A2-ILT: GPU Accelerated ILT with Spatial Attention Mechanism

Qijing Wang
CSE Department, CUHK
qjwang21@cse.cuhk.edu.hk

Bentian Jiang
CSE Department, CUHK
btjiang@cse.cuhk.edu.hk

Martin D.F. Wong
CSE Department, CUHK
mdfwong@cuhk.edu.hk

Evangeline F.Y. Young
CSE Department, CUHK
fyyoung@cse.cuhk.edu.hk

ABSTRACT

Inverse lithography technology (ILT) is one of the promising resolution enhancement techniques (RETs) in modern design-for-manufacturing closure, however, it suffers from huge computational overhead and unaffordable mask writing time. In this paper, we propose A2-ILT, a GPU-accelerated ILT framework with spatial attention mechanism. Based on the previous GPU-accelerated ILT flow, we significantly improve the ILT quality by introducing spatial attention map and on-the-fly mask rectilinearization, and strengthen the robustness by Reinforcement-Learning deployment. Experimental results show that, comparing to the state-of-the-art solutions, A2-ILT achieves 5.06% and 11.60% reduction in printing error and process variation band with a lower mask complexity and superior runtime performance.

1 INTRODUCTION

With the continuous shrinkage of technology nodes, resolution enhancement techniques (RETs) have been adopted ubiquitously to compensate for the printing errors caused by the lithography system limit. As one of the most prevailing RETs, Optical Proximity Correction (OPC) achieves the target layout image by directly modifying the on-mask features. Mainstream OPC can be roughly divided into three categories: (1) rule-based OPC [1][2], (2) model-based OPC [3][4] and (3) inverse lithography technology (ILT).

Rule-base OPC only relies on empirical rules, which is fast and efficient for specific object but difficult to generalize in complex patterns. For model-based OPC and ILT, they both leverage the dedicate computational lithography simulation model to guide the mask optimization process. The former is based on an iterative movement of the on-mask features guided by the resist images predicted by a lithography simulator, and it suffers from limited solution space and lacks flexibility. The latter adopts the idea of numerical optimization, treating the OPC task as an inverse imaging problem. ILT can be further divided into two kinds: parametric and implicit. The parametric one regards the OPC task as a process of pixel-wise translation from layout image to mask image. Classic attempts like [5][6][7] laid the foundation for current related works. On the other hand, the implicit methods, also known as the level set-based methods, treat the mask as a zero level set cross-section and obtain the optimized mask through an iterative update of the level set function [8][9].

With the recent development of computing hardware and the flourishing of deep learning technology, researchers start trying to combine ILT with GPU acceleration [10] and deep learning models. Regarding the parametric ILT, Yang et al. [11] developed an OPC-oriented generative adversarial nets (GAN) flow to learn the mapping between the target and the mask. Jiang et al. [12] proposed an end-to-end learning-based OPC framework consisting of a pretrained-model-based mask prediction module and an online ILT correction module. For implicit ILT, Chen et al. [13] migrated the level set-based method to deep neural network and shorten the average running time significantly by GPU speedup.

A well-optimized mask of a target layout can hardly be directly obtained with a single inference on a pretrained deep neural network. Instead, the deep models can produce a good initial solution, and some online update such as refinement [12] or evolution [13] can be adopted to iteratively improve the quality of the mask. This iterative mechanism is similar to the conventional ILT flow, but unfortunately, the conventional ILT flow has long running time overhead and can easily result in masks of high complexity.

In this paper, we propose A2-ILT, a GPU-accelerated ILT framework with spatial attention mechanism. First, on the basis of GPU-ILT proposed by [12], we introduce a spatial attention map to regulate the gradient update in the optimization process to suppress the occurrence of undesirable features and reduce mask complexity, while improving the printability. Secondly, we incorporate reinforcement learning (RL) to automatically modify the setting of the attention map for different layouts to enhance robustness. Finally, we develop the complete A2-ILT framework, with an on-the-fly mask rectilinearization during the ILT process to reduce the mask shot count. The main contributions include:

- A spatial attention map is introduced to regulate the gradient update in the optimization process for higher ILT quality.
- An RL architecture is incorporated to adaptively control the setting of the attention map for robustness.
- An on-the-fly mask rectilinearization during the ILT process is implemented to reduce the mask shot count.
- A complete A2-ILT framework is proposed, who outperforms the SOTA solutions in mask printability with less shot count and a comparable runtime performance.

The rest of the paper is organized as follows. Section 2 lists some preliminaries. Section 3 detailly discusses our algorithms and framework. Section 4 presents experimental results, followed by a conclusion in Section 5.

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 14209320).

