Bridging Hotspot Detection and Mask Optimization via Domain-Crossing Masked Layout Modeling

BINWU ZHU, Southeast University, Nanjing, China
SU ZHENG, The Chinese University of Hong Kong, Hong Kong, Hong Kong
YUZHE MA, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
BEI YU, The Chinese University of Hong Kong, Hong Kong, Hong Kong
MARTIN WONG, Hong Kong Baptist University, Hong Kong, Hong Kong

With the rapid development of semiconductors, the size of transistors is continuously scaling down. The shrinking circuit size poses great challenges to optical proximity correction (OPC) and hotspot detection (HSD). Recent advancements in OPC and HSD commonly employ deep neural networks, achieving impressive performance within a limited runtime. Based on these achievements, we observe that deep-learning-based models of both HSD and OPC require knowledge of layout structure information. Furthermore, these two tasks are closely related to the lithography process during chip manufacturing. Observing such strong relationships, we propose that integrating OPC and HSD into a unified deep learning model will contribute to the performance of both tasks. To bridge the relationship between OPC and HSD, we first pre-train a layout understanding model built on the mask modeling technique, which effectively captures the layout geometric information, and then the pre-trained model can be easily fine-tuned on HSD and OPC with limited data. To fully pre-train the layout understanding model (LUM), we create a large layout dataset using layout generation techniques, solving the data-hungry issues. Experimental results show that the fine-tuned LUM model achieves remarkable performance on both OPC and HSD tasks.

CCS Concepts: • **Hardware → Design for manufacturability**; **Methodologies for EDA;**

Additional Key Words and Phrases: Mask Optimization, Hotspot Detection, Masked Modeling

ACM Reference Format:
Binwu Zhu, Su Zheng, Yuzhe Ma, Bei Yu, and Martin Wong. 2025. Bridging Hotspot Detection and Mask Optimization via Domain-Crossing Masked Layout Modeling. *ACM Trans. Des. Autom. Electron. Syst.* 30, 4, Article 54 (June 2025), 20 pages. https://doi.org/10.1145/3728468

1 Introduction

In chip manufacturing, due to the proximity effect [1] during the lithography process, mask optimization techniques, including **optical proximity correction (OPC)**, are first adopted to obtain the desired mask patterns, which facilitates the reduction of printing errors. **Inverse lithography technology (ILT)** [2] is a mathematically rigorous approach that optimizes the shapes on the mask

Authors' Contact Information: Binwu Zhu, Southeast University, Nanjing, Jiangsu, China; e-mail: bwzhu@seu.edu.cn; Su Zheng, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: szheng22@cse.cuhk.edu.hk; Yuzhe Ma, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China; e-mail: yuzhema@hkust-gz.edu.cn; Bei Yu, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: byu@cse.cuhk.edu.hk; Martin Wong, Hong Kong Baptist University, Hong Kong, Hong Kong; e-mail: mdfwong@hkbu.edu.hk.

ACM 1084-4309/2025/06-ART54
https://doi.org/10.1145/3728468

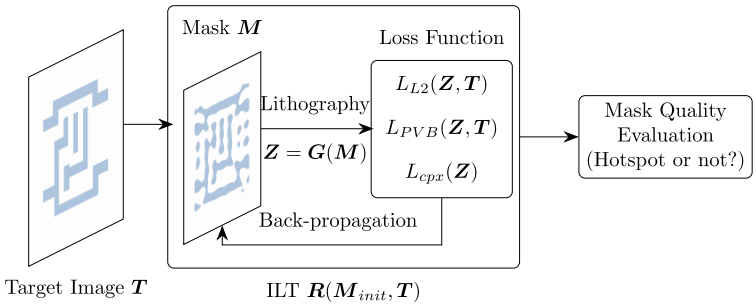


Fig. 1. Traditional mask optimization and hotspot detection process, where detecting hotspot regions has to rely on the result of previous mask optimization.

to achieve the desired wafer patterns. It has been explored and developed as the new generation of mask optimization techniques, which is expected to solve the challenges of advanced technology nodes such as **extreme ultraviolet (EUV)** [3]. However, traditional ILT algorithms [4–6] typically adopt iterative methods such as gradient descent to optimize the misfit between the printed image on the wafer and the target pattern, and the whole process requires many iterations for convergence, suffering from runtime overhead. To alleviate this problem, researchers propose multiple algorithms to boost the efficiency of ILT [7–15]. Some methods try to optimize the traditional ILT algorithm. For example, Yu et al. [8] rely on the CUDA toolkit to implement a GPU-accelerated Fourier transform algorithm, accelerating a critical and time-consuming step in the lithography simulation model. Inspired by the multigrid method, the proposed architecture in [14] involves a lithography simulation scheme that operates at multiple resolutions. The proposed coarse-to-fine simulation scheme can effectively achieve ILT acceleration. Chen et al. [15] introduce a differentiable lithography imaging framework based on differentiable programming. It fully leverages the computational power of GPUs to simplify the optimization of resolution enhancement.

In recent years, the advancement of **convolutional neural networks (CNNs)** has spawned powerful techniques for ILT acceleration. Related works try to learn an ILT solver using stacked convolutional layers. By fully utilizing the massive computation resources of GPU, they can output the optimized mask patterns within a short runtime. For example, GAN-OPC [9] utilizes a CNN-based generation model, GAN [16], 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.

Although considerable progress has been made in mask optimization, there may still exist defects that can potentially lead to open or short failures. As illustrated in Figure 1, traditional defect detection methods typically sample a series of points on the optimized masks of the layout pattern and then measure **edge placement errors (EPEs)**. The locations with unacceptable EPEs are regarded as hotspot regions. The main drawback is that these methods cannot achieve end-to-end defect detection since they always require mask optimization results, causing extra time overhead. Moreover, the calculation of EPE is not straightforward, and due to the varying complexities of layouts, the number of sampling points differs significantly, making it challenging to perform parallel computations. Therefore, it is not efficient to identify the potential defects after the lithography simulation. Besides, since such a detection process involves sampling, the accuracy is also influenced by the sampling frequency. Higher sampling frequencies result in higher detection accuracy but lower efficiency. We hope to advance the defect detection process. By performing hotspot detection before lithography simulation, potential hotspot regions can be identified early in the design flow, which allows design teams to take proactive measures to address these hotspots,

such as applying design rule modifications or layout optimizations, before the lithography simulation. Early detection of hotspots enables designers to mitigate potential issues and improve the overall manufacturability of the chip.

Various practical and efficient hotspot detection methods have been developed to overcome these limitations in recent years. The main idea is to directly detect hotspot regions from the given target patterns instead of analyzing the mask patterns after the time-consuming mask optimization process. In this way, the overall runtime can be remarkably reduced. Recent hotspot detection can be generally divided into two categories: (1) The first relies on pattern-matching techniques. These methods [17–19] first take a collection of hotspot layout patterns and use them to scan over new designs to identify any matched patterns as hotspots. For example, Wen et al. [18] propose a density-based layout encoder that can discriminate between the hotspots and non-hotspots via **principal component analysis (PCA)**. (2) The second category formulates the hotspot detection task as a binary classification problem, which can be effectively tackled by **deep neural network (DNN)** models. Related works [20–26] are encouraged by the great success of learning models in the image classification field. For example, Yang et al. [22] propose to extract layout features with discrete cosine transform and utilize a CNN architecture for hotspot detection. The performance is further improved with the proposed bias learning algorithm because of the imbalanced dataset. Inspired by the object detection problem in computer vision, Chen et al. [25] propose to detect multiple hotspots within large layouts simultaneously.

For pattern-matching methods, although they can overcome the time-consuming drawback, their generalization abilities are always unsatisfactory. To be specific, when given hotspot patterns that cannot match any pattern in the pre-collected dataset, these methods may fail to detect them. As a result, deep learning models, which can automatically learn to extract key features from layout patterns, are preferred in the state of the art. However, there are still some issues with learning-based methods: (1) Current deep-learning-based hotspot detection models are usually trained and tested on small labeled datasets, such as the ICCAD 2012 benchmark [27]. These small datasets lead to a data-hungry problem in academic research of EDA and may easily cause over-fitting of learning models. Therefore, learning models trained by these datasets usually have poor performance when transferred to real industrial scenarios. (2) As mentioned above, hotspot detection is closely related to mask optimization since the quality of mask optimization will determine whether any hotspot exists. However, these methods simply regard hotspot detection as an image classification problem and do not effectively utilize any prior knowledge of the mask optimization and lithography process.

