mask are first divided into segments, and these edges are moved under the guidance of the lithography simulation model. ILT-based methods represent the mask as a pixel-wise function [4], [5], [6], [7], [10] or level-set function [8], [9], [11], [12]. Then, the OPC process is modeled as an inverse problem, which can be effectively solved by optimizing the misfit between the printed image on the wafer and the target pattern. Compared with model-based methods, ILT-based methods optimize the mask within a larger solution space and thus achieve better performance.

ILT algorithms usually adopt iterative methods, such as gradient descent to optimize the objective function, which requires a lot of iterations for convergence. Although ILT has shown satisfactory performance on mask optimization [13], [14], [15], the shrinking size of the technology node and increasing complexity of mask patterns pose significant challenges to the runtime overhead. There have been many exciting explorations of ILT acceleration in recent years, and these works can be generally divided into two categories. The first one is to design GPU-accelerated algorithms by fully utilizing the massive computing resources in GPUs. For example, Yu et al. [8] relied on the CUDA toolkit to implement a GPU-accelerated Fourier transform algorithm, accelerating a critical and time-consuming step in the lithography simulation model. The second one is to learn the whole ILT solver using stacked convolutional layers. As illustrated in Fig. 1(b), related methods [5], [6], [7], [9] utilize a pretrained CNN-based generation model such as GAN [16], [17] or U-Net [18], [19] to quickly approximate an initial mask solution of the test target and then conduct further refinements on the initial mask to improve the solution quality. We summarize these methods as “generative ILT.”

Although ILT acceleration has made signification progress with the push of previous works [5], [6], [7], [8], [9], there are still some issues with these methods. GPU-accelerated ILT mainly focuses on accelerating the runtime of each iteration but does not necessarily reduce the number of iterations, thus still causing a long time to execute the entire algorithm. For example, the GPU-accelerated algorithm GLS-ILT proposed in [8] still spends around 100.1 s optimizing a 2048×2048 mask clip, which is unacceptable, especially when applied to a large full-chip mask. We hope that such an optimization task should be finished within a few seconds, and only in this way can we achieve efficient VLSI design. As for “generative ILT” depicted in Fig. 1(b), the initial mask solution approximated by the generation model may contribute to reducing the number of iterations, effectively improving the ILT runtime. However, there still exist some drawbacks. (1) According to our empirical study, we find that the quality of the initial solution is always low, thus demanding a long-time refinement. Take Neural-ILT [7] as an example, when evaluated on the ICCAD 2013 benchmark [20], the average

Manuscript received 2 March 2023; revised 3 August 2023; accepted 15 September 2023. Date of publication 10 October 2023; date of current version 21 February 2024. This work was supported in part by the Research Grants Council of Hong Kong SAR under Project CUHK14208021; in part by the National Key R&D Program of China under Grant 2020YFA0711900 and Grant 2020YFA0711903; and in part by the National Natural Science Foundation of China under Grant 62204066. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Bei Yu.)*

Binwu Zhu, Su Zheng, Ziyang Yu, Guojin Chen, Bei Yu, and Martin D. F. Wong 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 511453, China.

Fan Yang is with the State Key Laboratory of ASIC and System, Microelectronics Department, Fudan University, Shanghai 200437, China.

Digital Object Identifier 10.1109/TCAD.2023.3323164

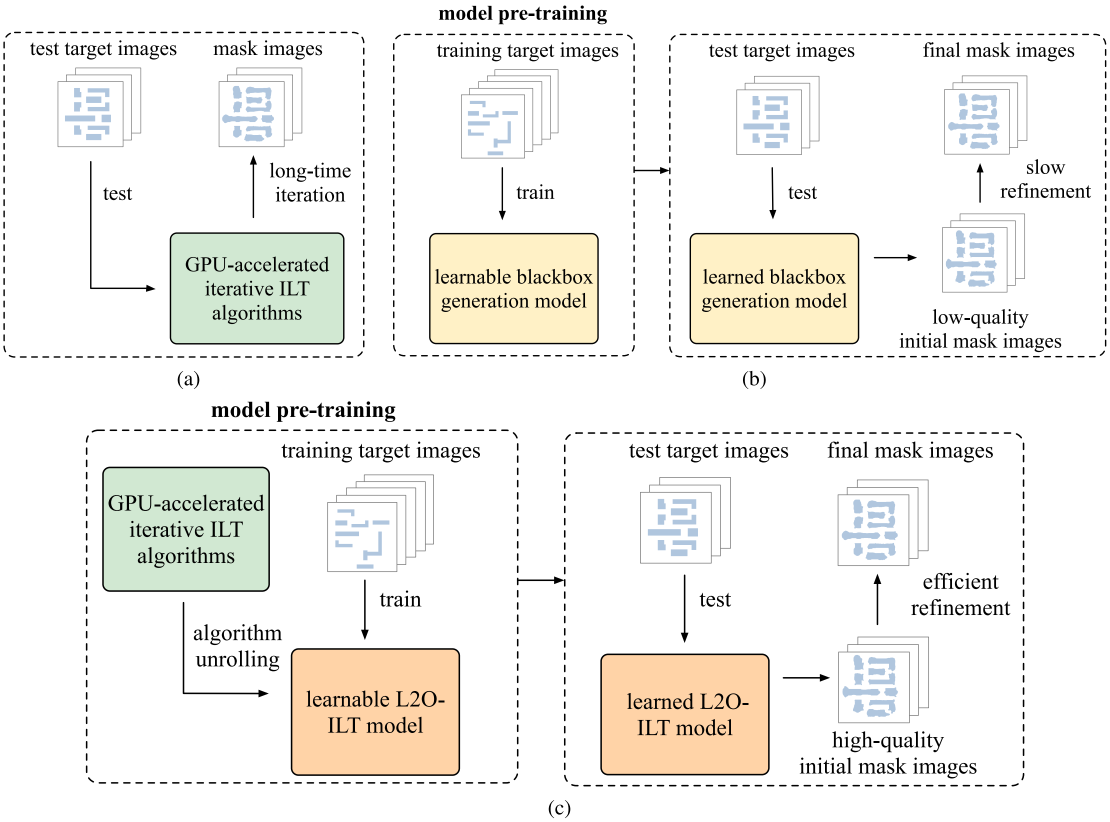


Fig. 1. Three ILT acceleration methodologies. (a) GPU-accelerated ILT. (b) Generative ILT. (c) Learning to optimize ILT.

L_2 loss between the target image and the wafer image of the initial mask is around 74023.7. Such a loss result is far away from their final loss result after refinement (37515.30) [7]. Even though the initial mask can be generated within 0.1 s by Neural-ILT [7], it still takes a significant amount of iterations for refinement. We suppose that such a drawback is mainly caused by the black-box nature of generation models. We are unaware of what these black-box models learn during the training process. The structures of these models are not specifically customized for mask optimization tasks, making it challenging to capture the domain-specific knowledge of OPC. (2) To model the ILT process more accurately, some works [21], [22] adopt convolution kernels with the same size as optical kernels in the lithography process. However, the optical kernel is usually large, e.g., 35×35, and convolving such a kernel with a 2048×2048 mask consumes excessive computation resources. Therefore, these works fail to achieve acceleration on large-scale OPC problems. Moreover, to reduce the model complexity, these models are usually built on simple ILT algorithms [23], [24], which only consider optimizing the design target under the nominal process condition but neglect the process window with different process corners.

In this work, we propose L2O-ILT, a deep learning-based framework keeping both advantages of conventional ILT and "generative ILT" and overcoming their issues. The basic framework is illustrated in Fig. 1(c), which is totally different from previous methods [5], [6], [7], [8], [9]. The novel ILT method is inspired by the learning-to-optimize (L2O) scheme [25], [26], [27], [28], [29] in machine learning, which aims to incorporate prior knowledge of the optimization algorithm into a learning model. Specifically, we build up an ILT-inspired learning model by unrolling the entire algorithm. The structure of our model is no longer stacking convolutional layers like previous methods [5], [6], [7], [9]. Instead, each layer is customized to represent each iteration of the ILT algorithm. This representation projects the ILT problem into a hyperspace that can be more efficiently solved by deep learning algorithms. And the model training can be regarded as automatically tuning the algorithm parameters, which are hand-crafted in the conventional ILT algorithm. In addition, such a model is inherently equipped with interpretability and prior knowledge of mask optimization, which will contribute to robustness and ensure a high-quality initial mask for efficient refinement. Besides, a specialized optimization mechanism called alternating optimization is designed for our model to jointly optimize the printed image under different process conditions. An adaptive solution space is developed to accelerate the convergence rate of our algorithm while saving computation resources. We summarize the contributions of this article as follows.