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530579

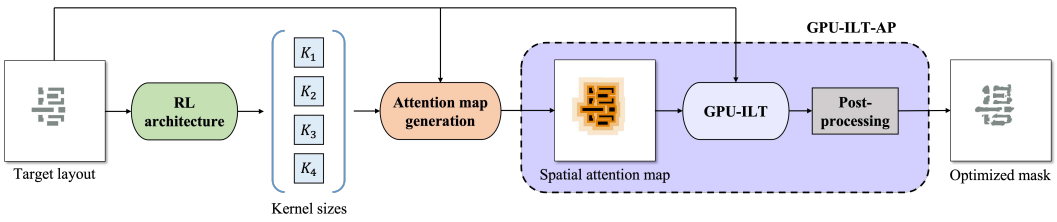

Figure 1: Overview of A2-ILT. The RL architecture is working to determine the kernel sizes that used to generate the spatial attention map according to the input layout. The generated attention map is fed to the GPU-ILT to assist the mask optimization.

2 PRELIMINARIES

In this section, we will introduce the background and formulate the problem. In consistent with previous works, we denote $\boldsymbol{Z}_t$ as the target layout, $\boldsymbol{M}$ as the mask image and $\boldsymbol{I}$ as the aerial image. The variables $\boldsymbol{Z}$, $\boldsymbol{Z}_{\text{in}}$, and $\boldsymbol{Z}_{\text{out}}$ represent the wafer image under nominal, min and max process condition $\boldsymbol{P}_{\text{nom}}$, $\boldsymbol{P}_{\text{min}}$, and $\boldsymbol{P}_{\text{max}}$ respectively.

2.1 Forward Lithography Simulation Model

The forward lithography simulation is to imitate the mapping from an input mask $\boldsymbol{M}$ to the output wafer image $\boldsymbol{Z}$. The wafer image $\boldsymbol{Z}$ is generated according to the light intensity distribution on the wafer plane provided by the aerial image $\boldsymbol{I}$, which is produced by the simulation model adopting Hopskin's diffraction theory [14]. Under the N_h^{th} order approximation [7], the model is expressed as:

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

where $N_k = 24$ in our implementation, $\otimes$ denotes the convolution operation, $\boldsymbol{h}_k$ is the k^{th} kernel and ω_k is the corresponding weight.

After that, the aerial image $\boldsymbol{I}$ is exposed on the photoresist, and a resist model is applied to generate the wafer image $\boldsymbol{Z}$ by checking whether the light intensity of an exposed area exceeds a threshold:

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

2.2 GPU-ILT

The objective of ILT is to find a well-optimized mask solution $\boldsymbol{M}^* = f^{-1}(\boldsymbol{Z}_t, \boldsymbol{P}_{\text{nom}})$ for the given layout $\boldsymbol{Z}_t$, where $f(\cdot, \boldsymbol{P}_{\text{nom}})$ is the forward lithography simulation under nominal process condition. However, there is no explicit closed-form solution for the inverse lithography process $f^{-1}(\cdot, \boldsymbol{P}_{\text{nom}})$ since different masks may yield the same wafer image. On the other hand, numerical approach like gradient decent is commonly used to minimize the mask printing error iteratively, which is usually given by a combination of two losses:

$$L = \alpha \cdot L_{\text{ilt}} + \beta \cdot L_{\text{pvb}}, \tag{3}$$

where α and β are configurable hyper-parameters. The L_{ilt} is calculated as:

$$L_{\text{ilt}} = \sum_{x=1}^{N_1} \sum_{y=1}^{N_2} (\boldsymbol{Z}(x,y) - \boldsymbol{Z}_t(x,y))^{\gamma}, \tag{4}$$

where $\boldsymbol{Z}_t$ is the target layout, $\boldsymbol{Z} = f(\boldsymbol{M}, \boldsymbol{P}_{\text{nom}})$ is the corresponding wafer image of $\boldsymbol{M}$, N_1 and N_2 are the width and height of the image, γ is a configurable parameter. The variable L_{pvb} is computed as:

$$L_{\text{pvb}} = \|\boldsymbol{Z}_{\text{in}} - \boldsymbol{Z}_{\text{out}}\|_2^2, \tag{5}$$

where $\boldsymbol{Z}_{\text{in}}$ and $\boldsymbol{Z}_{\text{out}}$ are the generated wafer images under min and max process condition $\boldsymbol{P}_{\text{min}}$ and $\boldsymbol{P}_{\text{max}}$ respectively.

In order to facilitate the conventional ILT flow, the GPU-ILT proposed in [12] utilizes GPU and CUDA toolkits to accelerate the lithography simulation, which finally speedups the ILT process with orders of magnitude. In this paper, we will use this GPU-ILT as a basic component.

