Efficient ILT via Multi-level Lithography Simulation

Shuyuan Sun[1,2], Fan Yang[1,2*], Bei Yu[4], Li Shang[1,3], Xuan Zeng[1,2*]
[1]State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai, China
[2] School of Microelectronics, Fudan University, Shanghai, China
[3]China and Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
[4]Chinese University of Hong Kong, Hong Kong, China

***Abstract*—Inverse Lithography Technology (ILT) is a widely investigated method to improve the yield of chip manufacturing. However, high computational complexity and difficulty in fabricating curvilinear shapes have hindered the widespread adoption of ILT. This paper presents an efficient ILT framework, including a multi-level resolution method for simulation acceleration, a downsampling strategy for mask optimization, and an improved mask binary function to improve mask printability. Experimental results show that the proposed method outperforms state-of-the-art methods with at least a 33.8% reduction in L2 loss and a 15.5% reduction in PVBand.**

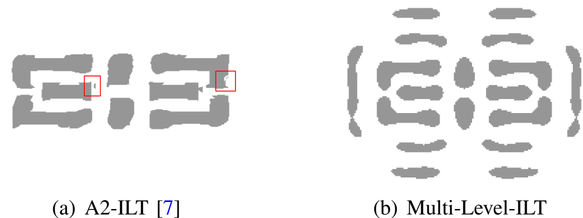
I. Introduction

As technology node continues to shrink, the mismatch between printed wafer shapes and designed graphics become more and more severe. At advanced technology nodes, inverse lithography technology (ILT) and model-based optical proximity correction (OPC) are two major resolution enhancement techniques (RETs) that used to improve the printability of masks. Compared to model-based OPC, ILT offers greater flexibility and produces masks with better printability and smaller process variation bands (PVBand). For instance, [1] and [2] are two representative works of ILT and model-based OPC, showing that ILT outperforms model-based OPC on the printability score tested using the same dataset. However, high runtime overhead and poor mask manufacturability are two major obstacles preventing the widespread adoption of ILT.

To reduce the runtime overhead of ILT, deep learning methods have recently been introduced to reduce the number of iterations by predicting an initial mask solution for the mask optimization problem [3]–[5]. In [3], a generative adversarial network (GAN) based method is proposed to predict optimized mask shapes. In [4], a neural network-based approach is proposed to support mask prediction, ILT correction, and shape regularization. In [5], a deep model is proposed for mask prediction, and then the predicted mask is further optimized using the GPU-accelerated level set-based method [6]. 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 such an initial mask will be further optimized by subsequent optimization iterations of ILT, since ILT is very sensitive to the initial solution, the mask quality may be much lower than that produced by the original ILT.

Recent efforts have also focused on pattern simplification in order to improve mask manufacturability. In [4], a loss term to evaluate mask complexity is introduced to eliminate small shapes around major features. In [5], a curvature penalty term is introduced in the loss function, aiming to obtain smoother shape boundaries. In [7], a spatial attention map is introduced to avoid holes and outliers on the optimized layout. The work [7] and [8] apply an average pooling operation 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 (as shown in Fig. 1(a)).

*Corresponding authors: {yangfan, xzeng}@fudan.edu.cn.



(a) A2-ILT [7] (b) Multi-Level-ILT

Fig. 1: Optimized mask outputs.

In summary, mask generation with high efficiency and good manufacturability remains the central goal of ILT after years of effort. To this end, this work presents an ILT framework, aiming to tackle the aforementioned challenges. The proposed ILT framework consists of a multi-level resolution method for simulation acceleration, a downsampling strategy for mask optimization, and an improved mask binary function to improve mask printability, specifically:

Runtime efficiency: Inspired by the multigrid method, we propose a multi-resolution lithography simulation scheme, starting with coarse-grain simulations to optimize computation efficiency, followed by high-precision lithography simulations to optimize accuracy. The proposed coarse-to-fine simulation scheme can accelerate ILT in various situations without model pretraining. We perform ILT across multiple resolution scale factors to speed up the mask optimization process. Experimental results show that with a scale factor of 4, low-resolution ILT is about 18$\times$ faster than high-resolution ILT. High-resolution ILT optimizes mask under accurate lithography simulation. Since the mask is iteratively optimized using ILT, the proposed method does not degrade the mask quality.

Manufacturability & printability: We employ an average pooling operation on neighboring pixels to improve the smoothness of shape boundaries. Similar to [7], we adopt high-resolution downsampling to avoid generating unwanted miniature shapes. As for printability, we propose an improved mask binary function to optimize mask printability. The improved binary function makes it easier for ILT to obtain Sub-Resolution Assist Features (SRAF), which improves the quality of the mask.