1) A deep learning-based and ILT-inspired neural network called L2O-ILT is developed, which incorporates domain-specific prior knowledge of mask optimization.
2) The network architecture is designed by unrolling the ILT algorithm and modeling each iteration as a neural network layer.

3) We develop an alternating optimization mechanism and an adaptive solution space method to improve the conventional ILT algorithm and further the performance.
4) L2O-ILT is able to generate high-quality initial mask solutions, which can be efficiently refined. Experimental results show that our model achieves better performance on both mask printability and runtime than previous methods.

The remainder of this article is organized as follows. Section II gives an introduction preliminaries about lithography model and inverse lithography technologies. Section III gives the detailed elaboration of the L2O-ILT model with an alternating optimization strategy and an adaptive solution space mechanism. Section IV details experimental results and comparisons, followed by conclusion in Section V.

II. Preliminaries

In this section, we will introduce the problem formulation and some preliminary knowledge related to this work.

A. Lithography Simulation Model

During the lithography process, an input mask $\boldsymbol{M}$ is first transformed through an optical projection system into the aerial image $\boldsymbol{I}$. The distribution of light intensity at the wafer plane then undergoes development and etching processes to form the printed image $\boldsymbol{Z}$.

To simulate the lithography process, a mathematical model is proposed in [30], which is composed of two components, optical projection model and photoresist model. For the optical projection process, the Hopkins diffraction model of the partially coherence imaging system is used to approximate the projection behavior. In mathematics, the aerial image $\boldsymbol{I}$ can be obtained by convolving the mask $\boldsymbol{M}$ with a set of optical kernels $\boldsymbol{H}$, formulated as

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

where "$\otimes$" represents the convolution operation, h_k is the kth optical kernel of the optical kernel set $\boldsymbol{H}$, and w_k is the corresponding weight of the coherent system. To save the computation resources, an N_hth order approximation to the partially coherent system is proposed in [4], represented as

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

where the kernel number N_h is 24 in our work. After optical simulation, the aerial image $\boldsymbol{I}$ is input into the photoresist model with an intensity threshold I_{th}, which indicates the exposure level. And the final binary printed image $\boldsymbol{Z}$ is calculated by the following step function:

$$\boldsymbol{Z}(x, y) = \begin{cases} 1, & \boldsymbol{I}(x, y) \geq I_{\text{th}} \\ 0, & \boldsymbol{I}(x, y) < I_{\text{th}} \end{cases} \tag{3}$$

Following the ICCAD 2013 contest settings [20], I_{th} is set as 0.225 in our implementation.

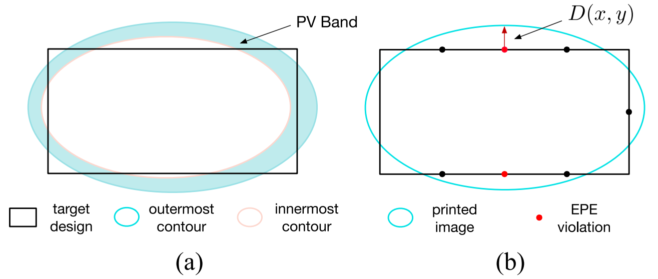


Fig. 2. (a) Visualization of PV Band measurement. (b) Visualization of EPE measurement.

B. OPC Evaluation Metrics

Process Variation Band (PVB): In the real-world lithography system, process variations will cause deviations in the final printed image, leading to printing failure. Under different lithography conditions, such as focus/defocus depth and incident light intensity, printed images have various contour results. PVB computes the bitwise-XOR region between the outermost and innermost contour as shown in Fig. 2(a) to evaluate the printing robustness.

Square L_2 Error: Given the target image $\boldsymbol{Z}_{\text{target}}$ and the printed image $\boldsymbol{Z}_{\text{nominal}}$, which represents the image printed via nominal lithography process condition, the square L_2 loss is calculated as $||\boldsymbol{Z}_{\text{nominal}} - \boldsymbol{Z}_{\text{target}}||_2^2$.

Edge Placement Error (EPE): EPE is used to evaluate the difference of the contour between the target design $\boldsymbol{Z}_t$ and the image $\boldsymbol{Z}_{\text{nom}}$. To calculate the EPE, a series of points are sampled along the contour of the target design, as shown in Fig. 2(b). If the distance $D(x, y)$ between the target design to the printed image is larger than an EPE constraint th_{EPE}, the point (x, y) is labeled as a EPE violation

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

Mask Manufacturing Shot Count: Since ILT naturally generates purely curvilinear features, conventional fracturing methods require a large number of small rectangles to approximate the shape. Mask data preparation (MDP) is used to fracture the shapes on the masks into nonoverlapping rectangles, known as variable shaped-beam (VSB) shots, to ensure mask printability. The shot count is used to evaluate the complexity of mask patterns.

With the evaluation metrics defined above, we formulate the mask optimization problem as follows:

Problem 1 (Mask Optimization): Given a target image $\boldsymbol{Z}_t$, the objective of mask optimization is to find a mask $\boldsymbol{M}$, whose printed image through the lithography process is supposed to be close to the target image and keep stable under different process conditions, such that the EPE, L_2 loss, PV Band and manufacturing shot count are minimized.

C. Inverse Lithography Techniques

The objective of the conventional ILT-based method is to find an optimized mask $\boldsymbol{M}_{\text{opt}} = g^{-1}(\boldsymbol{Z}_t, \boldsymbol{C}_{\text{nom}})$, where $\boldsymbol{Z}_t$ is the design target, and $g(\cdot, \boldsymbol{C}_{\text{nom}})$ stands for the lithography process under the nominal process condition. Usually, we can not obtain the inverse function of g to compute the closed-form solution. The optimal mask is searched by computing the gradient of an objective function F_{obj} and using the gradient to guide the adjustment of each pixel value.

To calculate the gradient, all variables during the lithography process have to be continuous. Therefore, the binarized and constrained pixel values of mask $\boldsymbol{M}$ and printed image $\boldsymbol{Z}$ should be relaxed. To achieve this, we first introduce an auxiliary and unconstrained variable $\boldsymbol{P}$ and assume that $\boldsymbol{M}$ is determined by $\boldsymbol{P}$. The relationship between them is depicted by a sigmoid function in (6)

$$\boldsymbol{M} = \frac{1}{1 + \exp\left(-\theta_M \boldsymbol{P}\right)} \tag{5}$$

where θ_M defines the steepness of the sigmoid function.

Then, the whole lithography process can be represented as: $\boldsymbol{P} \rightarrow \boldsymbol{M} \rightarrow \boldsymbol{I} \rightarrow \boldsymbol{Z}$. Note that the original function that maps $\boldsymbol{I}$ to $\boldsymbol{Z}$ is a threshold function as formulated in (3), which is also undifferentiable. Therefore, we approximate it using another sigmoid function

$$\boldsymbol{Z} = \frac{1}{1 + \exp\left(-\theta_Z(\boldsymbol{I} - I_{\text{th}})\right)} \tag{6}$$

where θ_Z defines the steepness and I_{th} represents the intensity threshold as shown in (3). In this way, the whole process becomes differentiable and each iteration of the optimization algorithm can be formulated as follows:

$$\boldsymbol{P}^{(j)} = \boldsymbol{P}^{(j-1)} - \eta \frac{\partial F_{\text{obj}}}{\partial \boldsymbol{P}^{(j-1)}} \tag{7}$$

where η is the step size of gradient descent. $\boldsymbol{P}^{(j)}$ indicates the variable $\boldsymbol{P}$ at the jth iteration. After finally obtaining the $\boldsymbol{P}_{\text{opt}}$ by minimizing the objective function F_{obj}, we binarize $\boldsymbol{P}_{\text{opt}}$ to $\boldsymbol{M}_{\text{opt}}$, which is the final optimized mask solution.