2.3 Evaluation Method and Problem Formulation

The commonly used evaluation metrics in mask optimization are printability and complexity, which characterize the quality of the printed wafer patterns and the quality of the optimized mask respectively. In this paper, we adopt squared L_2 error and Process Variation Band (PVB) to measure printability and use the mask fracturing shot count to measure complexity.

Definition 1 (Squared L_2 Error). Squared L_2 error is calculated by $\|\boldsymbol{Z} - \boldsymbol{Z}_t\|_2^2$, where $\boldsymbol{Z}_t$ is the target layout image and $\boldsymbol{Z}$ is the generated wafer image under normal process condition $\boldsymbol{P}_{\text{nom}}$.

Definition 2 (Process Variation Band). Process variation band is given by the summation of bitwise-XOR between the two generated wafer images $\boldsymbol{Z}_{\text{in}}$ and $\boldsymbol{Z}_{\text{out}}$ under the min and max process condition $\boldsymbol{P}_{\text{min}}$ and $\boldsymbol{P}_{\text{max}}$ (±2% dose error in this work).

Definition 3 (Mask Fracturing Shot Count). Mask fracturing shot count is the total number of rectangular shots to correctly replicate the shape of a given mask $\boldsymbol{M}$, which indicates the complexity of the mask.

Problem 1 (Mask Optimization). Given a target layout $\boldsymbol{Z}_t$, it is needed to find an optimized mask $\boldsymbol{M}^*$, of which the printing image has minimum squared L_2 error with $\boldsymbol{Z}_t$ and process variation band, while the mask $\boldsymbol{M}^*$ has a small mask fracturing shot count.

3 ALGORITHMS AND FRAMEWORK

In this section, we first introduce the spatial attention map to regulate the optimization process in GPU-ILT. We then detail how we incorporate the RL architecture to control the setting of the attention map. Finally, we synthesize the complete A2-ILT framework. Fig. 1 illustrates an overview of A2-ILT, where the RL architecture first receives the input target layout to adaptively determine the kernel sizes to generate the spatial attention map. Subsequently,

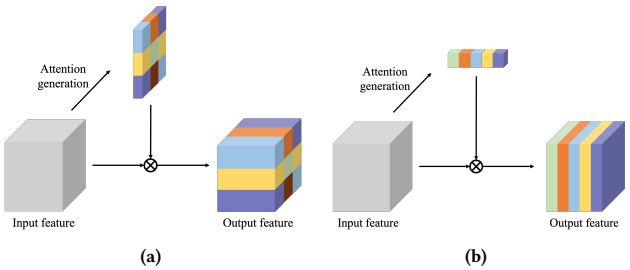

Figure 2: Two types of attention: (a) spatial attention and (b) channel attention.

the generated attention map together with the target layout are fed into the GPU-ILT to perform mask optimization, after which a post-processing is applied to obtain the final optimized mask.

3.1 Spatial Attention Map

The attention mechanism in computer vision is a simulation of a mechanism in the animal brain, which is used to suppress irrelevant information and emphasize important information by setting or learning a series of attention distributions. It can be roughly divided into spatial attention and channel attention based on the dimensionality of the operation, as shown in fig. 2. The former applies the attention distribution to the spatial dimension, so that the information at different locations in the space has different importance, while the latter varies across the channels. In this paper, we target at the binary layout images with single channel, so only the spatial attention mechanism is adopted.

Fig. 3(a) shows an optimized mask outputs for two target layouts using GPU-ILT. We can see that the conventional ILT flow may generate several different types of undesirable features in the masks, including holes, outliers and fractures, etc, and they will increase the mask complexity. These undesirable features are often unavoidable in conventional ILT flow without further regulations, because the procedures are sensitive to initial solution and step size. Repairing these features (e.g., filling holes or eliminating outliers) directly after the mask is optimized will definitely bring a big drop in printability. This motivates us to consider an on-the-fly regulation via spatial attention mechanism, which can help to allocate and control the importance of information in the spatial dimension, so as to adjust the update rate and suppress the occurrence of undesirable features as much as possible. Specifically, the spatial attention mechanism is realized by a pixel-wise multiplication of the gradient map and an attention map during the gradient update step in GPU-ILT.