To improve the generalization ability, we aim to design a large pre-trained model, which can leverage more training data. Such a model can be easily transferred to multiple downstream EDA tasks. Considering that both hotspot detection and mask optimization call for the understanding and knowledge of the layout geometry and the lithography process, we argue that hotspot detection and mask optimization should be integrated into a unified deep learning model, as shown in Figure 2. Based on these motivations, a strong pre-trained **layout understanding model (LUM)** is proposed, where we design a customized training scheme called domain-crossing masked layout modeling. The proposed framework is inspired by the large pre-trained model in deep learning, such as **mask auto-encoder (MAE)** [28] and **generative pre-trained transformer (GPT)** [29], which achieves incredible performance on many downstream tasks. In LUM, we first generate a large amount of layout data and randomly mask a portion of the layout tile. Then, a training scheme called domain-crossing mask layout modeling is proposed to guide the model training, which is illustrated in Figure 3. It can be seen that LUM is responsible for restoring both target images and ILT results from the given masked images. In this way, we can not only leverage massive masked layout data to pre-train the model sufficiently but also embed the awareness of

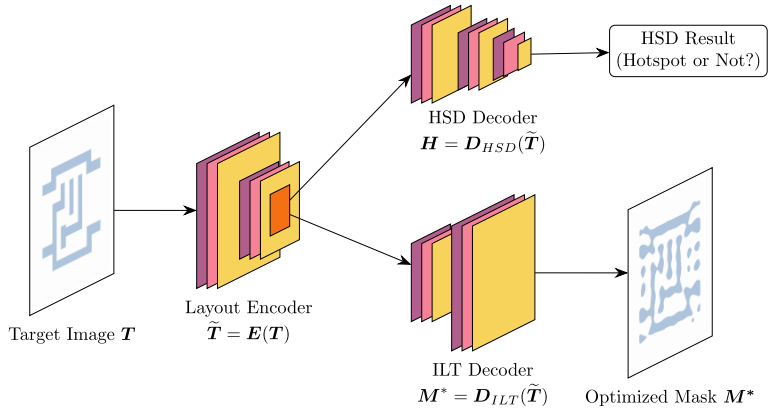


Fig. 2. Our proposed unified framework. A pre-trained layout understanding encoder can boost performance on both OPC and HSD.

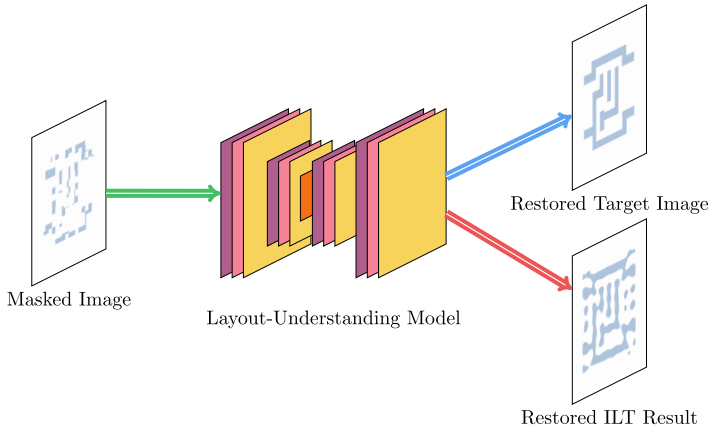


Fig. 3. Illustration of domain-crossing masked layout modeling. It restores the desired patterns from a masked layout tile.

the lithography process and mask optimization into LUM. After pre-training, we can easily fine-tune the pre-trained LUM with fewer data while achieving satisfactory performance on various layout understanding tasks, including both OPC and HSD. The main contributions of this article are listed as follows:

— We unify mask optimization and hotspot detection through the proposed pre-trained LUM.
— We create the SynLayout dataset, a large layout dataset, using layout generation techniques, which solves the data-hungry issue.
— Our proposed learning scheme, masked layout modeling, helps LUM better capture the geometric information of layout patterns, which contributes to the ability of mask optimization and hotspot detection.
— Experimental results show that our pre-trained LUM achieves remarkable performance when fine-tuned on both OPC and HSD tasks.

The rest of this article is organized as follows. Section 2 introduces preliminaries about OPC, HSD, and mask modeling. Section 3 elaborates on the LUM with customized domain-crossing mask modeling and target reconstruction mechanisms. Section 4 details our generated large layout dataset and experimental results, followed by the conclusion in Section 5.

2 Preliminaries

In this section, we will introduce some preliminary knowledge related to this work.

2.1 Optical Proximity Correction

During the lithography process, the 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 forms the printed image $\boldsymbol{Z}$ with the development and etching processes. OPC is a widely used mask optimization technique, which is used to find an optimized mask $\boldsymbol{M}_{opt} = f^{-1}(\boldsymbol{Z}_t)$, where $\boldsymbol{Z}_t$ is the design target, and $f(\cdot)$ stands for the lithography process under the nominal process condition.

A mathematical model is proposed to simulate the lithography process. It is composed of two parts: an optical projection model and a photoresist model. For the optical projection process, the Hopkins diffraction model of the partial coherence imaging system is adopted to approximate the projection behavior. In mathematics, the aerial image $\boldsymbol{I}$ can be approximated by convolving the mask $\boldsymbol{M}$ with a set of optical kernels $\boldsymbol{H}$ [2], formulated as

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

where $\boldsymbol{h}_k$ is the kth optical kernel of the optical kernel set $\boldsymbol{H}$, and w_k is the corresponding weight of the coherent system. "$\otimes$" represents the convolution operation. To save the computation resources, an N_hth-order approximation to the partially coherent system is proposed in [2], represented as

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

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:

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

To evaluate the quality of the optimized mask, we define the following three metrics.

2.1.1 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, as shown in Figure 4(a). The **process variation band (PV Band)** computes the difference between the outermost and innermost contour to evaluate the printing robustness, which is formulated as follows:

$$PVB(\boldsymbol{Z}_{out}, \boldsymbol{Z}_{in}) = ||\boldsymbol{Z}_{out} - \boldsymbol{Z}_{in}||_2^2. \tag{4}$$

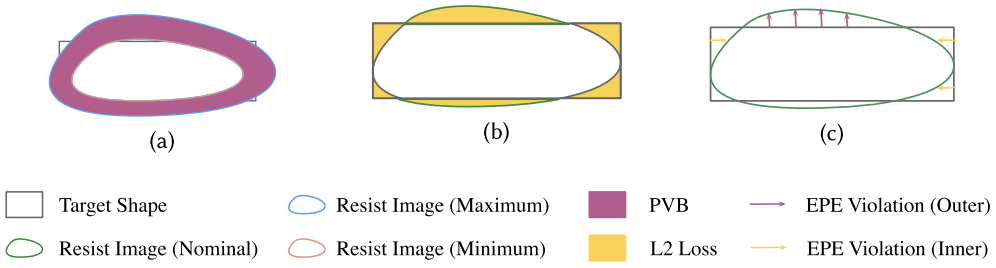


Fig. 4. Illustration of the metrics. (a) *PVB* quantifies the discrepancy between the maximum and minimum process corners. (b) *L2* measures the difference between the nominal resist image and the desired target image. (c) *EPE* estimates the geometric distortion of the nominal resist image.