Together, the proposed ILT framework offers high computa-

tion efficiency and produces masks of good manufacturability and printability. On the ICCAD 2013 contest benchmark [9], compared to state-of-art methods, our approach can reduce at least 33.8% in L_2 loss and 15.5% in PVBand. Fig. 1 compares the masks obtained by [7] and our method. It is shown that masks obtained by our approach have smoother and more regular shape boundaries. By employing a multi-resolution simulation scheme, our runtime of ILT iterations can be reduced by more than two times.

The rest of the paper is organized as follows. In Section II, we will present the background and evaluation metrics used in ILT. In Section III, we propose the multi-level ILT approach. In Section IV, experimental results are presented. In Section V, we conclude the paper.

II. Background

A. Forward Lithography Model

The forward lithography model is used to describe the mapping from the designed mask image $\boldsymbol{M} \in \mathbb{R}^{N\times N}$ to the wafer image $\boldsymbol{Z} \in \mathbb{R}^{N\times N}$. The wafer image $\boldsymbol{Z}$ is obtained from the aerial image $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ passed through the photoresist model. The aerial image $\boldsymbol{I}$ represents the distribution of lithography intensities. For simplicity, we use a compact photoresist model with a constant threshold I_{th} as shown in Equation (1).

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

The transformation from mask image $\boldsymbol{M}$ to aerial image $\boldsymbol{I}$ can be approximated by the Hopkin's model [10] as shown in Equation (2).

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

where $\boldsymbol{h}_k \in \mathbb{R}^{P\times P}$ represents the kth optical kernel, and w_k is the corresponding weight. N_k is the total number of kernels used for simulation. $\otimes$ denotes the convolution operation. The complexity of Equation (2) is $O(N_k \cdot P^2 \cdot N^2)$. It is proved that convolution in the spatial domain is equivalent to multiplication in the frequency domain. Therefore, computational efficiency can be further improved by using the fast Fourier transformation (FFT), as shown in Equation (3).

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

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent FFT and inverse FFT, respectively. $\boldsymbol{H}_k \in \mathbb{C}^{P\times P}$ is the representation of $\boldsymbol{h}_k$ in the frequency domain. $\odot$ denotes the element-wise multiplication. Note that we discard the high-frequency part of $\mathcal{F}(M)$ so that it can be multiplied by $\boldsymbol{H}_k$, and the complexity of $\boldsymbol{H}_k \odot F(\boldsymbol{M})$ is $O(P^2)$. To restore to the original size, the dimension of the inverse FFT is N^2, and its complexity is $O(N^2 \log N^2)$. The complexity of Equation (3) is $O\left((N_k+1)\cdot N^2 \log N^2 + N_k \cdot P^2\right)$. In our implementation, $P = 35$, $N = 2048$, and $N_k = 24$. Note that $P \ll N$, thus the largest part of computation lies on N_k inverse FFTs.

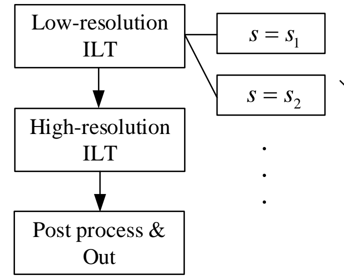


Fig. 2: The complete framework of multi-level ILT.

B. Evaluation Metrics and Problem Formation

***Definition** 1 (Squared L_2 Loss):* Given the target image $\boldsymbol{Z}_t$ and the wafer image $\boldsymbol{Z}_{norm}$, the squared L_2 loss is equal to $\|\boldsymbol{Z}_{norm} - \boldsymbol{Z}_t\|_2^2$. Note that the wafer image $\boldsymbol{Z}_{norm}$ is obtained under nominal dose and nominal focus.

***Definition** 2 (PVBand):* Process variation band denotes the XOR area between two aerial images $\boldsymbol{Z}_{in}$ and $\boldsymbol{Z}_{out}$ generated under different conditions.

We adopt the same settings as [9]: $\boldsymbol{Z}_{in}$ is generated under the defocus and -2% dose condition. $\boldsymbol{Z}_{out}$ is generated under nominal focus and $+2\%$ dose condition.

***Definition** 3 (Edge Placement Error):* Edge placement Error (EPE) is a metric used to evaluate how much the wafer image is distorted from the target image. The measurement points are evenly distributed along the horizontal and vertical contours of the shape on the target image. If the vertical distance $D(x,y)$ of the printed contour to the target one is greater than threshold distance thr, it is considered as an EPE violation.

Equation (4) shows an example of EPE violation. We set the threshold distance thr to $15nm$, following the setting of [9].

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

***Definition** 4 (Mask Fracturing Shot Count):* Mask fracturing shot count is the number of rectangles that used to replicate the optimized curvilinear mask shapes [4]. It is a metric used to evaluate 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 acceptable runtime, hoping to have improvements in terms of printability and manufacturability of masks.