The whole generation process of the spatial attention map is demonstrated in Fig. 4. We first design the attention map located inside the layout (m_1 in Fig. 4), which is obtained by corroding the layout with a kernel of size K_1. Subsequently, we dilate the layout three times using three kernels of different sizes (K_2, K_3, K_4) to obtain three external attention maps, namely m_2, m_3 and m_4 in Fig. 4. The four attention maps are then fused through a function $g(m_1, m_2, m_3, m_4)$, which can be expressed as:

$$g(m_1, m_2, m_3, m_4) = \sigma_2 m_2 + \sigma_3 m_3 + \sigma_4 m_4 - \sigma_1 m_1, \tag{6}$$

where $\sigma_1 = 1$, $\sigma_2 = 0.5$, $\sigma_3 = 0.3$, $\sigma_4 = 0.2$ in our implementation. It can be seen in Fig. 4 that the value of the innermost area of

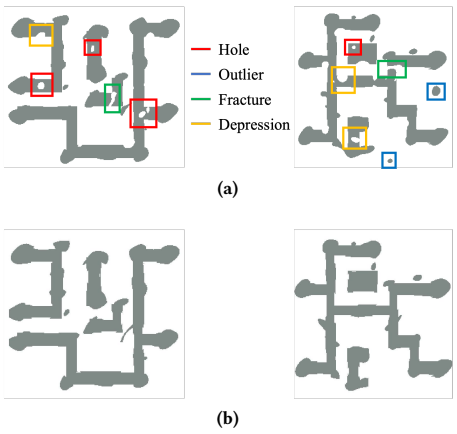

Figure 3: Optimized mask outputs from (a) GPU-ILT and (b) GPU-ILT with attention map.

the fused attention map is $0(= 0.5 + 0.3 + 0.2 - 1)$, which means that the pixels located in that area will not be updated during the optimization process. It helps to reduce the occurrence of holes and fractures as much as possible, and maintain a similar rectangular pattern as the target layout. The values of the three outer parts are $1(= 0.5 + 0.3 + 0.2)$, $0.5(= 0.3 + 0.2)$ and 0.2 respectively, meaning that the pixels closer to the layout boundaries will have a greater update rate, while the pixels farther from the layout patterns are updated slower. This kind of design is based on a prior knowledge of the ILT results from previous works that shape change of the mask should be focused around the mask contour, which is also effective to eliminate outliers. Fig. 3(b) shows the outputs from GPU-ILT with attention map. Compared with Fig. 3(a), the holes, fractures and far-away outliers can be successfully eliminated. However, it is still required to perform hole filling as a post-processing after GPU-ILT for full compliance of manufacturability. We call this GPU-ILT equipped with spatial attention map and hole filling post-processing GPU-ILT-AP (Fig. 1).

3.2 RL Architecture

The four kernel sizes in GPU-ILT-AP were carefully selected by humans, but this scheme is not robust for extension to different layouts. Reinforcement learning (RL) is good at learning human's decision making process, whose general form is shown in Fig. 5(a). It contains several important elements: environment, agent, action, state, and reward. The agent takes actions according to the state and receives reward from the environment, while the goal is to maximize the expected return. We formulate the problem of determining the kernel sizes as a sequential Markov Decision Processes (MDP), which can be fit into the general RL setting. In this way, as shown in Fig. 5(b), the general environment, agent, action, state, and reward are respectively replaced by our GPU-ILT-AP, PPO agent, kernel size of the attention map, generated attention map and loss-based reward, where the loss is calculated with equation 3. The PPO agent is realized based on a well known policy-gradient algorithm named Proximal Policy Optimization (PPO) [15], which ensures a relatively small deviation from the previous strategy during each iteration while effectively minimizing the cost function.

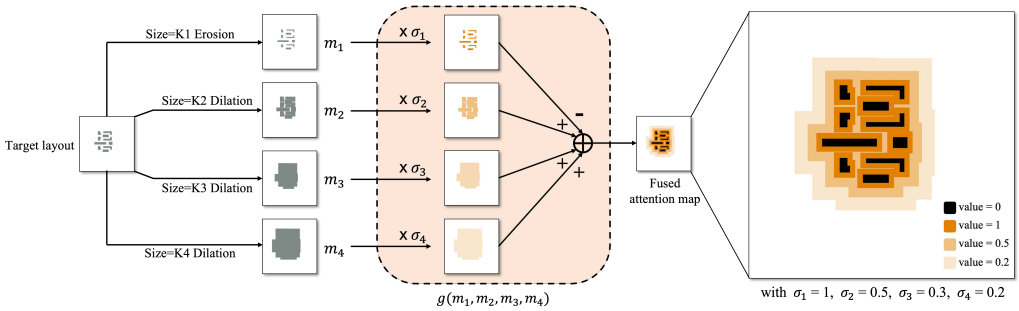

Figure 4: Generation process of spatial attention map.