2.1.2 Square L_2 Error. Given the target image Z_t and the printed image Z_{nom}, which represents the image printed via nominal lithography process condition, the square L_2 loss is calculated as:

$$L_2(Z_{nom}, Z_t) = ||Z_{nom} - Z_t||_2^2. \tag{5}$$

2.1.3 Edge Placement Error (EPE). Edge placement error is used to evaluate the difference of the contour between the target design Z_t and the image Z_{nom}. To calculate the EPE, a series of points are sampled along the contour of the target design. If the distance $D(x, y)$ between the target design and the printed image is larger than an EPE constraint th_{EPE}, the point (x, y) is labeled as an EPE violation:

$$\text{EPE Violation}(x, y) = \begin{cases} 1, & D(x, y) \geq th_{EPE}; \\ 0, & D(x, y) < th_{EPE}. \end{cases} \tag{6}$$

2.2 Hotspot Detection

The lithographic process always involves many variations, and some patterns are sensitive to these variations, which may cause potential open or short-circuit failures. These failures will undoubtedly reduce manufacturing yields. Layout pattern regions that are sensitive to lithographic process variations are defined as hotspot regions. To improve the manufacturing yield, an efficient and accurate hotspot detection technique must be developed to help locate the defect positions of layouts. A high-performance hotspot detector should correctly detect as many hotspots as possible and avoid mistaking non-hotspot patterns for hotspot patterns. To evaluate the performance, we define the following metrics.

2.2.1 Accuracy. The ratio between the number of correctly detected hotspots and the number of ground-truth hotspots.

2.2.2 False Alarm. The number of non-hotspots that are predicted as hotspots by the classifier.

2.3 Masked Modeling

Masked modeling is such a learning task: masking a portion of input contents and attempting to restore the contents hidden by the mask [30]. Masked language modeling, including BERT [31] and GPT [29], is the first successful application of masked modeling in natural language processing. Typically, it is a fill-in-the-blank self-supervised learning task, where a model learns representations by predicting what a masked word should be according to the context words surrounding the token. Recently, masked image modeling [28, 32] follows a similar way to learn representations by predicting the missing parts at the pixel or patch level. Once the model is trained and evaluated, it

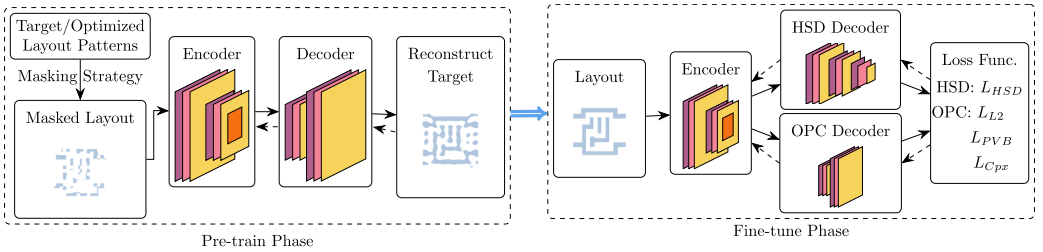


Fig. 5. Overview of the proposed framework, consisting of the pre-training, fine-tuning, and co-evolution phases. The solid and dashed lines indicate the feed-forward and back-propagation in training, respectively.

can be deployed in production systems for various applications, such as object detection, instance segmentation, semantic segmentation, or medical image analysis. Mask modeling plays a crucial role in computer vision tasks where precise localization and pixel-level understanding of objects or regions are required.

Nonetheless, the techniques described above have only been shown to be useful for natural language and image modeling. Masked modeling has not been fully explored in the EDA domain. It is widely known that there are many layout-related tasks that call for strong layout understanding abilities of the deep learning model. In this work, we aim to investigate the application of masked modeling for more robust layout feature extraction. We mask a portion of the layout, and our LUM is responsible for restoring the masked layout image. In addition, regarding the uniqueness of EDA layouts and tasks, we have made many customizations to the mask layout modeling mechanism, which is significantly different from the mask modeling mechanisms in computer vision and natural language processing fields, which will be detailed in Section 3.5.

With the evaluation metrics defined in Sections 2.1 and 2.2, we formulate the objective of LUM as follows:

Problem 1 (Layout Understanding Model). *Given a collection of layout patterns, the objective of the LUM is to learn to capture the geometry information so that it can achieve satisfactory performance on both OPC and HSD. For OPC, the printed image is supposed to be close to the target image under different process conditions, such that the EPE, L_2 loss, and PV Band are minimized. For HSD, LUM is supposed to locate and classify all hotspots and non-hotspots, such that the detection accuracy is maximized and the false alarm is minimized.*

3 Algorithms

In this section, the architecture of our LUM will be explained in detail. We will introduce the critical components that enable the LUM to be equipped with strong generalization abilities that differentiate LUM from previous attempts on layout feature learning models. We also show how to incorporate the prior knowledge of mask optimization into LUM, which remarkably contributes to the HSD and OPC performance.

3.1 Overview

As shown in Figure 5, the proposed framework consists of two phases:

3.1.1 Pre-training Phase. In this phase, we employ the proposed domain-crossing masked layout modeling method to pre-train the model. The masking strategy (Section 3.2) masks a specific portion of input patches, the LUM encoder (Section 3.3) maps the input patches to the latent space, and the LUM decoder (Section 3.4) aims to recover the desired layout patterns. During

the pre-training stage, the inputs and outputs can be target images or optimized masks. This training scheme enables the model to acquire knowledge of the two domains. The reconstruction target (Section 3.5) guides the pre-training process via back-propagation.

3.1.2 Fine-tuning Phase. Since our framework is targeted at OPC and HSD, we need an OPC decoder and an HSD decoder to accomplish these two tasks according to the encoding from the LUM encoder. We fine-tune the LUM decoder used in the pre-training phase to build the OPC decoder. In addition, we introduce a new HSD decoder for classification. Section 3.6 demonstrates how to fine-tune the model on these tasks.

3.2 Masking Strategy

Before training our LUM framework to restore the layout pattern, the first step is to mask a portion of the layout tile suitably. Most recent mask modeling approaches, such as MAE [28] and SimMIM [32], followed a uniformly random masking method at the patch level. Inspired by these works, the layout pattern in our work is first divided into non-overlapping patches of size 4×4, and part of the patches are randomly masked.

The masking ratio is critical for model training. A suitable ratio can ensure that the masking layout patterns effectively eliminate redundancy, resulting in a task that cannot be solved by extrapolation from visible neighboring patches. In previous vision works, a commonly adopted masking ratio is 75%. However, we find that such a high ratio is not suitable for our task since the meaning of the semantic information of layout patterns is more sparse than in regular images. To determine a suitable masking ratio, a series of experiments are investigated, and we finally set the masking ratio as 50%.

3.3 Encoder

The LUM encoder is responsible for modeling latent feature representations of the masked patches, which are then utilized to forecast the original signals in the masked area. The learned encoder should be capable of adapting to different tasks. Recently, Transformer [33] has become a powerful encoder in both vision and language areas. The transformer encoder consists of multiple layers, of which the most important one is multi-head self-attention, allowing the model to capture information at different positions globally [33]. To deal with image inputs, **Vision Transformer (ViT)** [34] splits an image into fixed-size patches, each of which is regarded as a token in sequential data. Given an input layout of size $H \times W \times C$ and the patch size P, the representation is first transformed into patches $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\frac{HW}{P^2}}\}$. The Transformer encoder first packs the patches as a matrix, represented as $\boldsymbol{X} \in \mathbb{R}^{\frac{HW}{P^2} \times CP^2}$. Next, a fully connected layer maps the patches to the input embeddings $\boldsymbol{X}_E \in \mathbb{R}^{\frac{HW}{P^2} \times N_H}$. To differentiate the positions of patches, positional encodings $\boldsymbol{E}_P \in \mathbb{R}^{\frac{HW}{P^2} \times N_H}$ are then added to the input embeddings:

$$\boldsymbol{X}_P = \boldsymbol{X}_E + \boldsymbol{E}_P. \tag{7}$$

Given the embedding matrix $\boldsymbol{X}_P$, the multi-head self-attention layer projects it onto three different subspaces, which can be represented as

$$\{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\} = \{\boldsymbol{X}_P \boldsymbol{W}^Q, \boldsymbol{X}_P \boldsymbol{W}^K, \boldsymbol{X}_P \boldsymbol{W}^V\}, \tag{8}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$ are three projection matrices, which project $\boldsymbol{X}_P$ onto $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, respectively. The output of the multi-head self-attention layer can be formulated as