III. Algorithms

A. Overall Framework

The overall framework of multi-level ILT is illustrated in this section and shown in Fig. 2. We first perform low-resolution ILT with different scale factors s_i from large to small to obtain computational efficiency. Then, high-resolution ILT is employed to fine-tune the layout and reduce mask complexity.

We detail the high- and low-resolution ILT in Section III-B. In Section III-C, we give a modified binary function to improve the printability of obtained masks. To obtain a more neat layout of the mask, we introduce the average pooling in Section III-D to smooth the contours of shapes. Note that this smoothing operation is only adopted by low-resolution ILTs.

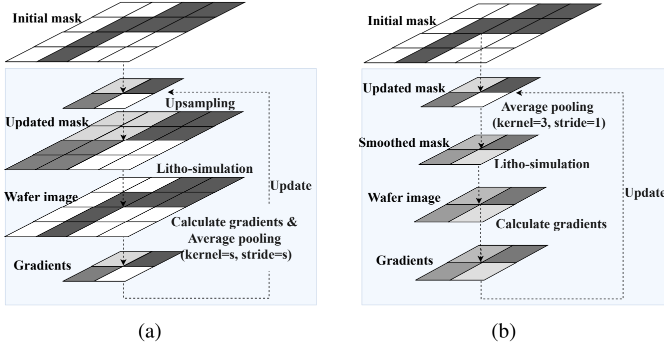


Fig. 3: The illustration of (a) high-resolution ILT and (b) low-resolution ILT.

B. Multi-level ILT

First, we give the definition of loss function L used in our method, as shown in Equation (5). It consists of two parts, the squared L_2 loss L_{l_2} and PVBand L_{pvb}. We replace the $\boldsymbol{Z}_{norm}$ with $\boldsymbol{Z}_{out}$ in the calculation of L_{l_2} to save computation time and GPU memory during the optimization process.

$$\begin{aligned} L &= L_{l_2} + L_{pvb}, \\ L_{l_2} &= \left\| \boldsymbol{Z}_{out} - \boldsymbol{Z}_t \right\|_2^2, \\ L_{pvb} &= \left\| \boldsymbol{Z}_{in} - \boldsymbol{Z}_{out} \right\|_2^2. \end{aligned} \tag{5}$$

We then combine low-resolution and high-resolution ILT in the proposed framework to achieve computational efficiency and good mask manufacturability, respectively. High-resolution ILT has the same computational complexity as conventional ILT, but it produces simpler mask patterns. It performs average pooling on the obtained wafer image to shrink it without losing information (line 9 in Algorithm 1). The downscaled mask and wafer images are then fed into the loss function, and gradients are computed to update the mask image (lines 14–15 in Algorithm 1). $\boldsymbol{M}_s'$ and $\boldsymbol{Z}_s$ denote the reduced mask and wafer images, respectively, with a scale factor s. Then, we restore the mask $\boldsymbol{M}_s$ to its original size to perform an accurate lithography simulation (line 7 in Algorithm 1). The computation flow of high-resolution ILT is shown in Fig. 3(a).

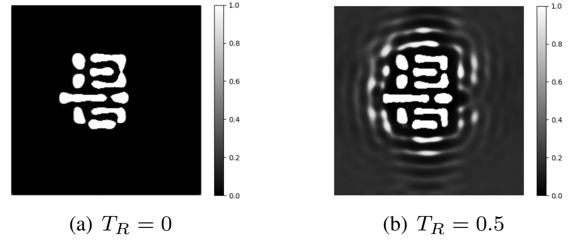
For the low-resolution ILT, lithography simulations are performed every s pixel in both X and Y directions. It reduces the dimensionality of the simulation and saves time considerably with an acceptable loss of accuracy. Since both the simulation and the optimization are carried out at a reduced size (lines 12 and 15 in Algorithm 1), there are no upsampling and downsampling pairs in the low-resolution ILT. Litho$(\cdot)$ represents the optical model described in Equation (3) and Equation (1). The Low-resolution simulation is stated in Equation (6), where $\mathcal{F}_N$ represents an FFT with transform dimension equal to N.

$$\boldsymbol{I}(sx, sy) = \sum_{k=1}^{N_k} w_k \cdot \boldsymbol{I}_k = \sum_{k=1}^{N_k} w_k \left\| \mathcal{F}_N^{-1} \left(\boldsymbol{H}_k \odot \mathcal{F}_N (\boldsymbol{M}) \right) (sx, sy) \right\|^2. \tag{6}$$

In Equation (6), there are many wasted computations in the scaled-down intensity image. The mathematical theorem states that the reduction of the spatial domain is equivalent to the expansion of the frequency domain. Therefore, the computation of $\boldsymbol{I}_k$ in Equation (6) can be replaced by Equation (7) to avoid redundant computations. Note that the transform dimension of the inverse FFT is $s\times$ smaller in Equation (7) than in Equation (6). Applying Equation (7) to obtain the intensity images can greatly improve efficiency since most of the computation is spent on the N_k inverse FFT.