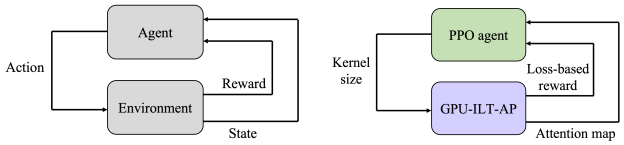

Figure 5: The (a) general form and the (b) corresponding migrated form of reinforcement learning.

The RL architecture is illustrated in Fig. 6, where the PPO agent consists of two deep neural networks named actor network and critic network. Each input target layout corresponds to a trajectory with 4 timestep, that is $\tau = (s_1, a_1, r_1, \cdots, s_4, a_4, r_4)$, where s_t, a_t and r_t $(t = 1, 2, 3, 4)$ represent the state, action and reward at the t-th timestep respectively. In each trajectory, the PPO agent takes the target layout as the input state s_1 and acts action a_1 accordingly, then the corresponding reward r_1 is received and kernel size K_1 is determined. Subsequently, s_2 is obtained by corroding the target layout using the kernel of size K_1. This sequence of steps are repeated until getting all the four kernel sizes. Since the actor network of the PPO agent has 10 outputs, as shown in Fig. 6, every action is selected from these 10 action candidates, i.e., $a_1, a_2, a_3, a_4 \in [1, 10]$. The kernel sizes K_1, K_2, K_3, K_4 are obtained from a_1, a_2, a_3, a_4 as follows:

$$\begin{aligned} K_1 &= a_1 \in [1, 10], \\ K_2 &= a_2 + 25 \in [26, 35], \\ K_3 &= a_3 + 45 \in [46, 55], \\ K_4 &= a_4 + 65 \in [66, 75]. \end{aligned} \tag{7}$$

The rewards r_1 to r_3 are set to be 0, while r_4 is calculated as follows:

$$r_4 = \frac{r_{\text{real}} - r_{\text{ref}}}{|r_{\text{ref}}|} = \frac{-(L_{\text{ilt}} + L_{\text{pvb}}) - (-(L_{\text{ilt-ref}} + L_{\text{pvb-ref}}))}{L_{\text{ilt-ref}} + L_{\text{pvb-ref}}}, \tag{8}$$

where L_{ilt} and L_{pvb} are the real losses calculated by formula 4 and 5 according to the output of GPU-ILT-AP, which takes the target layout and the four generated kernel sizes $[K_1, K_2, K_3, K_4]$ as input. $L_{\text{ilt-ref}}$ and $L_{\text{pvb-ref}}$ are the reference losses, which are calculated in the same way but using the manually selected kernel sizes [5, 30, 50, 70] as input. We can see from equation 8 that the loss function quantifies the normalized advantages of the current policy over the reference policy.

At the testing stage, the PPO agent directly takes the output with the highest probability of the trained actor network as the action at each timestep. Whenever a new target layout is fed into the network, the PPO agent completes a full four-step inference to obtain the four actions, after which the four kernel sizes are calculated and a fused attention map is generated.

Algorithm 1 A2-ILT

Input: Target layout $\boldsymbol{Z}_t$
function `Kernel_size_determination`($\boldsymbol{Z}_t$)
  $K_s = []$;
  $s_1 \leftarrow \boldsymbol{Z}_t$;
  for $i \leftarrow 1$ to 4 **do**
    $a_i \leftarrow PPO\_agent(s_i)$;
    $K_i \leftarrow$ calculate kernel size from a_i;
    $s_{i+1} \leftarrow$ obtained by corrosion or dilation of $\boldsymbol{Z}_t$ using K_i;
    $K_s.append(K_i)$;
  return K_s
Input: Target layout $\boldsymbol{Z}_t$, lithography kernels $\boldsymbol{H}$, $\boldsymbol{H}^*$, simulation weight ω, learning rate lr, scale factor s
function `A2-ILT`($\boldsymbol{Z}_t$, $\boldsymbol{H}$, $\boldsymbol{H}^*$, ω, lr, s)
  Load $\boldsymbol{Z}_t$, $\boldsymbol{H}$, $\boldsymbol{H}^*$, ω into GPU memory;
  $\boldsymbol{M} \leftarrow \boldsymbol{Z}_t$;
  $\boldsymbol{M}_{\text{p}} \leftarrow$ initialized corresponding to $\boldsymbol{M}$;
  $K_s \leftarrow$ `Kernel_size_determination`($\boldsymbol{Z}_t$);
  $\boldsymbol{A} \leftarrow$ generate attention map using kernel sizes K_s;
  repeat
    $G \leftarrow CUDA\_GRAD(\boldsymbol{Z}_t, \boldsymbol{M}, \boldsymbol{M}_{\text{p}}, \boldsymbol{H}, \boldsymbol{H}^*, \omega)$;
    $\underline{\boldsymbol{G}} \leftarrow Downsample(\boldsymbol{G}, s)$, $\underline{\boldsymbol{M}_{\text{p}}} \leftarrow Downsample(\boldsymbol{M}_{\text{p}}, s)$;
    $\boldsymbol{M}_{\text{p}} \leftarrow Upsample(\underline{\boldsymbol{M}_{\text{p}}} - lr \times \underline{\boldsymbol{G}} \cdot \boldsymbol{A}, s)$;
    $\boldsymbol{M} \leftarrow$ Recalculate pixel values from $\boldsymbol{M}_{\text{p}}$;
  until Maximum number of iterations;
  $\boldsymbol{M}^* \leftarrow \boldsymbol{M}$ with the best performance;
  return $\boldsymbol{M}^*$