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}\,(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_h)\, \boldsymbol{W}^O, \tag{9}$$

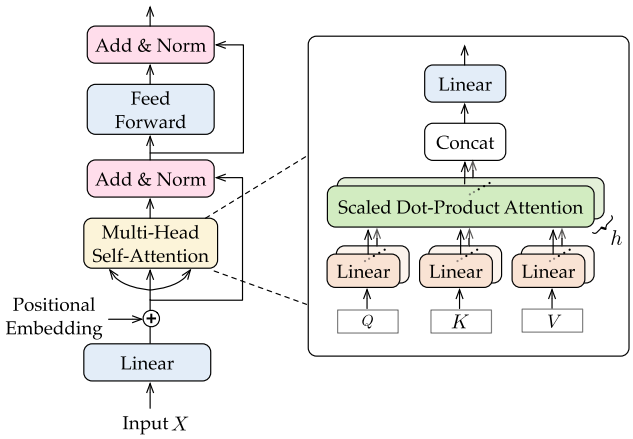


Fig. 6. The structure of the Transformer block.

where $\boldsymbol{H}_i, i \in \{1, 2, \ldots, h\}$ is the output of a single scaled dot-product attention head as shown in Figure 6, and h is the number of heads. $\boldsymbol{H}_i$ is computed by

$$\begin{aligned} \boldsymbol{H}_i &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right) \\ &= \text{Softmax}\left[\frac{\boldsymbol{Q}\boldsymbol{W}_i^Q \left(\boldsymbol{K}\boldsymbol{W}_i^K\right)^\top}{\sqrt{d_k}}\right]\boldsymbol{V}\boldsymbol{W}_i^V. \end{aligned} \tag{10}$$

For each attention head, the original inputs $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are further projected onto different subspaces via projection matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K$, and $\boldsymbol{W}_i^V$, so that different heads deal with different input to learn richer information [33]. The attention head then computes the similarity between the projected *query* and *key* via scaled dot-product and a softmax function is applied to obtain the weights on the projected *value*. The multi-head self-attention layer concatenates all the outputs $\boldsymbol{H}_i, i \in \{1, 2, \ldots, h\}$ from different heads and then reduces the high-dimension feature $\text{Concat}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_h)$ to low dimension via another projection matrix $\boldsymbol{W}^O$ as formulated in Equation (9).

However, as formulated in Equations (9) and (10), the vanilla Transformer [33] is extremely computationally expensive. Inspired by Swin Transformer [35], our encoder module introduces three extra mechanisms to further improve the efficiency while keeping the advantage of the Transformer [33]. The first one is the local attention mechanism. Instead of applying attention to all patches, we group the patches into $W \times W$ windows, and a multi-head self-attention layer is employed within each window. Such a mechanism is implemented as a **window multi-head self-attention (W-MSA)** layer. Local attention improves computation efficiency but lacks global perception. The second mechanism solves this problem by introducing shifted windows to partition the original layout pattern. Specifically, the local attention mechanism uses a regular window partitioning strategy that starts from the top left, while the shifted window mechanism displaces the windows by $(\lfloor \frac{H}{2} \rfloor, \lfloor \frac{W}{2} \rfloor)$. Figure 7 illustrates the cross-connection ability of the shifted windows mechanism. This mechanism is called a **shifted window multi-head self-attention (SW-MSA)** layer. Furthermore, the patch merging mechanism downscales the features by concatenating neighboring patches, which can enable the aggregation of multiscale information. Our encoder contains multiple stages, and Figure 8 presents a single stage, which consists of a patch merging layer and

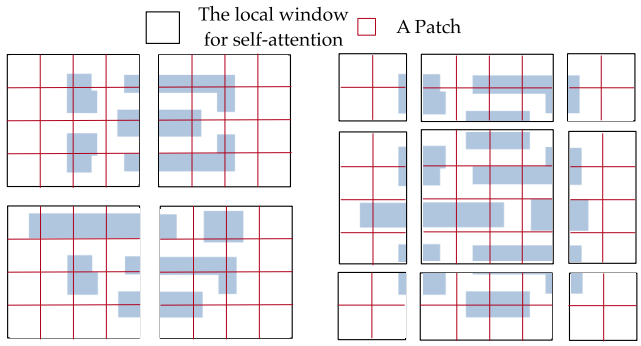


Fig. 7. The shifted window approach for computing self-attention in our proposed encoder architecture. In layer W-MSA, we adopt a regular window partitioning scheme, as shown in the left part, and self-attention is computed within each local window. In layer SW-MSA, the window is shifted, as shown in the right part.

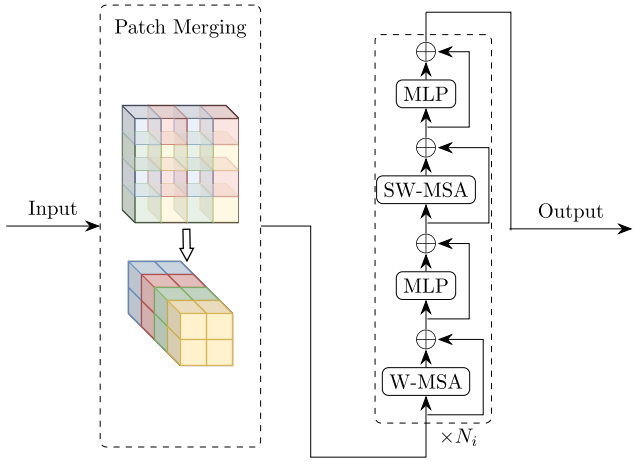


Fig. 8. Illustration of a stage in the Swin Transformer architecture, which consists of a patch merging layer and N_i Swin Transformer blocks.

N_i Swin Transformer blocks. Each Swin Transformer block includes W-MSA, MLP, SW-MSA, and MLP layers, where MLP means multi-layer perceptrons.

3.4 Decoder

The LUM decoder is designed based on the **feature pyramid network (FPN)** [36], which is a widely used and effective technique in the field of computer vision. FPN serves as a versatile feature extractor that capitalizes on the inherent and pyramidal hierarchy of features. This allows for the extraction of multi-level feature representations.

In our framework, the LUM decoder takes as input the output features obtained from the four stages of the LUM encoder depicted in Figure 9. These features serve as the foundation for the subsequent decoding process. To generate feature maps with varying levels of detail, the LUM decoder employs a top-down pathway. This pathway utilizes convolutional layers, residual connections, and a pyramid pooling module [37] to process the input feature maps. The result is the creation of multi-scale feature maps, each containing features at different levels of abstraction.

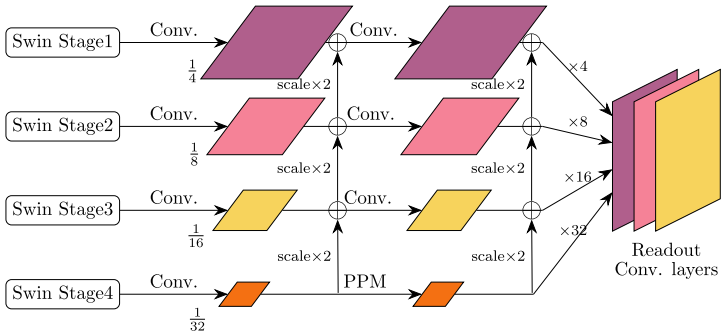


Fig. 9. LUM decoder. The inputs to the LUM decoder are the output features from the four stages of the LUM encoder. With a top-down pathway, multi-scale feature maps can be generated, each containing different level features.

In addition, lateral connections are introduced to integrate information from different levels of the feature hierarchy. These connections combine feature maps from different levels and facilitate the creation of a single consolidated feature map. This final feature map is then fed through readout convolutional layers to produce the ultimate output, which represents the generated layout.

3.5 Target Reconstruction

Our model is able to receive or output target images or optimized masks. For example, receiving the target image and generating the optimized mask is equivalent to the OPC. Receiving the masked target image and generating the restored target image is the layout reconstruction process. Combining different input and reconstruction targets enables our LUM to acquire knowledge of different domains.