III. L2O-ILT Algorithm

In this section, we first discuss an optimization mechanism in Section III-A called alternating optimization, which solves the issue that the conventional ILT [4] does not achieve satisfactory joint optimization of design targets under different conditions. Then, we develop our learning model L2O-ILT in Section III-B, where each layer is constructed based on our proposed ILT algorithm with alternating optimization, and the whole architecture is equipped with strong prior knowledge and highly interpretable. The model training and refinement strategy is explained in Section III-C and III-D. A technique called adaptive solution space is proposed in Section III-E to help our model accelerate the convergence rate as well as keep the solution quality.

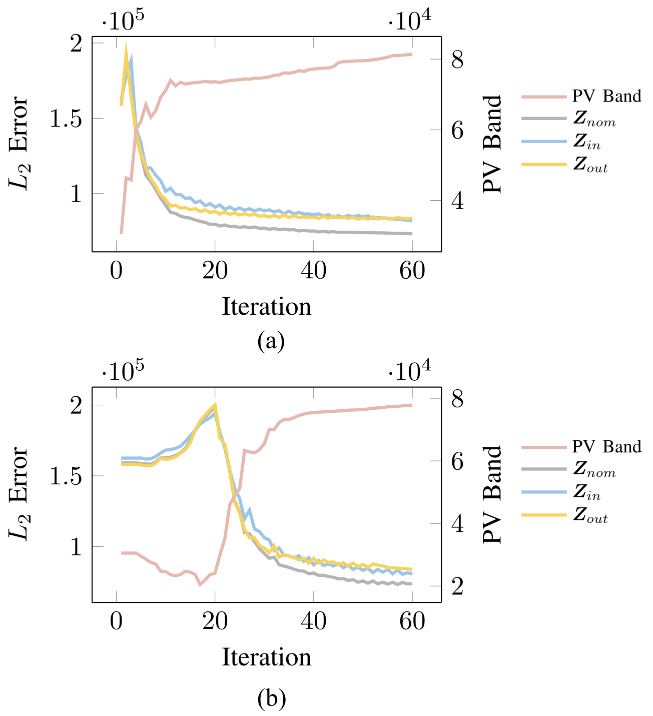


Fig. 3. Change of loss terms and PV Band in (a) conventional ILT and (b) proposed alternating optimization scheme.

A. Alternating Optimization

As illustrated in Section II-C, the general implementation of ILT-based methods is to first define an objective function of the mask, which is then optimized using numerical approach. Therefore, the quality of final solution is closely related to the definition of the objective function. Given an input $\boldsymbol{P}$, the classical pixel-based ILT [4] gives the objective function to be minimized as follows:

$$\begin{aligned} F_{\text{target}} &= L_{\text{nominal}} + L_{\text{out}} + L_{\text{in}} \\ &= \left\|\boldsymbol{Z}_{\text{nominal}} - \boldsymbol{Z}_{\text{target}}\right\|_2^2 + \left\|\boldsymbol{Z}_{\text{out}} - \boldsymbol{Z}_{\text{target}}\right\|_2^2 \\ &\quad + \left\|\boldsymbol{Z}_{\text{in}} - \boldsymbol{Z}_{\text{target}}\right\|_2^2 \end{aligned} \tag{8}$$

where $\boldsymbol{Z}_{\text{nominal}} = g(\boldsymbol{P}, \boldsymbol{C}_{\text{nominal}})$, $\boldsymbol{Z}_{\text{out}} = g(\boldsymbol{P}, \boldsymbol{C}_{\text{out}})$, and $\boldsymbol{Z}_{\text{in}} = g(\boldsymbol{P}, \boldsymbol{C}_{\text{in}})$. $\boldsymbol{C}_{\text{out}}$ and $\boldsymbol{C}_{\text{in}}$ stand for two extreme conditions, under which the outer-most and inner-most images will be printed.

Under the guidance of F_{target}, the printed images under different process conditions are jointly pushed toward the target pattern, which is actually a desired property of an optimized mask. However, according to our empirical study as shown in Fig. 3(a), we find that while all three loss terms are gradually minimized, the PV Band metric is negatively optimized. This is because minimizing $||\boldsymbol{Z}_{\text{out}} - \boldsymbol{Z}_{\text{target}}||_2^2 + ||\boldsymbol{Z}_{\text{in}} - \boldsymbol{Z}_{\text{target}}||_2^2$ cannot guarantee the minimization of $||\boldsymbol{Z}_{\text{out}} - \boldsymbol{Z}_{\text{in}}||_2^2$ in mathematics. We also depict the change of L_{nominal}, L_{in}, L_{out}, and PV Band in Fig. 3(b). Therefore, although optimizing F_{target} contributes to reducing the error between the printed image and the real target, it results in a high PV Band value, leading to a large process window. We call such an objective function optimization "target-driven optimization" and we propose that the optimization configuration is supposed to be improved.

It can be easily seen that convergence rates of all three loss terms are drastically reduced after a certain number of iterations. Based on such an observation, we replace several iterations of optimizing F_{target} with optimizing F_{pvb}, formulated as

$$F_{\text{pvband}} = ||\boldsymbol{Z}_{\text{out}} - \boldsymbol{Z}_{\text{in}}||_2^2 \tag{9}$$

which is directly related to PV Band performance. Optimizing F_{pvband} can be regarded as "PV Band-driven optimization." Instead of optimizing a weighted sum of F_{target} and F_{pvband}, we choose to decouple the optimization of these two objectives. This will make it more convenient to encapsulate basic modules in our deep learning-based framework, which will be shown in the following Section III-B. And according to our experimental results shown in Fig. 3(b), we find that alternating "target-driven optimization" and "PV Band-driven optimization" can achieve satisfactory improvement in PV Band while slightly affecting L_{nominal}, L_{in}, and L_{out}. Our designed optimization scheme can be represented as

$$\boldsymbol{P}^{(j)} = \begin{cases} \boldsymbol{P}^{(j-1)} - \eta \frac{\partial F_{\text{pvband}}}{\partial \boldsymbol{P}^{(j-1)}}, & j < Q \\ \boldsymbol{P}^{(j-1)} - \eta \frac{\partial F_{\text{target}}}{\partial \boldsymbol{P}^{(j-1)}}, & j \geq Q \end{cases} \tag{10}$$

where Q is a hyper-parameter to balance the performance of the process window and the L_2 error between the printed image and target design.

B. Model Architecture of L2O-ILT

Classic ILT-based mask optimization algorithms are built upon numerical approaches in a theoretically justified manner. In spite of the high interpretability, the performance heavily depends on human experiences, such as how to select appropriate parameters in the algorithm. Since these algorithms are sensitive to initial conditions and parameters chosen, the optimization results may be easily stuck in a local optimum state. Furthermore, a large number of optimization iterations are usually required to achieve an acceptable performance level, and thus these algorithms can be computationally expensive.

As explained in Section I, "generative ILT" methods use learning-based models to quickly generate initial mask solutions and conduct further refinement. Regardless of its higher efficiency compared with conventional ILT, we notice that the quality of the initial mask is always low, which still requires a long-time correction to improve the solution quality. The inferior mask is mainly caused by the black-box property of generation models, structures of which are difficult to be customized for mask optimization problems.

In accordance with the aforementioned observations, we propose an ILT algorithm-inspired learning model, L2O-ILT, which can generate a high-quality mask solution for fast refinement. The structure of L2O-ILT is not simply composed of stacking convolutional layers like "generative ILT" methods [5], [6], [7], [9]. Instead, each layer of our model is customized with prior knowledge for mask optimization tasks. To be specific, we unroll the entire ILT algorithm and use a neural network layer to represent each iteration of gradient descent as formulated in (10), where the $\boldsymbol{P}^{(j)}$ and $\boldsymbol{P}^{(j-1)}$ can be regarded as the output and input of the jth layer.