Algorithm 1 Multi-level ILT

1: **Input :** Target image $\boldsymbol{Z}_t \in \mathbb{R}^{N \times N}$; scale factor s; N_k kernels $\boldsymbol{H}_k$ and corresponding weights w_k; learning rate lr; $flag$ indicates the type of ILT used;
2: $\boldsymbol{Z}_{t,s} \leftarrow \text{AvgPool}\left(\boldsymbol{Z}_t, \text{kernel_size} = s, \text{stride} = s\right)$;
3: $\boldsymbol{M}_s' \leftarrow \boldsymbol{Z}_{t,s}$;
4: **repeat**
5: $\quad \boldsymbol{M}_s = f_{binary}\left(\boldsymbol{M}_s'\right)$;
6: $\quad$ **if** $flag == 1$ **then** ▷ high-resolution ILT
7: $\quad\quad \boldsymbol{M}(x, y) = Upsample\left(\boldsymbol{M}_s\right)$;
8: $\quad\quad \boldsymbol{Z}(x, y) = \text{Litho}\left(\boldsymbol{M}, \boldsymbol{H}_k, w_k, I_{th}\right)$;
9: $\quad\quad \boldsymbol{Z}_s = \text{AvgPool}\left(\boldsymbol{Z}, \text{kernel_size} = s, \text{stride} = s\right)$;
10: $\quad$ **else** ▷ low-resolution ILT
11: $\quad\quad \boldsymbol{M}_s = \text{AvgPool}\left(\boldsymbol{M}_s, \text{kernel_size} = 3, \text{stride} = 1\right)$;
12: $\quad\quad \boldsymbol{Z}_s(x, y) = \text{Litho}\left(\boldsymbol{M}_s, \boldsymbol{H}_k, w_k, I_{th}\right)$;
13: $\quad$ **end if**
14: $\quad \boldsymbol{G}_s = \frac{\partial L(\boldsymbol{Z}_{t,s}, \boldsymbol{Z}_s)}{\partial \boldsymbol{M}_s'}$;
15: $\quad \boldsymbol{M}_s' = \boldsymbol{M}_s' - lr \cdot \boldsymbol{G}_s$;
16: **until** exit conditions are met.



(a) $T_R = 0$ (b) $T_R = 0.5$

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.

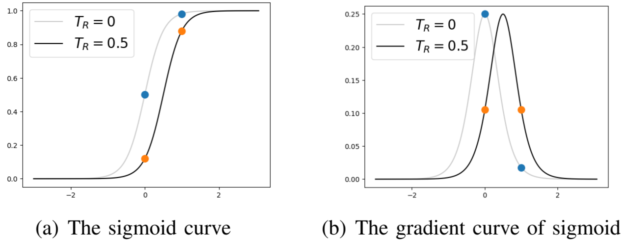
$$\begin{aligned} \boldsymbol{I}_k(sx, sy) &= \mathcal{F}_N^{-1}\left(\frac{1}{s^2}\left(\boldsymbol{H}_k \odot \mathcal{F}_N(\boldsymbol{M})\right)\left(\frac{x}{s}, \frac{y}{s}\right)\right) \\ &= \mathcal{F}_{N/s}^{-1}\left(\frac{1}{s^2}\left(\boldsymbol{H}_k \odot \mathcal{F}_N(\boldsymbol{M})\right)\right). \end{aligned} \tag{7}$$

In our implementation, we use Equation (8) to approximate Equation (7), and experimental results show that this approximation holds. The running time of 200 forward lithography simulations using Equation (3), Equation (7) and Equation (8) are 8.173, 0.767 and 0.466 seconds, respectively. The high- and low-resolution ILT are illustrated in Fig. 3. In Algorithm 1, the binary function f_{binary} (line 5) and the smoothing pooling (line 11) will be explained in detail in Section III-C and Section III-D.

$$\boldsymbol{I}_k \approx \mathcal{F}_{N/s}^{-1}\left(\boldsymbol{H}_k \odot \mathcal{F}_{N/s}(\boldsymbol{M}_s)\right). \tag{8}$$

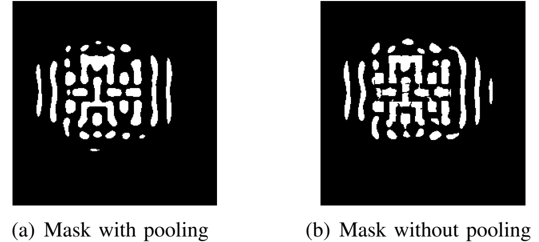
C. Binary Function

During ILT iterations, the mask and wafer images should be continuous-valued matrices, since gradients are required to update values of mask pixels. Therefore, we use Equation (9)



(a) The sigmoid curve (b) The gradient curve of sigmoid

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.