To distinguish between target images and optimized masks, we design an additional type of embedding for the LUM encoder. Specifically, given the input embeddings $\boldsymbol{X}_E$, we apply the type embedding by

$$\boldsymbol{X}_P = \begin{cases} \boldsymbol{X}_E + \boldsymbol{E}_T, & \text{if the input is a target image,} \\ \boldsymbol{X}_E + \boldsymbol{E}_M, & \text{if the input is an optimized mask.} \end{cases} \tag{11}$$

Similarly, we also introduce the type embedding to the LUM decoder. The embedding matrix $\widetilde{\boldsymbol{E}}_T$ or $\widetilde{\boldsymbol{E}}_M$ is added to the input of the MLM decoder. We set the type embedding matrices $\boldsymbol{E}_T$, $\boldsymbol{E}_M$, $\widetilde{\boldsymbol{E}}_T$, and $\widetilde{\boldsymbol{E}}_M$ as trainable. This mechanism enables the encoder and decoder to differentiate between the two domains.

During the pre-training stage, when given a layout tile $\boldsymbol{X}$, we first obtain its corresponding target image $\boldsymbol{X}_T$ and optimized mask $\boldsymbol{X}_M$. Next, we mask the images and get $\overline{\boldsymbol{X}}_T$, $\overline{\boldsymbol{X}}_M$. Denoting the mask layout modeling with $\boldsymbol{f} : \mathbb{R}^{H\times W} \rightarrow \mathbb{R}^{H\times W}$, the reconstruction target can be formulated as

$$\begin{aligned} L_{cross}(\boldsymbol{X}) = \|\boldsymbol{f}_{TT}(\overline{\boldsymbol{X}}_T) - \boldsymbol{X}_T\|_2^2 + \|\boldsymbol{f}_{TM}(\overline{\boldsymbol{X}}_T) - \boldsymbol{X}_M\|_2^2 + \\ \|\boldsymbol{f}_{MT}(\overline{\boldsymbol{X}}_M) - \boldsymbol{X}_T\|_2^2 + \|\boldsymbol{f}_{MM}(\overline{\boldsymbol{X}}_M) - \boldsymbol{X}_M\|_2^2. \end{aligned} \tag{12}$$

In Equation (12), the two subscripts of $\boldsymbol{f}$ indicate the input and output domains. For example, $\boldsymbol{f}_{TM}$ uses $\boldsymbol{E}_T$ as the encoder-type embedding and $\widetilde{\boldsymbol{E}}_M$ as the decoder-type embedding.

3.6 Fine-tuning

In the fine-tuning phase, we adapt the pre-trained model to HSD and OPC tasks by freezing the parameters of the pre-trained LUM encoder, and then we fine-tune the OPC decoder and train

ALGORITHM 1: LUM Fine-tuning

Input: Pre-trained LUM encoder $\boldsymbol{P}_{encoder}$;
Output: Fine-tuned OPC decoder $\boldsymbol{P}_{opc}$ and HSD decoder $\boldsymbol{P}_{hsd}$

```
Fix P_encoder;
P_opc ← Initialize OPC decoder parameter;
for t = 1 to Epoch do
    Sample a train set from dataset 𝔻_𝕄;
    Train P_opc to minimize L_OPC;
end for
P_hsd ← Initialize HSD decoder parameter;
for t = 1 to Epoch do
    Sample a train set from dataset 𝔻_𝕄;
    Train P_hsd to minimize L_HSD;
end for
```

an HSD decoder. The whole fine-tuning process is illustrated in Algorithm 1. For OPC, we simply fine-tune the pre-trained mask layout modeling decoder to predict the optimized masks. Given the target image $\boldsymbol{X}_T$ and the ground-truth optimized mask $\boldsymbol{X}_M$, the loss function is defined as

$$L_{OPC}(\boldsymbol{X}_T, \boldsymbol{X}_M) = L_M(\boldsymbol{X}_T, \boldsymbol{X}_M) + L_{L2}(\boldsymbol{X}_T) + L_{PVB}(\boldsymbol{X}_T). \tag{13}$$

The loss function (Equation (13)) involves the following components:

— The similarity between the predicted mask and the optimized mask is minimized by

$$L_M(\boldsymbol{X}_T, \boldsymbol{X}_M) = \|\boldsymbol{f}_{TM}(\boldsymbol{X}_T) - \boldsymbol{X}_M\|_2^2. \tag{14}$$

— The L_2 loss minimizes the distance between the printed image and the target image, which is formulated as

$$L_{L2}(\boldsymbol{X}_T) = \|\boldsymbol{f}_Z(\boldsymbol{f}_{TM}(\boldsymbol{X}_T)) - \boldsymbol{X}_T\|_2^2. \tag{15}$$

The printed image $\boldsymbol{f}_Z(\boldsymbol{f}_{TM}(\boldsymbol{X}_T))$ is computed by the lithography simulation model, which is described by Equations (2) and (3).

— $L_{PVB}(\boldsymbol{X}_T)$ improves the robustness of the OPC results by minimizing the distance between the printed images at the maximum and minimum process corners. Specifically, the maximum and minimum process corners are modeled by lithography kernels $\boldsymbol{H}_{max}$ and $\boldsymbol{H}_{min}$, respectively. Given the printed images $\boldsymbol{f}_{Z_{max}}(\boldsymbol{f}_{TM}(\boldsymbol{X}_T))$ and $\boldsymbol{f}_{Z_{min}}(\boldsymbol{f}_{TM}(\boldsymbol{X}_T))$, the PVB loss is defined as

$$L_{PVB}(\boldsymbol{X}_T) = \|\boldsymbol{f}_{Z_{max}}(\boldsymbol{f}_{TM}(\boldsymbol{X}_T)) - \boldsymbol{f}_{Z_{min}}(\boldsymbol{f}_{TM}(\boldsymbol{X}_T))\|_2^2. \tag{16}$$

Since the objective of HSD is very different from the OPC task, a customized HSD decoder is specifically designed to output whether there exist hotspot regions in the layout patterns. As shown in Figure 10, the HSD decoder is composed of three consecutive convolution layers, which receive the output of the previous encoder and predict whether there exists a hotspot in each patch as discussed in Section 3.2. Therefore, the output of the HSD decoder is an $N \times N$ feature map, where each pixel value denotes the probability of the hotspot region. To guide the learning of the HSD decoder, we adopt a loss function based on the cross-entropy loss, which is defined as

$$L_{HSD}(\boldsymbol{X}_T) = \sum_{x=1}^{N}\sum_{y=1}^{N} -p_{xy}\log\hat{p}_{xy} - (1-p_{xy})\log(1-\hat{p}_{xy}), \tag{17}$$

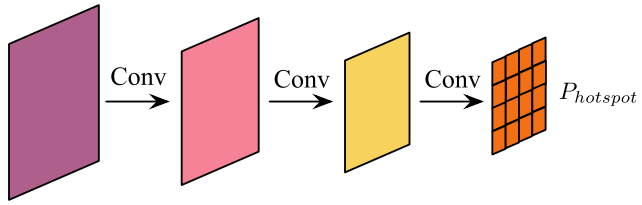


Fig. 10. HSD decoder.

Table 1. Design Rules [9] Followed by Our SynLayout Dataset

Design Rule	Min Size(nm)
M1 critical dimension	80
Pitch	140
Tip-to-tip Distance	60

where p_{xy} and $\hat{p}_{xy}$ denote the ground truth and prediction probability of the hotspot region, respectively. Once we have constructed the training data and completed the model training, those using our model to find hotspots in test data will no longer need to perform lithography simulations, greatly enhancing detection efficiency. Therefore, we only need to make a one-time effort to achieve a once-and-for-all process.

4 Experiments

We implement our entire framework LUM with the widely used deep learning library Pytorch [38]. The model is tested on a Linux system with a 2.3 GHz Intel Xeon CPU and a single NVIDIA GeForce RTX 3090 GPU. To verify the efficiency and robustness of our approach, we synthesized a large layout dataset, called SynLayout Dataset, to test the performance of LUM.