Based on our proposed alternating optimization scheme in Section III-A, two kinds of neural network layers, target-driven block and PV Band-driven block, are, respectively, designed as shown in Fig. 4. The architecture of L2O-ILT can be regarded as a time-unfolded recurrent neural network. In addition, our model can also keep the consistency advantage of ILT algorithms, i.e., we can still get a similar mask even if the pattern is offset. This is ensured by the translation invariance property of the lithography process as proved in [31].

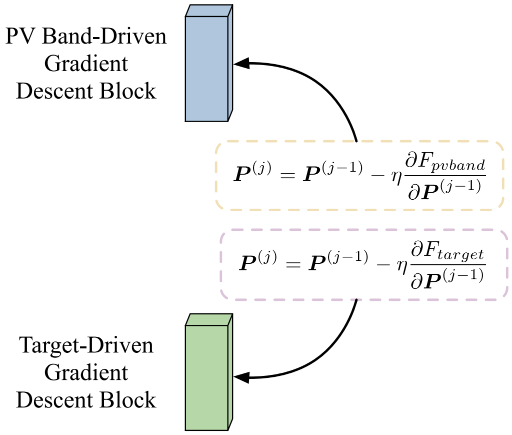

PV Band-Driven Gradient Descent Block
$\boldsymbol{P}^{(j)} = \boldsymbol{P}^{(j-1)} - \eta\frac{\partial F_{pvband}}{\partial \boldsymbol{P}^{(j-1)}}$
$\boldsymbol{P}^{(j)} = \boldsymbol{P}^{(j-1)} - \eta\frac{\partial F_{target}}{\partial \boldsymbol{P}^{(j-1)}}$
Target-Driven Gradient Descent Block

Fig. 4. Each iteration of L2O-ILT algorithm is represented as a neural network layer.

The computation of the Target-Driven block exactly represents $\boldsymbol{P} = \boldsymbol{P} - \eta(\partial F_{\text{target}}/\partial \boldsymbol{P})$. To further illustrate the concrete computation operation, we first represent the $(\partial F_{\text{target}}/\partial \boldsymbol{P})$ as follows:

$$\frac{\partial F_{\text{target}}}{\partial \boldsymbol{P}} = \frac{\partial L_{\text{nominal}}}{\partial \boldsymbol{P}} + \frac{\partial L_{\text{out}}}{\partial \boldsymbol{P}} + \frac{\partial L_{\text{in}}}{\partial \boldsymbol{P}}. \tag{11}$$

The gradient calculations of all three terms are similar, and here we take $(\partial L_{\text{nominal}}/\partial \boldsymbol{P})$ as an example, which is computed by

$$\begin{aligned} \frac{\partial L_{\text{nominal}}}{\partial \boldsymbol{P}} &= 2\theta_Z\theta_M \boldsymbol{M}\circ(1-\boldsymbol{M})\big\{\boldsymbol{H}_{\text{nominal}}\otimes\big[\big(\boldsymbol{Z}_{\text{nominal}} \\ &\quad - \boldsymbol{Z}_{target}\big)\circ\boldsymbol{Z}_{\text{nominal}}\circ(1-\boldsymbol{Z}_{\text{nominal}})\circ\big(\boldsymbol{M}\otimes\boldsymbol{H}^*_{\text{nominal}}\big)\big] \\ &\quad + \boldsymbol{H}^*_{\text{nominal}}\otimes[(\boldsymbol{Z}_{\text{nominal}} - \boldsymbol{Z}_t)\circ\boldsymbol{Z}_{nominal}\circ(1-\boldsymbol{Z}_{\text{nominal}}) \\ &\quad \circ(\boldsymbol{M}\otimes\boldsymbol{H}_{\text{nominal}})]\big\} \end{aligned} \tag{12}$$

where "$\circ$" indicates the matrix element-wise multiplication and "$\otimes$" stands for the convolution operation. The PV Band-driven block is designed in a similar way, which precisely represents the computation of $(\partial \boldsymbol{F}_{\text{pvband}}/\partial \boldsymbol{P})$. In addition, all convolution operations in our algorithm are implemented via FFT convolution to save computation resources. Since the optical kernel size is quite large, given a $k \times k$ (e.g., 35×35) optical kernel and $N \times N$ mask (e.g., 2048×2048), the computation complexity of FFT convolution is $\mathcal{O}(N^2 \log N^2)$, less than the complexity $\mathcal{O}(k^2N^2)$ of direct convolution. All the matrix computations can be easily implemented with the deep learning toolkit, such as Pytorch [32], which provides matrix computing with strong acceleration implemented by CUDA kernel.

Stacking N_{pvband} PV Band-driven blocks and N_{target} target-driven blocks forms a deep neural network, which is exactly our L2O-ILT as shown in Fig. 5, and passing through the entire neural network is equivalent to executing the ILT algorithm a number of iterations. In L2O-ILT, we set both N_{pvband} and N_{target} as 5. The convergence rate of our L2O-ILT can be boosted via model training. All learnable parameters, such

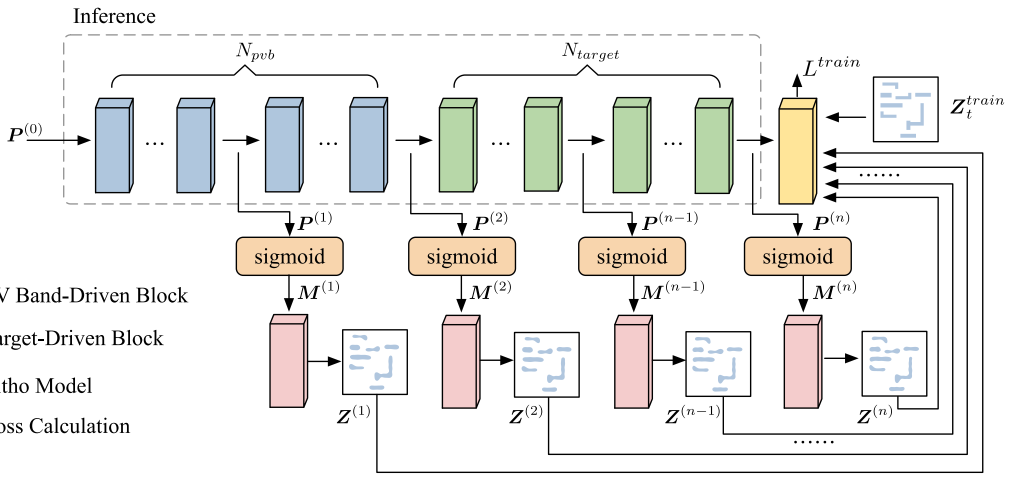


Fig. 5. Stacking multiple layers forms a neural network and passing through L2O-ILT is equivalent to executing an iterative ILT algorithm. Training L2O-ILT can be interpreted as tuning the parameter that is manually determined in the original ILT algorithm.

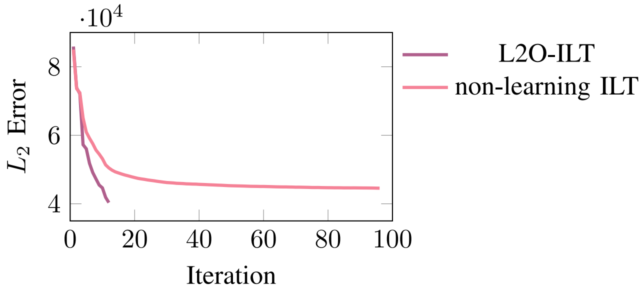


Fig. 6. Convergence rate comparison between conventional ILT and L2O-ILT model during inference.

as the step size of gradient descent, updated during the training process are all from the original ILT algorithm. Therefore, the model training can be naturally interpreted as a parameter auto-tuning process to achieve much faster convergence than nonlearning ILT with hand-crafted parameters. We show the convergence rate comparison in Fig. 6. Compared with conventional nonlearning ILT, L2O-ILT is able to achieve convergence within a much smaller number of iterations, resulting in significant ILT acceleration. Passing through one layer can be regarded as executing the original nonlearning ILT algorithms for multiple iterations. In addition, the learnable parameters can also help avoid the local optimum state, contributing to the robustness of our algorithm.