(a) Mask with pooling (b) Mask without pooling

Fig. 6: The 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.

instead of Equation (1) as the resist model during the ILT optimization.

$$\boldsymbol{Z} = \frac{1}{1 + \exp\left[-\alpha \times (\boldsymbol{I} - I_{th})\right]}, \tag{9}$$

where we take $I_{th} = 0.225$ according to settings in [9].

Various binary functions $f_{binary}(\cdot)$ have been studied to constrain the pixel values of the mask between 0 and 1. [11] takes the cosine transformation as its binary function, as shown in Equation (10).

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

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

Due to the periodicity of Equation (10), the initial learning rate for ILT using Equation (10) should be chosen carefully, otherwise the optimization will have difficulty converging. Therefore, [12] propose a monotonic sigmoid transformation as the binary function, as shown in Equation (11).

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

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 first iteration of ILT, values in the binarized mask $\boldsymbol{M}$ are $\{0.5, \approx 1\}$ while values in the original mask are $\{0, 1\}$. Due to the difference between the binarized mask and the original mask, ILT assigns a large negative value to the opaque areas of the optimized mask $\boldsymbol{M'}$ after the first iteration. It makes the generation of SRAFs on the opaque areas very difficult because the sigmoid function tends to fall into local optima. Fig. 4(a) shows the binarized mask $\boldsymbol{M}$ with $T_R = 0$ after 40 ILT iterations. The areas outside the main features are almost completely opaque and have no SRAFs. In our observations, there will be SRAFs after more iterations but this process is very slow.

Therefore, we take $T_R = 0.5$ so that values in the initial binarized mask $\boldsymbol{M}$ are $\{\approx 0.1, \approx 0.9\}$, which narrows the difference from the original mask and facilitate the generation of SRAFs. In Fig. 5 we plot the sigmoid transformation and its gradient. In Fig. 5(a), we denote the initial values of the binarized mask M using $T_R = 0$ and $T_R = 0.5$ in blue and orange, respectively. At the first iteration, for the sigmoid transformation using $T_R = 0$, $M(x, y)$ corresponding to $M'(x, y) = 0$ not only much deviates from the initial mask, but also its gradient value is the largest.

Fig. 4(b) shows the binarized mask using $T_R = 0.5$ with the same other settings as Fig. 4(a). The results show that using the improved binary function helps to obtain SRAF more easily. Previous works like [4] all make efforts to avoid small shapes close to main features, like shapes shown in the red box in Fig. 1(a). It is not contradictory to our proposal, because of downsampling and pooling operations, our generated SRAFs have simple shapes as shown in Fig. 1(b). SRAFs of simple shapes can improve the printability of the mask without degrading its manufacturability.

Since values in the binarized mask $\boldsymbol{M} \in (0, 1)$ are real numbers, another binarization is needed at last to generate the complete binarized mask $\boldsymbol{M}_{out}$. Equation (12) shows this binarization, which is adopted by most ILT methods [3]–[5], [13].

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

where t_m is the threshold value and set to 0.5.

In Fig. 4(b), pixels of main features are very close to 1 making them hardly change in later iterations. In our observations, the adjustments to SRAFs dominate subsequent iterations of ILT, but due to the small gradients obtained at each iteration, the SRAFs appeared on the final mask $\boldsymbol{M}_{out}$ does not many. To obtain more SRAFs, we use a smaller T_R for the final output. In our implementation, we use $T_R = 0.5$ for the optimization and $T_R = 0.4$ for the final output. The L_2 loss and PVBand listed in Fig. 4 follow the same settings.

D. Simplify Shapes via Average Pooling

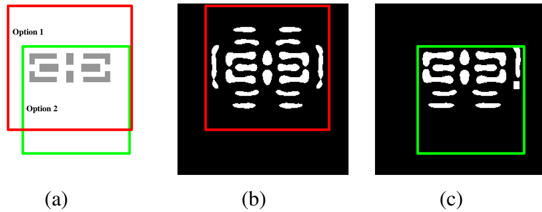
In this section, we introduce another pooling operation to smooth shape contours in the low-resolution ILT. This average pooling is applied on $n \times n$ adjacent pixels, but its stride size is not equal to n. For example, we take $n = 3$ and stride size equal to 1 to keep the input and output the same size (line 11 in Algorithm 1). It is different from the average pooling used in the high-resolution ILT Section III-B, which is used for downsampling.