4.1 SynLayout Dataset

In previous works, the ICCAD 2012 [27] and ICCAD 2013 [39] benchmarks are two commonly used layout datasets for hotspot detection and mask optimization. However, these benchmarks are only used for academic research, and they contain very limited layouts. To be specific, ICCAD 2013 [39] only contains 10 $2\mu m \times 2\mu m$ metal layer clips, which are too small to fit AI solutions. To create a larger dataset, GAN-OPC [9] releases around 4k synthetic tiles of metal layer. However, its scale still cannot meet the training requirements of deep learning models currently applied in layout-related tasks. Previous learning models trained by these small datasets usually have unsatisfactory performance when applied in real industrial scenarios due to weak generalization ability.

In order to further improve the generalization ability of our model, we are supposed to generate and leverage more layout pattern data to train our LUM effectively. Therefore, we create a new layout dataset called SynLayout following the layout generation method proposed in [40]. The SynLayout dataset is based on the $32nm$ technology node, which is the same as the ICCAD 2013 benchmark [39]. We adjust the wire sizes to make sure the shapes in synthesized layouts are similar to those in the given benchmark. To generate experimental cells, all the shapes are randomly placed together based on a series of design rules as shown in Table 1. We synthesize 16,000 tiles, and we split 70% of the data for training and 30% of the data for testing. Figure 11 shows synthetic layout pattern examples that follow legal design rules.

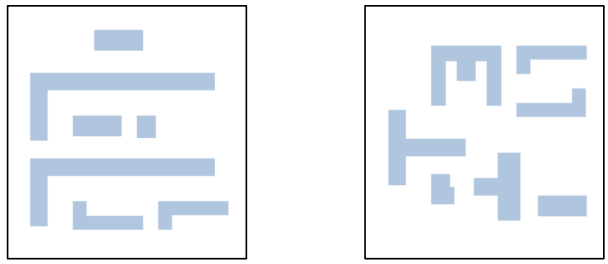


Fig. 11. Examples of synthetic layout patterns.

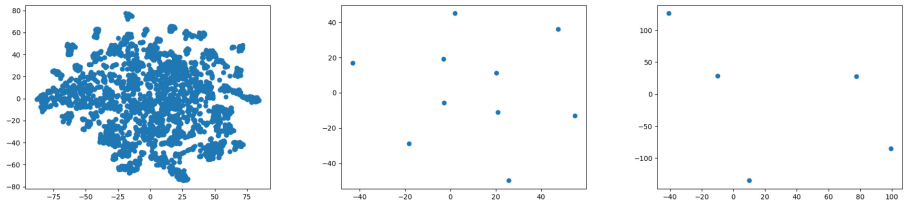
Table 2. Benchmark Information of Our SynLayout Dataset, ICCAD 2012 [27] Dataset, and ICCAD 2013 Dataset [39]

Benchmark	SynLayout	ICCAD 2012 [27]	ICCAD 2013 [39]
Layout Count	16000	10	5

In addition, we need to obtain the corresponding optimized masks and hotspot regions of all generated layout patterns, which act as supervised signals. Optimization-based ILT methods are adopted to obtain the accurate optimized masks of our generated layouts. L2 loss (Equation (15)), PVB loss (Equation (16)), and curvature loss [7] are employed in the ILT method to ensure the mask quality while minimizing the mask complexity. To find out hotspots that cause open and short circuits, we sample a series of points, calculate their EPE values, and identify locations with EPE violations as hotspot regions. According to the EPE violation count, there are 43,269 hotspots in training layouts and 18,120 hotspots in test layouts. Additionally, we compare the benchmark statistics between SynLayout and ICCAD 2012 [27], as well as ICCAD 2013 [39], in Table 2. In Figure 12, we utilize t-SNE to visualize the distribution of the three datasets. T-SNE is a statistical method for visualizing high-dimensional data by assigning each data point a location in a two-dimensional map. Based on Table 2 and Figure 12, it is evident that SynLayout surpasses both ICCAD 2012 [27] and ICCAD 2013 [39] in terms of layout number and diversity. Such a large dataset allows LUM to learn from diverse layout patterns and acquire the knowledge necessary for accurate downstream layout tasks, including both OPC and HSD. Although generating this training data may require additional effort, it plays a vital role in training a robust and effective model.

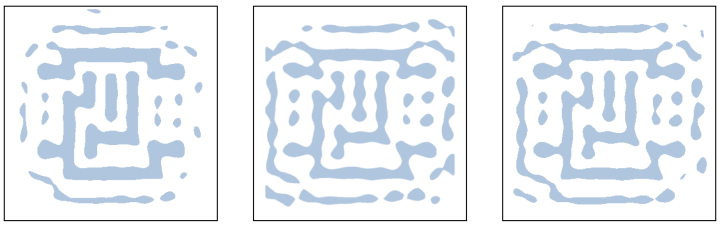
4.2 Results Comparison

Table 3 shows the comparison hotspot detection results of our proposed framework and several other state-of-the-art hotspot detectors. The comparison results illustrate that our model LUM has satisfactory performance. Specifically, the average accuracy of our framework is 94.21% compared to 91.94%, 89.76%, and 85.66% for ICCAD '21 [42], DAC '19 [25], and TCAD '19 [41], respectively. Besides, the advantage of our framework is that it suppresses the false alarm effectively, which decreases 55.2%, 30.0%, and 11.8% of the false alarm reported by TCAD '19 [41], DAC '19 [25], and ICCAD '21 [42], respectively. As for runtime, it on average takes 0.22 s for LUM to detect a hotspot on a single layout, which is a little bit slower than ICCAD '21 [42] (0.12 s) and faster than the other two works. This is because our model is designed for multiple tasks, including both hotspot detection and mask optimization, and the architecture of LUM is more complex than ICCAD '21 [42]. In contrast, the model in ICCAD '21 [42] is only used for hotspot detection and has a simple structure. However, we think the runtime is still comparable, and it is worth achieving much better performance at the cost of extra limited time.



(a) (b) (c)

Fig. 12. T-SNE of (a) SynLayout dataset, (b) ICCAD 2012 dataset, and (c) ICCAD 2013 dataset.



(a) (b) (c)

Fig. 13. Examples of OPC results. (a) shows a mask output by the pre-trained LUM without further training. (b) is a well-optimized mask from the "Ours-Exact" setting. (c) is a mask from the "Ours-Fast" setting which also shows satisfactory performance.

Table 3. Comparison with State-of-the-art HSD Methods on SynLayout Dataset

	Accuracy (%)	False Alarm	Runtime (s)
TCAD '19 [41]	85.63	6384	0.40
DAC '19 [25]	89.76	4629	0.28
ICCAD '21 [42]	91.94	3241	**0.12**
Ours	**94.21**	**2860**	0.22

4.3 What Does LUM Learn?

Our motivation for designing LUM is to incorporate prior knowledge of mask optimization into the model, which will improve the accuracy of hotspot detection. To prove that LUM has also learned the process of OPC, we concatenate the encoder and the OPC decoder and ask LUM to receive the target image and directly output the optimized masks without any fine-tuning process. Figure 13(a) presents an example of the pre-trained results. Figure 13(b) is the well-optimized mask output by our fine-tuned model and further refined using the optimization-based ILT method. Comparing Figure 13(a) to Figure 13(b), the generated masks without OPC-specific training are close to the well-optimized mask. This observation indicates that the pre-trained LUM can be equipped with OPC-based knowledge, which contributes to the outstanding HSD results.