3.3 Complete A2-ILT Framework

By integrating the above RL architecture to adaptively determine the kernel sizes for GPU-ILT-AP, we construct the complete A2-ILT framework, as revealed in Fig. 1. Algorithm 1 depicts the A2-ILT flow. The ILT process involves an intermediate variable $\boldsymbol{M}_{\text{p}}$ to relax the binary pixel value in $\boldsymbol{M}$ into real numbers by the following sigmoid approximation:

$$\boldsymbol{M} = Sigmoid(\boldsymbol{M}_{\text{p}}) = \frac{1}{1 + e^{-\theta_M \times \boldsymbol{M}_{\text{p}}}}, \tag{9}$$

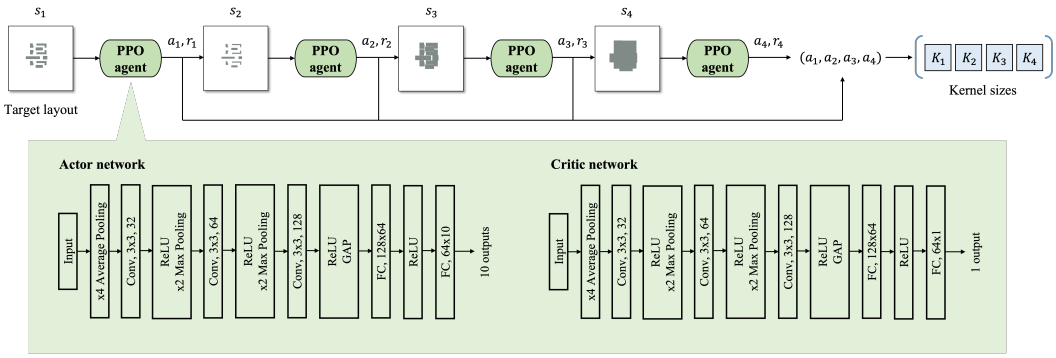

Figure 6: Illustration of RL architecture.

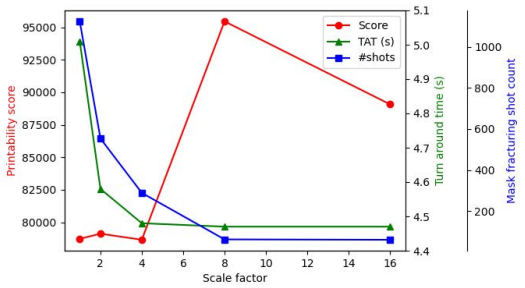

Figure 7: Different scale factors with corresponding printability score, mask fracturing shot count, turn around time.

where θ_M is the steepness and set to 4 in our implementation. The *CUDA_GRAD* function represents the whole process of using CUDA toolkits to calculate the gradients in GPU-ILT. In A2-ILT, a down sampling and up sampling pair with scale factor s (line 18, 19 in algorithm 1) is adopted to realize the on-the-fly rectilinearization during the ILT optimization to reduce the mask shot count while maintaining the printability. The experimental results in section 4.1 verify this property, and a study of the trade-off between printability and complexity by changing this scale factor is investigated.

4 EXPERIMENTAL RESULTS

The A2-ILT framework is developed on PyTorch with CUDA. All the tests are performed on Linux system with 2.2GHz Intel Xeon CPU and a single Nvidia GeForce RTX 3090 GPU. The layout dataset used for training our PPO agent is obtained from the authors of GAN-OPC [11], which is a 1,000 random-selection from the original 4300 instances. The evaluation dataset is composed of ten industrial M1 designs on 32nm design node provided by ICCAD 2013 CAD Contest [16]. For simplicity, in this section we use "L2", "PVB", "#shots", "TAT" and "Score" to denote squared L_2 error, process variation band, mask fracturing shot count, turn around time and printability score (= squared L_2 error + process variation band) respectively. We set the hyper-parameters as follows: $\gamma = 4$, $\alpha = 0.9$, $\beta = 0.1$, $lr = 1$, $s = 4$.