By adopting this operation, each mask pixel can obtain average attention on its adjacent pixels and then make adjustments after considering their values. It enable each pixel to have an average attention on its neighbors, so their changes will be more continuous. In our proposed method, since we introduce more SRAF into the optimized mask, the smoothing operation is more needed to simplify graphics of masks.

We perform this pooling before binarizing the mask in each iteration of the low-resolution ILT, as shown in Fig. 3(b).

TABLE I: 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



(a) (b) (c)

Fig. 7: (a) Two options for optimizing regions; (b) the mask obtained using option 1; (c) the mask obtained using option 2.

Experimental results show that this pooling operation can effiently avoid holes and fractures on the resulting masks. Fig. 6(a) and Fig. 6(b) are mask images with and without the pooling operation, respectively. The mask without pooling has a smaller squared L_2 loss, but it has a higher shape complexity. For the optional post-processing, we eliminat too small shapes and replaced medium-sized irregular SRAFs with rectangles to further simplify the mask pattern.

IV. Experimental Results

The multi-level 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 [9] 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, process variation band, edge placement error, 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.

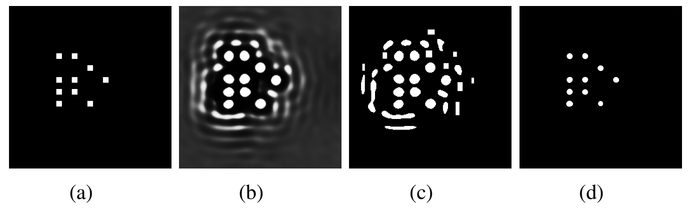
A. Ablation Study

To validate the effectiveness of downsampling methods, 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 [9] are listed in TABLE I. We set the scale factor for both low- and high-resolution downsampling equal to 4 and learning rate equal to 1.

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 $18\times$ speedup over high-resolution ILT and obtains masks of comparable quality, which proves its good computational efficiency.

B. Results on M1 Designs

In TABLE II and TABLE III, we compare our approach with SOTA methods on ten M1 designs of 32nm technology nodes provided by ICCAD 2013 benchmarks [9]. Columns “Neural-ILT [4]”, “A2-ILT [7]”, “GLS-ILT [6]” and “DevelSet [5]” denote the results of selected baseline ILT methods, respectively. All these methods limit mask optimization to a defined region, but they define the region differently. As shown in Fig. 7(a), Neural-ILT and A2-ILT adopt the option 1 while GLS-ILT and DevelSet adopt the option 2 as the optimizing region. For fairness of comparison, we follow their settings in TABLE II and TABLE III, respectively. Divergence in the optimizing regions has little effect on their results, as they produce few SRAFs, but affects the performance of our method due to the constraints on SRAFs. Fig. 7(b) and Fig. 7(c) are the masks obtained by “Our-exact” under these two area options, respectively.



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

Fig. 8: One example of via patterns: (a) Target mask; (b) Binarized mask; (c) Final mask; (d) Wafer image.

Column ”Our-fast” represents the result using 35 iterations of the low-resolution ILT with a scale factor of 4 plus 5 iterations of the high-resolution ILT with a scale factor of 8. Column ”Our-exact” represents the result using 80 iterations of the low-resolution ILT with a scale factor of 4 plus 10 iterations of the high-resolution ILT with a scale factor of 8. Note that the runtime of our method is consisted of ILT iteration time and post-processing time. The average runtime for ”Our-fast” in TABLE III includes an ILT time of 0.48 seconds and a post-processing time of 1.27 seconds. And most of post-processing time is spent on data transfer between GPU and CPU.

In TABLE II, we outperform A2-ILT by a 34.8% reduction in squared L_2 loss and a 25.8% reduction in process variation band. Compared with A2-ILT, although the “#shots” of our mask is larger, its graph is more regular as shown in Fig. 1. Compared with DevelSet, the exact version of multi-level ILT reduces “L2”, “PVBand” by 33.8% and 15.5%, respectively, and reduces “#shots” by $2.4\times$. The exact version of multi-level ILT has the fewest average EPE violations compared to other methods.

We further verify the scalability of our proposed method on ten additional testcases (as shown in TABLE IV) released by [4]. These ten cases contain more graphics than ICCAD 2013 benchmarks [9]. Compared with Neural-ILT, the exact version of multi-level ILT achieves a 40.3% reduction in L2 loss and a 24.3% reduction in PVBand, and a speedup of over $4.8\times$. And the fast version of multi-level ILT obtains masks with the smallest “#shots”.