To provide quantitative analysis, we further conduct OPC-specific training. The methods are tested on the ICCAD 2013 benchmark [39]. Tables 4 and 5 compare the mask printability and runtime performance of various OPC methods. To mask a fair comparison, we adopt the same optimization area as the works we compare with. The optimization region is the 1280×1280 region centered around the center of the layout. "MOSAIC" [43] is a pixel-based ILT method. "DevelSet" [7] is a level set-based ILT method that incorporates DNN models to reduce the

Table 4. Mask Printability Comparison with Sate-of-the-art Methods on ICCAD 2013 Benchmark

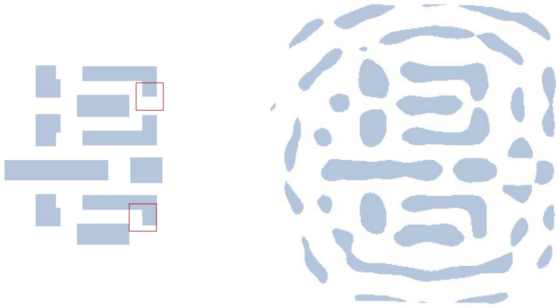
Benchmarks	MOSAIC [43]			DevelSet [7]			CTM-SRAF [44]			Multi-ILT [14]			Ours-Fast			Ours-Exact		
	EPE	L_2 (nm^2)	PVB (nm^2)	EPE	L_2 (nm^2)	PVB (nm^2)	EPE	L_2 (nm^2)	PVB (nm^2)	EPE	L_2 (nm^2)	PVB (nm^2)	EPE	L_2 (nm^2)	PVB (nm^2)	EPE	L_2 (nm^2)	PVB (nm^2)
case1	6	49,893	65,534	10	49,142	59,607	3	46,405	57,584	3	40,779	50,661	5	46,655	47,519	3	39,561	47,788
case2	10	50,369	48,230	1	34,489	52,010	1	33,481	45,756	2	34,201	44,322	0	33,893	37,084	0	31,261	38,279
case3	59	81,007	108,608	64	93,498	76,558	28	77,734	92,660	22	66,486	71,527	40	80,918	76,726	15	63,798	76,121
case4	1	20,044	28,285	2	18,682	29,047	0	13,183	26,061	0	10,942	21,500	1	12,014	22,039	0	8,939	23,679
case5	6	44,656	58,835	1	44,256	58,085	1	41,569	54,553	0	30,231	51,277	0	36,695	55,481	0	30,208	53,504
case6	1	57,375	48,739	2	41,730	53,410	1	38,608	48,134	0	30,741	44,982	0	37,338	47,979	0	30,284	47,809
case7	0	37,221	47,120	0	35,329	46,071	0	32,443	43,697	0	17,101	40,294	0	30,640	43,268	0	28,579	43,012
case8	2	19,782	22,846	0	15,460	24,836	1	15,178	20,657	0	11,935	20,357	0	12,751	20,535	0	10,813	20,192
case9	6	55,399	66,331	0	50,834	64,950	0	49,073	60,754	0	35,805	57,930	0	42,860	58,123	0	34,738	60,962
case10	0	24,381	18,097	0	10,140	21,619	0	8,231	17,426	0	8,825	18,470	0	10,323	16,544	0	7,714	16,234
Average	9.1	44,012	50,899	8	38,402	48,672	3.5	34,765	46,753	2.5	28,704.6	42,132	4.6	33,292	42,179	1.8	27,349	42,577

Table 5. Runtime Comparison with State-of-the-art Methods on ICCAD 2013 Benchmark

Benchmarks	MOSAIC [43]	DevelSet [7]	CTM-SRAF [44]	Multi-ILT [14]	Ours-Fast	Ours-Exact
case1	318	1.50	121	3.49	0.01	6.6
case2	256	1.40	93	3.47	0.01	6.6
case3	321	1.29	179	3.47	0.01	6.6
case4	322	1.65	128	3.47	0.01	6.6
case5	315	0.91	73	3.47	0.01	6.6
case6	314	0.84	72	3.47	0.01	6.6
case7	239	0.76	78	3.50	0.01	6.6
case8	258	1.14	66	3.47	0.01	6.6
case9	322	1.21	74	3.50	0.01	6.6
case10	231	0.42	57	3.48	0.01	6.6
Average	289	1.1	94.1	3.48	0.01	6.6

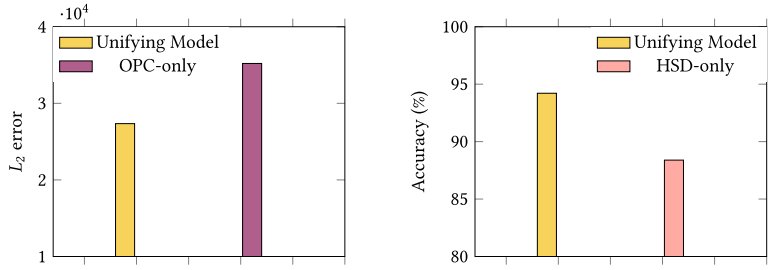
optimization time. "CTM-SRAF" [44] proposes a robust constraint-aware SRAF generation method based on **continuous transmission mask (CTM)**. Then, it co-optimizes the generated SRAF with the main target pattern using an ILT method to evaluate the effectiveness of SRAF. Multi-ILT [14] proposes a lithography simulation scheme working on multiple resolutions of a given mask, achieving satisfactory ILT acceleration. "Ours-Fast" shows the performance of the masks generated by our OPC decoder. Figure 13(c) shows a mask example from the "Ours-Fast" setting and is very close to Figure 13(b), which indicates that the "Our-Fast" version of the model also produces mask results with satisfactory performance. It can be seen that "Ours-Fast" achieves satisfactory L2 and PVB performance with a significantly lower runtime. With acceptable runtime overhead, "Ours-Exact" not only improves the L2 but also gets remarkable EPE results. The superior performance indicates that our LUM-based OPC can achieve better manufacturability. In summary, "Ours-Fast" achieves instant mask optimization with nice results, while "Ours-Exact" attains outstanding performance with acceptable runtime. Our method indeed has not surpassed Multi-ILT [14] in each metric, but overall, we think they are comparable. The primary goal of our work is to develop a pre-trained LUM that consolidates various layout tasks within a single framework. Thus, achieving results that are comparable with the state-of-the-art model, Multi-ILT [14], on mask optimization and surpassing other state-of-the-art works on hotspot detection can be regarded as the first step in this direction. Moving forward, we plan to investigate the creation of specialized decoder frameworks to enhance the performance of our model on individual tasks.

Finally, we also visualize the results in Figures 14(a) and 14(b) obtained by applying both the OPC decoder and the HSD decoder to the same layout, aiming to demonstrate that the detected hotspot regions correspond to the potential open-circuit-prone or short-circuit-prone regions in the optimized mask. It can be observed that as the light intensity increases, there is a significant



(a) (b)

Fig. 14. HSD and OPC results of one layout. (a) shows the hotspot detection result; (b) is the corresponding mask optimization result.



(a) (b)

Fig. 15. Ablation study results of different training strategies.

possibility of short-circuit occurrence at the highlighted regions on the optimized mask. This further demonstrates the accuracy of our LUM framework for both OPC and HSD.

4.4 Ablation Studies

LUM is trained with the intention of equipping it with the ability to understand the geometric information of layout patterns, enabling it to be applicable to various downstream layout tasks. To further verify the benefit of unifying hotspot detection and mask optimization, we conducted additional ablation studies by removing certain supervised loss signals. As shown in Figure 15(a), in the case of OPC, even with the same model architecture and training data, if the training is conducted with only the input target and output mask, the model's performance on OPC will deteriorate. It can be seen that the unifying model on average outperforms the OPC-only training model with a 28.7% reduction in L_2 error. For HSD, if the training is conducted with only the input target and output hotspot predictions, the model's performance on HSD will also degrade. As illustrated in Figure 15(b), compared with the HSD-only training model, the hotspot detection accuracy of the unifying model achieves 6.6% enhancements. Therefore, such ablation studies can verify that during the pre-training phase, our proposed LUM is equipped with the ability to understand the layout geometry information, which further benefits the layout tasks, including both OPC and HSD.