Note that the number of iterations also indicates the number of layers in L2O-ILT. The training strategy will be explained in Section III-C. Overall, our proposed framework seamlessly incorporates the prior knowledge of mask optimization and achieves ILT acceleration with deep learning, and therefore we call it "learning to optimize ILT."

C. Interpretable Self-Supervised Learning

In order to accelerate the convergence rate of L2O-ILT, a specialized interpretable training strategy is proposed. As illustrated in Section III-B, each layer of our neural network is equivalent to an optimization iteration, and each layer outputs a mask that has not been fully optimized. Therefore, we can directly supervise the intermediate-generated mask $\boldsymbol{M}^{(i)}$ computed from $\boldsymbol{Z}^{(i)}$ using the training target design $\boldsymbol{Z}_{\text{target}}^{\text{train}}$. Such a training strategy can be regarded as providing a look-ahead mechanism, which forces $\boldsymbol{Z}^{(i)}$ to be close to the real target $\boldsymbol{Z}_{\text{target}}^{\text{train}}$. As a result, the error between the printed image $\boldsymbol{Z}^{(n)}$ of the final mask $\boldsymbol{M}^{(n)}$ and $\boldsymbol{Z}_{\text{target}}^{\text{train}}$ will be reduced efficiently. The training loss function can be formulated as

$$L^{\text{train}} = \sum_{i=1}^{n} l^{(i)}\left(\boldsymbol{M}^{(i)}, \boldsymbol{Z}_{\text{target}}^{\text{train}}\right) \tag{13}$$

where n is a configurable hyper-parameter, representing the number of intermediate masks that we supervise with $\boldsymbol{Z}_{\text{target}}^{\text{train}}$, as shown in Fig. 5. The loss between $\boldsymbol{M}^{(i)}$ and $\boldsymbol{Z}_{\text{target}}^{\text{train}}$ is decided by its printed image $\boldsymbol{Z}^{(i)}$, and the computation of $l^{(i)}$ is calculated as

$$l^{(i)} = \left\|\boldsymbol{Z}_{\text{nominal}}^{(i)} - \boldsymbol{Z}_{\text{target}}^{\text{train}}\right\|_2^2 + \left\|\boldsymbol{Z}_{\text{out}}^{(i)} - \boldsymbol{Z}_{in}^{(i)}\right\|_2^2 \tag{14}$$

where $\boldsymbol{Z}_{\text{nominal}}^{(i)}$, $\boldsymbol{Z}_{\text{out}}^{(i)}$, and $\boldsymbol{Z}_{\text{in}}^{(i)}$ stands for the printed image through our lithography module under different conditions. Such a training loss function design contributes to jointly optimizing PV Band, L_2 error, and EPE. It should be reminded that *the parameters of all lithography modules mapping from* $\boldsymbol{M}^{(i)}$ *to* $\boldsymbol{Z}^{(i)}$ *are fixed and unlearnable*, so as to ensure the correctness of the lithography process.

In addition, it can be observed that our training scheme is self-supervised learning. To be specific, we directly adopt the target design as the supervision signal, which is totally different from previous "generative ILT" methods [6], [7], [9]. When given a set of training target design $\mathcal{Z}_{\text{target}}^{\text{train}}$, they demand a corresponding optimized mask set $\mathcal{M}^*$ acting as the "ground truth" signal to supervise the mask output by the black-box generation model. We argue that this training scheme is not reasonable. This is because when given a target design, there is no way to know what its actual corresponding mask is. Therefore, the optimized masks $\mathcal{M}^*$ utilized by previous methods [6], [7], [9] are actually approximated optimized masks, which are obtained from conventional ILT algorithms. In this way, the "ground truth" signals themselves are not accurate

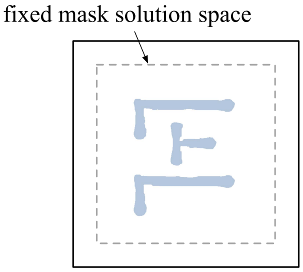


Fig. 7. Mask optimization result of a simple target design pattern.

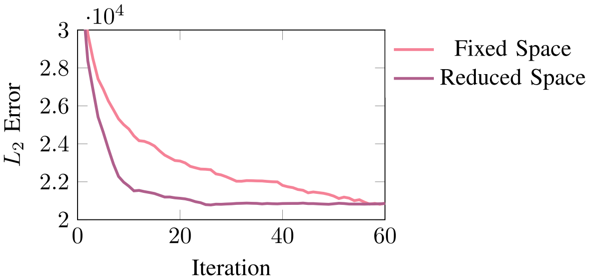


Fig. 8. Comparison of the convergence rate in fixed solution space and reduced solution space.

to act as the supervision signals for training. Moreover, the model training process will further accumulate the error. This also provides another reason to explain why the initial solution generated by these generation models is inferior.

D. Inference and Refinement

We have finished discussions about the model architecture and training process, both of which are highly interpretable. Overall, the model architecture is unrolling the iterative algorithm, and the model training can be regarded as tuning the parameters to improve the original manual parameter configuration to accelerate optimization convergence.

Finally, when applied to design targets from the test dataset, our learned L2O-ILT model is able to generate a superior initial mask solution instantly. To further improve the mask quality, refinement is conducted on the initial mask. When given a test target $\boldsymbol{Z}_t^{\text{test}}$, the mask refinement is achieved by finetuning the parameters of the last layer by optimizing the following loss function:

$$L^{\text{finetune}} = \gamma \left\| \boldsymbol{Z}_{\text{nominal}}^{(n)} - \boldsymbol{Z}_{\text{target}}^{\text{test}} \right\|_2^2 + \frac{1}{\gamma} \left\| \boldsymbol{Z}_{\text{out}}^{(n)} - \boldsymbol{Z}_{\text{in}}^{(n)} \right\|_2^2 \quad (15)$$

where γ is an adaptive factor to balance these two loss terms, and it is computed as the ratio between the L_2 error and PV Band of the initial mask. Because of the high-quality initial mask, the refinement can be quickly converged within a very small number of iterations.

E. Adaptive Solution Space

Conventional ILT algorithms [4], [5], [6], [7], [8], [9] optimize the mask pixel within a determined wide-ranging area, e.g., 1280 × 1280. However, we observe that the areas of the optimized mask are always close to the targets. Specifically, the updated pixels always lie in the neighbourhood of the target patterns. Therefore, we try to leverage this prior knowledge in our framework. We propose that the space can be suitably narrowed while keeping a high-quality mask solution. Based on such a motivation, we design an adaptive solution space in our algorithm, and this mechanism can also effectively avoid the emergence of outlier features in the optimized masks. In addition, a smaller solution space will also contribute to a faster convergence rate as well as saving computation resources. The experimental results show that the convergence rate of mask optimization will increase, as shown in Fig. 8.