C. Results on Via Designs

We randomly choose fifteen via patterns of 2048×2048 size from the dataset obtained from [14]. Via shapes are smaller than shapes on the M1 layer and require finer adjustments. Therefore, we perform 100, 100, and 50 low-resolution ILTs with scale factors of 8, 4, and 2 in sequence. Finally, 15 high-resolution ILTs with a scale factor of 8 are performed. Note that

TABLE II: Comparison on ICCAD 2013 benchmarks.

Benchmarks		Neural-ILT [4]					A2-ILT [7]					Our-fast					Our-exact				
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)
case1	215344	49817	55975	8	428	11	45824	59136	7	242	4.53	41919	47144	3	272	1.70	38495	47015	3	385	3.45
case2	169280	38174	52010	3	256	17	33976	52054	3	211	4.5	28904	37734	0	235	1.70	28173	37555	0	284	3.44
case3	213504	89411	91357	52	557	10	94634	82661	62	282	4.54	68975	68447	28	265	1.70	67949	69361	22	316	3.44
case4	82560	16744	29982	2	136	9	20405	29435	2	103	4.51	11387	22938	0	175	1.72	10307	21514	0	241	3.45
case5	281958	45598	58900	3	380	11	37038	62068	1	319	4.53	31442	51292	0	326	1.73	28482	49683	0	411	3.46
case6	286234	43836	54969	5	383	10	40701	54842	2	244	4.52	31963	46177	0	323	1.72	30334	44127	0	415	3.42
case7	229149	20324	50542	0	244	16	21840	48474	0	206	4.51	16772	41396	0	216	1.72	14635	36961	0	382	3.46
case8	128544	13337	26353	0	285	15	14912	24598	0	156	4.48	12747	20708	0	193	1.73	11194	20985	0	271	3.42
case9	317581	49401	68817	2	444	11	47489	68056	2	248	4.52	36988	57528	0	366	1.72	34900	54948	0	490	3.47
case10	102400	8511	20734	0	208	14	9399	20243	0	126	4.5	8248	17351	0	144	1.73	7266	16581	0	164	3.47
Average		37515.3	50963.9	7.5	332.1	12.4	36621.8	50156.7	7.9	**213.7**	4.51	28916.5	41144	3.1	251.5	**1.72**	**27173.5**	**39873**	**2.5**	335.9	3.45
Ratio		1.381	1.278	3.0	0.989	3.592	1.348	1.258	3.160	**0.636**	1.307	1.064	1.032	1.240	0.749	**0.497**	**1**	**1**	**1**	1	1

TABLE III: Comparison on ICCAD 2013 benchmarks.

Benchmarks	GLS-ILT [6]					DevelSet [5]					Our-fast					Our-exact				
	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)
case 1	46032	62693	4	1476	123	49142	59607	-	969	1.5	42503	49784	3	233	1.75	40779	50661	3	307	3.49
case 2	36177	50642	1	861	81	34489	52012	-	743	1.4	34693	43801	2	169	1.74	34201	44322	2	186	3.47
case 3	71178	100945	29	2811	214	93498	76558	-	889	1.29	69698	72255	29	246	1.76	66486	71527	22	308	3.47
case 4	16345	29831	0	432	184	18682	29047	-	376	1.65	11829	22716	0	176	1.75	10942	21500	0	233	3.47
case 5	47103	56328	1	963	76	44256	58085	-	902	0.91	35226	53649	0	268	1.75	30231	51277	0	374	3.47
case 6	46205	51033	1	942	65	41730	53410	-	774	0.84	33883	47716	0	302	1.75	30741	44982	0	365	3.47
case 7	28609	44953	0	548	64	25797	46606	-	527	0.76	21732	44725	0	142	1.73	17101	40294	0	196	3.50
case 8	19477	22541	1	439	67	15460	24836	-	493	1.14	13236	21178	0	158	1.77	11935	20357	0	243	3.47
case 9	52613	62568	0	881	63	50834	64950	-	932	1.21	38781	58845	0	327	1.75	35805	57930	0	435	3.50
case 10	22415	18769	0	333	64	10140	21619	-	393	0.42	11122	19106	0	90	1.75	8825	18470	0	114	3.48
Average	38615.4	50030.3	3.7	968.6	100.1	38402.8	48673	-	699.8	**1.112**	31270.3	43377.5	3.4	**211.1**	1.75	**28704.6**	**42132**	**2.7**	286.1	3.48
Ratio	1.345	1.187	1.370	3.386	28.764	1.338	1.155	-	2.446	**0.320**	1.089	1.030	1.259	**0.738**	0.503	**1**	**1**	**1**	1	1

TABLE IV: Comparison on larger benchmarks.