4.1 Comparison of Scale Factors

We tested GPU-ILT-AP with different scale factors ($s = 1/2/4/8/16$) to see their corresponding printability score, mask fracturing shot count and turn around time. The results are shown in Fig. 7. We can see that as the scale factor s increases, printability is stable and good when s is less than 4, but it deteriorates quickly when s is greater than 4. On the other hand, the shot count and runtime show a downward trend, which validates the effectiveness of the on-the-fly rectilinearization introduced in section 3.3. When $s = 4$, a balanced performance is achieved, so we adopt this value in our implementation.

Table 1: Mask printability, complexity, and runtime comparison concerning the existence of attention map

	L2 (nm^2)	PVB (nm^2)	#shots	TAT (s)
GPU-ILT-AP	**35630.2**	**43026.5**	**288.4**	4.48
GPU-ILT-AP (w/o. attention map)	37081.7	44717.9	341.2	**4.47**

Table 2: Mask printability comparison among inference-mode A2-ILT and two control groups

	L2 (nm^2)	PVB (nm^2)	Score
A2-ILT (inference)	35615.4	43027.7	**78643.1**
A2-ILT (reference)	35630.2	43026.5	78656.7
A2-ILT (random strategy) (100 times average)	37241.2	42639.2	79880.4

4.2 The Effectiveness of Attention Map

To verify the effectiveness of the spatial attention map, we compare the GPU-ILT-AP with kernel sizes = [5, 30, 50 ,70] and the attention-map-removed GPU-ILT-AP on the validation set. Table 1 shows the comparison results. It is obvious that the GPU-ILT-AP outperforms the one without attention map in terms of "L2", "PVB" and "#shots" with a comparable runtime performance, which indicates the importance and effectiveness of using the attention map.

4.3 The Validity of the RL Architecture

We trained the PPO agent on a single Nvidia GeForce RTX 3090 GPU for 25 hours. After training, we test A2-ILT on the validation dataset. Besides the direct inference mode, we set two control groups here: (1) reference baseline using manually selected kernel sizes [5, 30, 50, 70] and (2) random strategy. The results of the random strategy are obtained by taking the average of 100 tests. Quantitative results are listed in Table 2. We can see that A2-ILT with automatic inference

Table 3: Mask printability and complexity comparison with SOTA methods

Benchmarks		ILT [7]			PGAN-OPC [11]			Neural-ILT [12]			DevelSet [13]			A2-ILT		
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	#shots	L2 (nm^2)	PVB (nm^2)	#shots	L2 (nm^2)	PVB (nm^2)	#shots	L2 (nm^2)	PVB (nm^2)	#shots	L2 (nm^2)	PVB (nm^2)	#shots
case1	215344	49893	65534	2478	52570	56267	931	49817	55975	428	49142	59607	969	44477	49064	304
case2	169280	50369	48230	704	42253	50822	692	38174	52010	256	34489	52012	743	34031	40165	258
case3	213504	81007	108608	2319	83663	94498	1048	89411	91357	557	93498	76558	889	88901	94850	493
case4	82560	20044	28285	1165	19965	28957	386	16744	29982	136	18682	29047	376	17464	24636	218
case5	281958	44656	58835	1836	44733	59328	950	45598	58900	380	44256	58085	902	35685	50664	351
case6	286234	57375	48739	993	46062	52845	836	43836	54969	383	41730	53410	774	41432	43516	301
case7	229149	37221	43490	577	26438	47981	515	20324	50542	244	25797	46606	527	21901	37688	245
case8	128544	19782	22846	504	17690	23564	286	13337	26353	285	15460	24836	493	15257	18548	177
case9	317581	55399	66331	2045	56125	65417	1087	49401	68817	444	50834	64950	932	45885	56668	382
case10	102400	24381	18097	380	9990	19893	338	8511	20734	208	10140	21619	393	11121	14478	152
Average		44012.7	50899.5	1300.1	39948.9	49957.2	706.9	37515.3	50963.9	332.1	38402.8	48673.0	699.8	**35615.4**	**43027.7**	**288.1**
Ratio		1.236	1.183	4.513	1.122	1.161	2.454	1.053	1.184	1.153	1.078	1.131	2.429	**1.000**	**1.000**	**1.000**