We design an adaptive mechanism to dynamically adjust the solution space according to the specific target patterns. It is based on such an observation that the optimized mask area always lies in the neighborhood of the target pattern. Therefore, our adaptive solution space is achieved by expanding the target pattern. Usually, this can be implemented by convolution with a square kernel $\mathbf{1} \in \mathbb{R}^{s\times s}$, where $\mathbf{1}$ is a matrix in which all elements are 1, and N is the image size. Such a dilation operation is feasible but not efficient, which has $\mathcal{O}(s^2N^2)$ complexity. In this work, an agile algorithm is designed to satisfy our requirements. To specific, we can directly move the vertices to adjust the solution space. It is noted that the layout patterns tested in this work are from ICCAD 2013 CAD Contest [20] where all patterns are regular polygon shapes and represented as a vector of vertices as shown in Fig. 9. Therefore, all vertices are off the shelf. And there are no extra workloads to transfer the pixel representation to the vertex representation. The movement direction of each vertex $\vec{v}_i = (x_i, y_i)$ is determined by its convexity-concavity and two neighborhood vertices $\boldsymbol{v}_{i-1}$ and $\boldsymbol{v}_{i+1}$, which can be formulated as

$$\begin{aligned} \boldsymbol{u} &= (\vec{v}_{i+1} - \vec{v}_i) \times (\vec{v}_i - \vec{v}_{i-1}) \\ &= (0, 0, (x_{i+1} - x_i)(y_i - y_{i-1}) - (y_{i+1} - y_i)(x_i - x_{i-1})) \end{aligned} \quad (16)$$

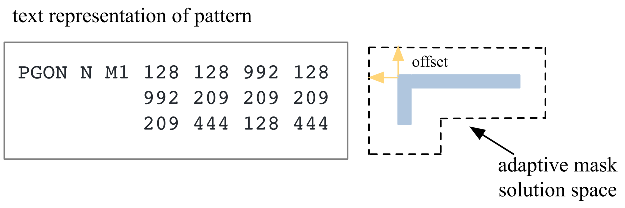
$$c = \text{sign}(\boldsymbol{u}_z) \quad (17)$$

$$x_i' = x_i + c \cdot \text{sign}((x_i - x_{i-1}) - (x_{i+1} - x_i)) \cdot \text{offset} \quad (18)$$

$$y_i' = y_i + c \cdot \text{sign}((y_i - y_{i-1}) - (y_{i+1} - y_i)) \cdot \text{offset}. \quad (19)$$

In (16), to compute the cross product, we assume that vectors $(\vec{v}_{i+1} - \vec{v}_i)$ and $(\vec{v}_i - \vec{v}_{i-1})$ have a 0 z-axis component. The coefficient c is to determine whether the vertex is convex or concave according to the positive or negative of the z-axis component $\boldsymbol{u}_z$ of $\boldsymbol{u}$, and sign$(\cdot)$ represents the sign function. We have $c = 1$ when the vertex is convex. (x_i', y_i') indicates the coordinate of vertex $\boldsymbol{v}_i$ after movement, and "offset" is a configurable hyper-parameter to control the size of solution space and further accelerates the convergence rate.

Adjusting the space by moving vertices requires $\mathcal{O}(p)$ computation complexity, where p represents the number of vertices, typically less than 100. Therefore, such an algorithm is much more efficient than the traditional dilation operation with complexity $\mathcal{O}(s^2N^2)$. The generated adaptive solution space $\boldsymbol{S}_{\text{ada}}$ can be incorporated into our original gradient descent formulation in Fig. 4 to restrict the range of mask pixels update.



text representation of pattern

```
PGON N M1 128 128 992 128
          992 209 209 209
          209 444 128 444
```

Fig. 9. Adaptive solution space via the movement of vertices.

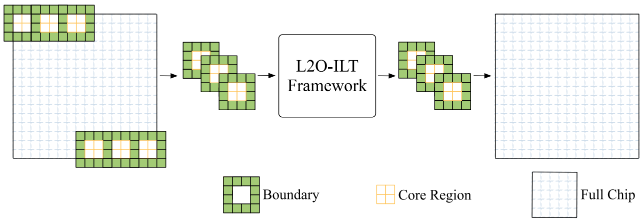


Fig. 10. Adaptation of L2O-ILT on full chip.

As shown in Fig. 9, we only allow the pixels within the space to be updated during the optimization process. The gradient descent formulations combined with the adaptive solution space are now represented as

$$\boldsymbol{P}^{(j)} = \boldsymbol{P}^{(j-1)} - \eta \frac{\partial F_{\text{pvband}}}{\partial \boldsymbol{P}^{(j-1)}} \circ \boldsymbol{S}_{\text{ada}} \tag{20}$$

$$\boldsymbol{P}^{(j)} = \boldsymbol{P}^{(j-1)} - \eta \frac{\partial F_{\text{target}}}{\partial \boldsymbol{P}^{(j-1)}} \circ \boldsymbol{S}_{\text{ada}} \tag{21}$$

where $\boldsymbol{S}_{\text{ada}}$ acts as a filter. In $\boldsymbol{S}_{\text{ada}}$, the values within the adaptive solution space are 1 and others are 0.

F. Applied on Full Chip

The above methodology mainly discusses mask optimization on layout patterns of small size, i.e., 2048×2048. With the development of semiconductors and the shrinking size of transistors, the chip scale is constantly growing, which is usually much larger than the patterns used in academic research. To overcome this issue, we also explore the adaptation of our L2O-ILT on the full chip. Inspired by [33], our proposed algorithm is illustrated in Fig. 10, a combination of our L2O-ILT and the large tile global perception algorithm proposed by [33]. As shown in Fig. 10, we adopt a sliding window to scan over the entire chip, dividing the large full chip into smaller chips. It is noted that each window includes two parts, the core region and the boundary region. The layout patterns within the boundary region in each sliding window will be ignored, and the core part is the mask region that we want to obtain its mask optimization result. As discussed in [33], such a sliding-window manner can help minimize boundary distortion effects. After feeding each tile into our L2O-ILT framework, the optimized mask of all core parts can be obtained, which will then be concatenated back. The stitching result is the optimized mask of the full chip.

IV. Experimental Results

We implement our entire framework L2O-ILT with the Pytorch library [32] and test it on a Linux system with 2.3 GHz Intel Xeon CPU and a single Nvidia GeForce RTX 3090 GPU. The evaluation data to test the model performance are from ICCAD 2013 CAD Contest [20], which includes ten industrial M1 designs on the 32 nm design node and also provides the lithography engine. The dataset used for training our L2O-ILT is obtained from the authors of GAN-OPC [6].

TABLE I
Benchmark Information of ICCAD 2013 Dataset

Bench	Area (nm^2)
case1	215344
case2	169280
case3	213504
case4	82560
case5	281958
case6	286234
case7	229149
case8	128544
case9	317581
case10	102400

A. Comparison With State-of-the-Art

We compare the performance of the proposed L2O-ILT with other state-of-the-art mask optimization methods, and the detailed results are listed in Tables II and III. We learned that there exists offsets between the initial generated mask between the work in [4], [6], [7], [9], and [34] and we noted that the offset of the initial mask would slightly affect the mask printability and complexity. To make a fair comparison, we set two versions of L2O-ILT results following corresponding experimental settings.

As listed in Table II, compared with classical ILT [4] (denoted as ILT), the L_2 and PV Band are reduced by 47.1% and 21.6%, respectively, and the EPE count is less than one-third of [4]. Compared with two "generative ILT" GAN-OPC [6], and DevelSet [9], which, respectively, adopt GAN [16] and U-Net [18] to generate initial mask solution, our model L2O-ILT also shows superiority. The performance of L_2 achieves 33.5% and 28.3% enhancements, and PV Band could obtain 19.3% and 16.3% improvements. For the EPE count, the number of our EPE is only 2.60 on average, which is much smaller than GAN-OPC [6] (11.30) and DevelSet [9] (8.00). As for the runtime, our model is also faster than prior work. According to Table II, compared with ILT [4], GAN-OPC [6] and DevelSet [9], our L2O-ILT achieves 396.712×, 508.630× and 1.523× speedup, respectively.