Benchmarks		Neural-ILT [4]					Our-fast					Our-exact				
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	#shots	TAT (s)
case11	494560	79933	120577	12	669	20	64345	93486	3	534	1.70	61534	94116	4	628	3.48
case12	448496	86995	104266	15	556	12	53402	86606	0	443	1.72	50037	84984	0	537	3.46
case13	492720	133281	152718	70	766	15	98597	118403	29	536	1.69	94496	120889	26	610	3.49
case14	361776	43797	92137	0	455	14	36101	69043	2	415	1.70	32478	68470	1	504	3.47
case15	561174	69521	122115	3	808	19	59208	99443	0	475	1.70	55936	101929	0	544	3.46
case16	565450	73790	117359	2	764	19	63194	96831	0	485	1.69	57169	95182	0	557	3.45
case17	445365	49031	92320	0	531	19	36329	79834	0	424	1.69	32709	75742	0	513	3.45
case18	407760	47409	84971	0	478	16	36753	66672	0	434	1.70	33981	67838	0	511	3.48
case19	596797	93922	115028	5	614	14	68550	110297	0	508	1.71	61824	107744	0	567	3.48
case20	381616	28028	80127	0	452	19	31816	63866	0	382	1.71	30118	63327	0	387	3.46
Average		71570.7	108162	10.7	609.3	16.7	54829.5	88448.1	3.4	**463.6**	**1.70**	**51028.2**	**88022.1**	**3.1**	535.8	3.47
Ratio		1.403	1.229	3.452	1.137	4.817	1.074	1.005	1.097	**0.865**	**0.491**	**1**	**1**	**1**	1	1

the number we set is only an upper bound of iterations. We exit early when ILT cannot obtain a new minimum loss within 15 iterations. In Fig. 8, we show the target image, binarization mask, wafer image, and mask image for one case, which is the worst of fifteen randomly selected cases. Although its L2 and PVBand are the largest among 15 cases, all its via shapes are printed on the wafer image as shown in Fig. 8(d).

V. Conclusion

In this paper, we propose an ILT method based on multilevel lithography simulation with an improved binary function and a simple but effective smoothing operation. The multi-level ILT framework greatly reduces the running time of ILT and the complexity of masks, which are two obstacles preventing ILT from being widely used. Experimental results show that the proposed approach outperforms the SOTA method on the ICCAD 2013 benchmark in mask printability and with superior runtime performance. Furthermore, our method also shows its adaptability and advantages on via patterns.

Acknowledgement

This research is supported partly by National Key R&D Program of China 2020YFA0711900, 2020YFA0711903, partly by National Natural Science Foundation of China (NSFC) research projects 62090025, 62141407 and 61929102, and The Research Grants Council of Hong Kong SAR (No. CUHK14208021).

References

[1] Jhih-Rong Gao and etc. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *Proc. DAC*, pages 1–6. IEEE, 2014.

[2] Jian Kuang and etc. A robust approach for process variation aware mask optimization. In *Proc. DATE*, pages 1591–1594. IEEE, 2015.

[3] Haoyu Yang and etc. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. *IEEE TCAD*, 39(10):2822–2834, 2019.

[4] Bentian Jiang and etc. Neural-ILT 2.0: Migrating ILT to Domain-specific and Multi-task-enabled Neural Network. *IEEE TCAD*, 2021.

[5] Guojin Chen and etc. DevelSet: Deep neural level set for instant mask optimization. In *Proc. ICCAD*, pages 1–9. IEEE, 2021.

[6] Ziyang Yu and etc. A gpu-enabled level set method for mask optimization. In *Proc. DATE*, pages 1835–1838. IEEE, 2021.

[7] Qijing Wang and etc. A2-ILT: GPU accelerated ILT with spatial attention mechanism. In *Proc. DAC*, pages 967–972, 2022.

[8] Xu Ma and etc. Fast optical proximity correction method based on nonlinear compressive sensing. *Optics Express*, 26(11):14479–14498, 2018.

[9] Shayak Banerjee and etc. Iccad-2013 cad contest in mask optimization and benchmark suite. In *Proc. ICCAD*, pages 271–274. IEEE, 2013.

[10] Harold Horace Hopkins. The concept of partial coherence in optics. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 208(1093):263–277, 1951.

[11] Amyn Poonawala and Peyman Milanfar. Mask design for optical microlithography—an inverse imaging problem. *IEEE TIP*, 16(3):774–788, 2007.

[12] Xin Zhao and Chris Chu. Line search-based inverse lithography technique for mask design. *VLSI Design*, 2012.

[13] Guojin Chen and etc. DAMO: Deep agile mask optimization for full chip scale. In *Proc. ICCAD*, pages 1–9, 2020.

[14] Hao Geng and etc. Hotspot detection via attention-based deep layout metric learning. In *Proc. ICCAD*, pages 1–8, 2020.