Table 4: Runtime comparison with SOTA methods

Benchmarks ID	ILT [7] TAT (s)	PGAN-OPC [11] TAT (s)	Neural-ILT [12] TAT (s)	DevelSet [13] TAT (s)	A2-ILT TAT (s)
case1	1280	358	11	**1.5**	4.53
case2	381	368	17	**1.4**	4.5
case3	1123	368	10	**1.29**	4.54
case4	1271	377	9	**1.65**	4.51
case5	1120	369	11	**0.91**	4.53
case6	391	364	10	**0.84**	4.52
case7	406	377	16	**0.76**	4.51
case8	388	383	15	**1.14**	4.48
case9	1138	383	11	**1.21**	4.52
case10	387	366	14	**0.42**	4.5
Average	788.5	371.3	12.4	**1.11**	4.51
Ratio	709.083	333.903	11.171	**1.000**	4.059

Note: The devices used in different methods may have variance.

can achieve better performance in printability than the reference baseline, in which kernel sizes are selected carefully by humans, and the random strategy on this validation dataset, which has not been seen in the training process. This verifies the validity and robustness of the RL architecture.

4.4 Comparison with the State-of-the-art Works

We compare A2-ILT with several recent works on the validation dataset. Detailed experimental results are given in Table 3 and 4. We can see that A2-ILT outperforms all other methods in printability and complexity. Specifically, compared with the previous best results, "L2", "PVB", and "#shots" are reduced by 5.06%, 11.60% and 13.25% respectively. In addition, the average "TAT" of A2-ILT is 99.43% less than the conventional ILT [7] and 63.63% less than Neural-ILT [12].

5 CONCLUSION

In this paper, we propose A2-ILT, a GPU-accelerated ILT framework with spatial attention mechanism. We significantly improve the ILT quality of the previous GPU-accelerated ILT flow by introducing the spatial attention map as a regulation mechanism of the optimization process and implementing the on-the-fly rectilinearization to reduce the mask shot count. An RL framework is integrated into our flow to adaptively control the setting of the attention map for robustness, which is well applied to unseen target layouts. The results show that A2-ILT outperforms other SOTA solutions in printability with a lower mask complexity and superior runtime performance.

REFERENCES

[1] X. Dong and L. Zhang, "Process-variation-aware rule-based optical proximity correction for analog layout migration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 8, pp. 1395–1405, 2016.

[2] J.-S. Park, C.-H. Park, S.-U. Rhie, Y.-H. Kim, M.-H. Yoo, J.-T. Kong, H.-W. Kim, and S.-I. Yoo, "An efficient rule-based opc approach using a drc tool for 0.18/spl mu/m asic," in *Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat. No. PR00525)*. IEEE, 2000, pp. 81–85.

[3] J. Kuang, W.-K. Chow, and E. F. Young, "A robust approach for process variation aware mask optimization," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1591–1594.

[4] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 35, no. 8, pp. 1345–1357, 2016.

[5] A. Poonawala and P. Milanfar, "Mask design for optical microlithography—an inverse imaging problem," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 774–788, 2007.

[6] F. Liu and X. Shi, "An efficient mask optimization method based on homotopy continuation technique," in *2011 Design, Automation & Test in Europe*. IEEE, 2011, pp. 1–6.

[7] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "Mosaic: Mask optimizing solution with process window aware inverse correction," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–6.

[8] D. S. Abrams and L. Pang, "Fast inverse lithography technology," in *Optical Microlithography XIX*, vol. 6154. International Society for Optics and Photonics, 2006, p. 61541J.

[9] W. Lv, S. Liu, Q. Xia, X. Wu, Y. Shen, and E. Y. Lam, "Level-set-based inverse lithography for mask synthesis using the conjugate gradient and an optimal time step," *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, vol. 31, no. 4, p. 041605, 2013.

[10] I. Torunoglu, A. Karakas, E. Elsen, C. Andrus, B. Bremen, B. Dimitrov, and J. Ungar, "A gpu-based full-chip inverse lithography solution for random patterns," in *Design for Manufacturability through Design-Process Integration IV*, vol. 7641. International Society for Optics and Photonics, 2010, p. 764115.

[11] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Young, "Gan-opc: Mask optimization with lithography-guided generative adversarial nets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 10, pp. 2822–2834, 2019.

[12] B. Jiang, L. Liu, Y. Ma, B. Yu, and E. F. Young, "Neural-ilt 2.0: Migrating ilt to domain-specific and multi-task-enabled neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2021.

[13] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "Develset: Deep neural level set for instant mask optimization," in *Proc. ICCAD*, 2021, pp. 1–9.

[14] H. H. Hopkins, "The concept of partial coherence in optics," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 208, no. 1093, pp. 263–277, 1951.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 271–274.