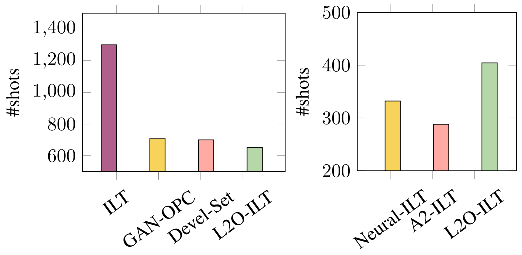
As for the other experimental results listed in Table III, when following the same settings as [7], L2O-ILT also achieves the best performance. Specifically, our model averagely outperforms Neural-ILT [7] with 33.3% and 28.6% reduction in L_2 error and PV Band. And the EPE count is only one-third of Neural-ILT [7]. Compared with A2-ILT [34], which relies on the reinforcement learning technique to improve the ILT performance, the L_2 and PV Band of our model are still reduced by around 30.1% and 26.6%. Also, our average EPE count is 2.50, much smaller than the EPE count of A2-ILT [34]. For the total runtime, L2O-ILT

TABLE II
MASK PRINTABILITY AND RUNTIME COMPARISON WITH STATE-OF-THE-ART METHODS (EXPERIMENTAL SETTINGS FOLLOW [4])

Bench	ILT [4]				GAN-OPC [6]				DevelSet [9]				L2O-ILT			
	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)
case1	6	49893	65534	318	8	52570	56267	358	10	49142	59607	1.50	3	39742	50432	0.73
case2	10	50369	48230	256	13	42253	50822	368	1	34489	52010	1.40	0	31550	42620	0.72
case3	59	81007	108608	321	51	83663	94498	368	64	93498	76558	1.29	22	67612	73850	0.76
case4	1	20044	28285	322	2	19965	28957	377	2	18682	29047	1.65	1	12550	20306	0.72
case5	6	44656	58835	315	8	44733	59328	369	1	44256	58085	0.91	0	34056	50982	0.72
case6	1	57375	48739	314	12	46062	52845	364	2	41730	53410	0.84	0	31830	47237	0.73
case7	0	37221	43490	239	7	26438	47981	377	0	25797	46606	0.76	0	20443	37207	0.75
case8	2	19782	22846	258	0	17690	23564	383	0	15460	24836	1.14	0	13429	19702	0.74
case9	6	55399	66331	322	12	56125	65417	383	0	50834	64950	1.21	0	39652	58708	0.72
case10	0	24381	18097	231	0	9990	19893	366	0	10140	21619	0.42	0	8363	17561	0.71
Average	9.10	44012.70	50899.50	289.60	11.30	39948.90	49957.20	371.30	8.00	38402.80	48672.80	1.11	**2.60**	**29922.70**	**41860.50**	**0.73**
Ratio	3.500	1.471	1.216	396.712	4.346	1.335	1.193	508.630	3.077	1.283	1.163	1.523	**1.000**	**1.000**	**1.000**	**1.000**

TABLE III
MASK PRINTABILITY AND RUNTIME COMPARISON WITH STATE-OF-THE-ART METHODS (EXPERIMENTAL SETTINGS FOLLOW [7])

Bench	Neural-ILT [7]				A2-ILT [34]				L2O-ILT			
	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)	EPE	L_2 (nm^2)	PVB (nm^2)	TAT (s)
case1	8	49817	55975	13.96	7	45824	59136	4.43	3	39636	46905	1.12
case2	3	38174	52010	15.87	3	33976	52054	4.48	0	29108	37099	1.11
case3	52	89411	91357	12.95	62	94634	82661	4.52	21	67263	69115	1.13
case4	2	16744	29982	9.53	2	20405	29435	4.44	1	10807	20694	1.12
case5	3	45598	58900	8.43	1	37038	62068	4.47	0	31909	48797	1.10
case6	5	43836	54969	8.50	2	40701	54842	4.44	0	31474	45453	1.14
case7	0	20324	50542	13.09	0	21840	48474	4.42	0	16942	35942	1.11
case8	0	13337	26353	12.94	0	14912	24598	4.47	0	12236	19496	1.13
case9	2	49401	68817	12.95	2	47489	68056	4.50	0	34849	56706	1.11
case10	0	8511	20734	11.66	0	9399	20243	4.35	0	7203	15976	1.11
Average	7.50	37515.30	50963.90	11.99	7.90	36621.80	50156.70	4.45	**2.50**	**28142.70**	**39618.30**	**1.12**
Ratio	3.000	1.333	1.286	10.705	3.160	1.301	1.266	4.045	**1.000**	**1.000**	**1.000**	**1.000**



(a) (b)

Fig. 11. Comparison of the mask manufacturability with state-of-the-art methods.

achieves 11.091× and 4.038× speedup in comparison with Neural-ILT [7] and A2-ILT [34].

Besides, we also evaluate the mask manufacturability in terms of the shot count, which stands for the number of rectangles that are used to approximate the optimized mask patterns. The comparison results are listed in Fig. 11(a) and (b). Among the above-mentioned methods, the shot number of L2O-ILT is reduced by 99.2%, 8.3%, and 7.2% compared with ILT [4], GAN-OPC [6], and DevelSet [9]. Although the masks generated by L2O-ILT contain 21.7% and 28.8% more shots than Neural-ILT [7] and A2-ILT [34], the mask printability and runtime performance is much better as listed in Table V. The quality and simplicity of the mask make a tradeoff, and we think it is acceptable to remarkably improve the mask printability within less runtime at the cost of a little bit higher complexity.

In addition, the memory usage of L2O-ILT is also compared versus other state-of-the-art methods, and the comparison results are listed in Table IV. It can be seen that our proposed model requires 7.4 GB GPU memory, which is comparable with other state-of-the-art methods. This also indicates that we successfully incorporate the prior knowledge of ILT into the deep learning-based model while not remarkably increasing the complexity of the model.

TABLE IV
MEMORY USAGE COMPARISON WITH STATE-OF-THE-ART METHODS

	Memory Usage(GB)
GAN-OPC [7]	6.5
DevelSet [9]	8.0
Neural-ILT [7]	6.6
A2-ILT [34]	5.7
L2O-ILT	7.4

B. Evaluation of Initial Mask Qualities

To prove the benefit of our model that high-quality mask solutions can be generated by L2O-ILT, we compare the L_2 error of the initial solutions with other "generative ILT" methods. Also, considering different experimental settings, we split the result comparison into two groups, as shown in Fig. 12(a) and (b). The average L_2 error of our initial masks is much lower than the initial masks of GAN-OPC [6], DevelSet [9], and Neural-ILT [7]. (A2-ILT [34] is not considered as "generative ILT" since it does not adopt a generation model). Combined with the results in Tables II and III, we can observe that for these "generative ILT" methods, there exists a large gap between the performance of the initial mask and the final result. Therefore,

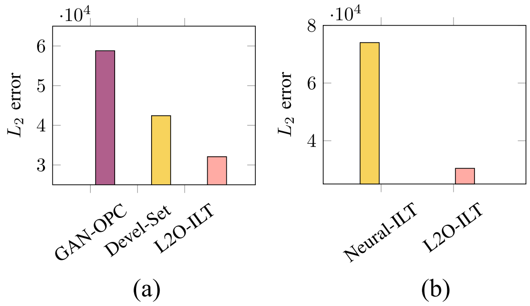


Fig. 12. Comparison of the initial mask solution with GAN-OPC [6] and DevelSet [9], and with Neural-ILT [7].

TABLE V
NEURAL-ILT VERSUS NEURAL-L2O-ILT (NEURAL-L2O-ILT IS TO ADOPT L2O-ILT TO REFINE THE INITIAL MASK GENERATED BY NEURAL-ILT)

	EPE	$L_2(nm^2)$	PVBand(nm^2)	TAT(s)
Neural-ILT [7]	9.20	38567.90	50636.70	11.80
Neural-L2O-ILT	3.00	30412.70	39626.70	1.81

a long-time refinement is always required. As for the initial solutions of L2O-ILT, the performance gap is really small, and thus the refinement can be finished rapidly, i.e., within 20 iterations, which contributes to the remarkable runtime improvement.

C. L2O-ILT Acts as Plugin

Another benefit of L2O-ILT is that our model can be incorporated into other models like Neural-ILT [7] and GAN-OPC [6] as a plugin, which can improve their mask optimization performance. We take the Neural-ILT [7] as an example. Given an initial mask solution generated by Neural-ILT [7], the original refinement process in Neural-ILT [7] has to finetune the entire model, including the U-Net [18], which contains a lot of parameters. Therefore, the refinement process is not efficient.

An improved method is to combine L2O-ILT with Neural-ILT [7] by directly feeding the low-quality mask into our model. As explained in Section III-D, given an initial solution and a test target pattern, the refinement process is achieved by tuning the last layer of our model, which includes fewer parameters than Neural-ILT [7]. Therefore, the refinement process is much more efficient to execute.

To verify the effectiveness of L2O-ILT as a plugin, we conduct further experiments using the ICCAD 2013 CAD benchmark [20]. We list the average performance of the mask refined by our L2O-ILT along with the required runtime in Table V, where we use Neural-L2O-ILT to denote the combination of Neural-ILT [7] and L2O-ILT. It can be seen that in spite of the low-quality initial mask, with the L2O-ILT, the mask generated by Neural-ILT [7] can still be more efficiently refined and even achieve better results in comparison with original Neural-ILT [7]. Note that the "TAT" of Neural-L2O-ILT has considered the generation runtime of the initial mask.

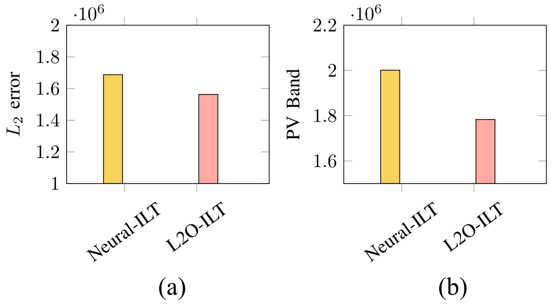


Fig. 13. Comparison of the full-chip mask optimization performance with Neural-ILT [7].

D. Evaluation on Full Chip

We also evaluate the performance of L2O-ILT on a large-scale chip of size 144 μm^2 using the pipeline illustrated in Section III-F. In our experiment, the large-scale chip is divided into smaller chips of size 2048 × 2048 and the size of the core region is set as 1024 × 1024. Such a setting effectively reduces the distortion effect while achieving satisfactory runtime performance.

We compare the performance of our proposed L2O-ILT and Neural-ILT [7] method in terms of L_2 and PV Band metrics. When evaluating the performance of Neural-ILT on the full-chip, we directly replace L2O-ILT in the pipeline shown in Fig. 10 with Neural-ILT. The presented results demonstrate that L2O-ILT achieves a reduction of 8.0% and 12.2% in L_2 error and PV Band, respectively. These results illustrate the benefit of L2O-ILT for full-chip mask optimization.

V. CONCLUSION

In this work, we present L2O-ILT, a deep learning based-model that achieves mask optimization acceleration and keeps remarkable printability performance. Our model structure is implemented by unrolling our ILT algorithm, and thus the model structure is highly incorporated into prior knowledge of mask optimization. Such an ILT algorithm-inspired model is able to generate an initial mask solution with better performance than previous methods, and the high-quality initial mask can be instantly refined to obtain the final solution. The experimental results demonstrate the superiority of our framework over current ILT acceleration works on both accuracy and efficiency.

REFERENCES

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

[2] 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.

[3] T. Matsunawa, B. Yu, and D. Z. Pan, "Optical proximity correction with hierarchical bayes model," *J. Micro/Nanolithography, MEMS, MOEMS*, vol. 15, no. 2, 2016, Art. no. 21009.

[4] 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 Des. Automa. Conf. (DAC)*, 2014, pp. 1–6.

[5] T. Cecil, K. Braam, A. Omran, A. Poonawala, J. Shu, and C. Vandam, "Establishing fast, practical, full-chip ILT flows using machine learning," in *Proc. SPIE*, 2020, Art. no. 1132706.

[6] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Y. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2822–2834, Oct. 2020.

[7] 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.

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

[9] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "DevelSet: Deep neural level set for instant mask optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–9.

[10] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2017, pp. 81–88.

[11] 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.

[12] 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.

[13] K. Hooker, B. Kuechler, A. Kazarian, G. Xiao, and K. Lucas, "ILT optimization of EUV masks for sub-7nm lithography," in *Proc. SPIE*, 2017, pp. 9–20.

[14] R. Pearman et al., "How curvilinear mask patterning will enhance the EUV process window: A study using rigorous wafer+ mask dual simulation," in *Proc. SPIE*, 2019, pp. 59–67.

[15] 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.

[16] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[17] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784.*

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

[19] H. Huang et al., "UNet 3+: A full-scale connected unet for medical image segmentation," in *Proc. ICASSP IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 1055–1059.

[20] 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.

[21] X. Ma, Q. Zhao, H. Zhang, Z. Wang, and G. R. Arce, "Model-driven convolution neural network for inverse lithography," *Opt. Exp.*, vol. 26, no. 5, pp. 32565–32584, 2018.

[22] X. Zheng, X. Ma, Q. Zhao, Y. Pan, and G. R. Arce, "Model-informed deep learning for computational lithography with partially coherent illumination," *Opt. Exp.*, vol. 28, no. 26, pp. 39475–39491, 2020.

[23] A. Poonawala and P. Milanfar, "OPC and PSM design using inverse lithography: A nonlinear optimization approach," in *Proc. Opt. Microlithogr. 19th*, 2006, pp. 1159–1172.

[24] X. Ma and G. R. Arce, "Generalized inverse lithography methods for phase-shifting mask design," *Opt. Exp.*, vol. 15, no. 23, pp. 15066–15079, 2007.

[25] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 399–406.

[26] T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin, "Learning to optimize: A primer and a benchmark," 2021, *arXiv:2103.12828.*

[27] H. Vu, G. Cheung, and Y. C. Eldar, "Unrolling of deep graph total variation for image denoising," in *Proc. ICASSP IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2021, pp. 2050–2054.

[28] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[29] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Trans. Signal Process.*, vol. 69, pp. 3699–3713, Jun. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9453145

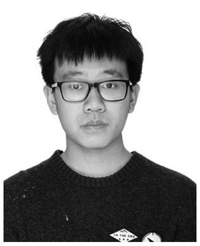
[30] H. H. Hopkins, "The concept of partial coherence in optics," in *Proc. Royal Soc. A. Math. Physic. Sci.*, 1951 pp. 263–277.

[31] W. Zhao et al., "AdaOPC: A self-adaptive mask optimization framework for real design patterns," in *Proc. 41st IEEE/ACM Int. Conf. Comput.-Aided Design*, 2022, pp. 1–9.

[32] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.

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

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



Binwu Zhu received the B.Eng. degree in information engineering from Zhejiang University, Hangzhou, China, in 2020. 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 interest includes machine learning for EDA.

Su Zheng received the B.Eng. and M.S. degrees from Fudan University, Shanghai, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. B. Yu and Prof. M. D. F. Wong.

His research interest is to solve critical problems in electronic design automation with advanced artificial intelligence methods.

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 with 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 machine learning in VLSI design for manufacturability and physics-informed networks for solving EDA area problems.

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, Chinese University of Hong Kong, Hong Kong, in 2020.

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

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

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.

From 2008 to 2011, he was an Assistant Professor with Fudan University. He is currently a Professor with the Microelectronics Department, Fudan University. His research interests include model order reduction, circuit simulation, high-level synthesis, 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.

Prof. Yu received the 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 has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of IEEE TCCPS Newsletter.

Martin D. F. Wong received the B.Sc. degree in mathematics from the University of Toronto, Toronto, ON, Canada, in 1979, and the M.S. degree in mathematics and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign (UIUC), Champaign, IL, USA, in 1981 and 1987, respectively.

He was a Faculty with The University of Texas at Austin (UT-Austin), Austin, TX, USA, from 1987 to 2002 and UIUC from 2002 to 2018. He was a Bruton Centennial Professor of Computer Science with UT-Austin and an Edward C. Jordan Professor of Electronics and Communication Engineering with UIUC. From August 2012 to December 2018, he was the Executive Associate Dean of the College of Engineering, UIUC. Since January 2019, he has been joined The Chinese University of Hong Kong, Hong Kong, as the Dean of Engineering and Choh-Ming Li Professor of Computer Science and Engineering. He has published around 500 papers and graduated over 50 Ph.D. students in Electronic Design Automation (EDA). His main research interest is in EDA.

Prof. Wong is a Fellow of ACM.