In addition, we conduct ablation studies to test different masking ratios during the pre-training stage of our model. A suitable mask ratio can help LUM better learn and predict the masked parts. We compared the effects of 30%, 40%, 50%, 60%, and 70% masking ratios on the performance in the OPC and HSD tasks. The experimental results shown in Figure 16 indicate that a 50% masking ratio

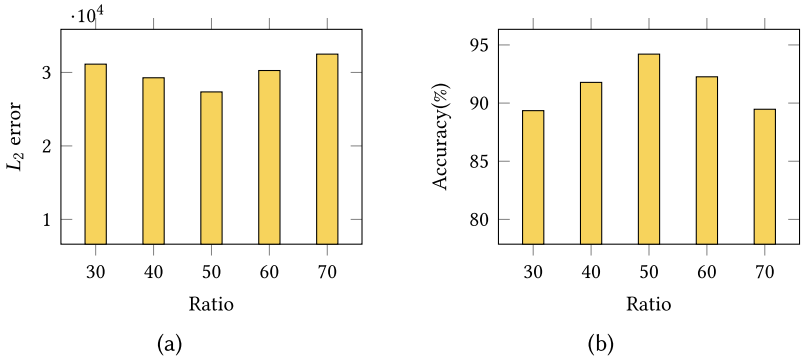


Fig. 16. Ablation study results of different masking ratios.

contributes to the best outcome. We notice that when the masking ratio is too low, although the model achieves high accuracy in layout reconstruction during the pre-training phase, its generalization ability is insufficient. Conversely, when the masking ratio is too high, the model's accuracy in layout reconstruction is compromised, preventing it from accurately understanding the layout geometry information.

5 Conclusion

In this work, we present a LUM, a deep-learning-based model that achieves remarkable performance on both mask optimization and hotspot detection tasks. Our model structure is inspired by the mask modeling method, which masks a portion of input signals and asks the model to restore the content. Such a model can leverage more training data to improve the generalization ability. We propose different decoders for HSD and OPC tasks while maintaining the same encoder component. During the training stage, we design multiple reconstruction tasks to enable the model to effectively learn the geometric information of layout patterns as well as the process of mask optimization. The experimental results demonstrate that our model has shown remarkable performance on OPC tasks, and the learned mask optimization knowledge is also beneficial for improving the HSD performance. We hope to use this work to provide an approach for the layout pre-training model. In the future, we will continue to explore how to design task-specific decoder frameworks to further improve the model's performance on each task.

Acknowledgments

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. RFS2425-4S02 and No. CUHK14211824).

References

[1] Rayleigh. 1879. XXXI. Investigations in optics, with special reference to the spectroscope. *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 8, 49 (1879), 261–274.

[2] Jhih-Rong Gao, Xiaoqing Xu, Bei Yu, and David Z. Pan. 2014. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *ACM/IEEE Design Automation Conference (DAC'14)*. 52:1–52:6.

[3] Vivek Bakshi. 2009. EUV lithography. (2009).

[4] Yijiang Shen, Ngai Wong, and Edmund Y. Lam. 2009. Level-set-based inverse lithography for photomask synthesis. *Optics Express* 17, 26 (2009), 23690–23701.

[5] Yijiang Shen, Ningning Jia, Ngai Wong, and Edmund Y. Lam. 2011. Robust level-set-based inverse lithography. *Optics Express* 19, 6 (2011), 5511–5521.

[6] Yuzhe Ma, Jhih-Rong Gao, Jian Kuang, Jin Miao, and Bei Yu. 2017. A unified framework for simultaneous layout decomposition and mask optimization. In *2017 IEEE/ACM International Conference on Computer-aided Design (ICCAD'17)*. 81–88. DOI: http://dx.doi.org/10.1109/ICCAD.2017.8203763

[7] Guojin Chen, Ziyang Yu, Hongduo Liu, Yuzhe Ma, and Bei Yu. 2021. DevelSet: Deep neural level set for instant mask optimization. In *IEEE/ACM International Conference on Computer-aided Design (ICCAD'21)*.

[8] Ziyang Yu, Guojin Chen, Yuzhe Ma, and Bei Yu. 2021. A GPU-enabled level set method for mask optimization. In *IEEE/ACM Proceedings on Design, Automation and Test in Europe (DATE'21)*.

[9] Haoyu Yang, Shuhe Li, Zihao Deng, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young. 2019. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 39, 10 (2019), 2822–2834.

[10] Bentian Jiang, Lixin Liu, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young. 2021. Neural-ILT 2.0: Migrating ILT to domain-specific and multitask-enabled neural network. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 41, 8 (2021), 2671–2684.

[11] Binwu Zhu, Su Zheng, Ziyang Yu, Guojin Chen, Yuzhe Ma, Fan Yang, Bei Yu, and Martin D. F. Wong. 2023. L2O-ILT: Learning to optimize inverse lithography techniques. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 43, 3 (2023), 944–955.

[12] Xiaoxiao Liang, Yikang Ouyang, Haoyu Yang, Bei Yu, and Yuzhe Ma. 2023. RL-OPC: Mask optimization with deep reinforcement learning. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 43, 1 (2023), 340–351.

[13] Qijing Wang, Bentian Jiang, Martin D. F. Wong, and Evangeline F. Y. Young. 2022. A2-ILT: GPU accelerated ILT with spatial attention mechanism. In *ACM/IEEE Design Automation Conference (DAC'22)*. 967–972.

[14] Shuyuan Sun, Fan Yang, Bei Yu, Li Shang, and Xuan Zeng. 2023. Efficient ILT via multi-level lithography simulation. In *ACM/IEEE Design Automation Conference (DAC'23)*.

[15] Guojin Chen, Hao Geng, Bei Yu, and David Z. Pan. 2024. Open-source differentiable lithography imaging framework. In *DTCO and Computational Patterning III*, Vol. 12954. SPIE, 118–127.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM* 63, 11 (2020), 139–144.

[17] Andrew B. Kahng, Chul-Hong Park, and Xu Xu. 2006. Fast dual graph based hotspot detection. In *Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE)*, Vol. 6349, 2006.

[18] Wan-Yu Wen, Jin-Cheng Li, Sheng-Yuan Lin, Jing-Yi Chen, and Shih-Chieh Chang. 2014. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 33, 11 (2014), 1671–1680.

[19] Yen-Ting Yu, Ya-Chung Chan, Subarna Sinha, Iris Hui-Ru Jiang, and Charles Chiang. 2012. Accurate process-hotspot detection using critical design rule extraction. In *ACM/IEEE Design Automation Conference (DAC'12)*. 1167–1172.

[20] Bei Yu, Jhih-Rong Gao, Duo Ding, Xuan Zeng, and David Z. Pan. 2015. Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering. *Journal of Micro/Nanolithography, MEMS, and MOEMS* 14, 1 (2015), 011003.

[21] Tetsuaki Matsunawa, Jhih-Rong Gao, Bei Yu, and David Z. Pan. 2015. A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In *Design-process-technology Co-optimization for Manufacturability IX*, Vol. 9427. SPIE, 201–211.

[22] Haoyu Yang, Jing Su, Yi Zou, Bei Yu, and Evangeline F. Y. Young. 2017. Layout hotspot detection with feature tensor generation and deep biased learning. In *ACM/IEEE Design Automation Conference (DAC'17)*. 62:1–62:6.

[23] H. Y. Yang, Shuhe Li, Cyrus Tabery, Bingqing Lin, and Bei Yu. 2020. Bridging the gap between layout pattern sampling and hotspot detection via batch active learning. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 40, 7 (2020), 1464–1475. DOI: http://dx.doi.org/10.1109/TCAD.2020.3015903

[24] Hao Geng, Haoyu Yang, Lu Zhang, Jin Miao, Fan Yang, Xuan Zeng, and Bei Yu. 2020. Hotspot detection via attention-based deep layout metric learning. In *IEEE/ACM International Conference on Computer-aided Design (ICCAD'20)*.

[25] Ran Chen, Wei Zhong, Haoyu Yang, Hao Geng, Xuan Zeng, and Bei Yu. 2019. Faster region-based hotspot detection. In *ACM/IEEE Design Automation Conference (DAC'19)*. 146:1–146:6.

[26] Yiyang Jiang, Fan Yang, Bei Yu, Dian Zhou, and Xuan Zeng. 2020. Efficient layout hotspot detection via binarized residual neural network ensemble. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 40, 7 (2020), 1476–1488.

[27] Andres J. Torres. 2012. ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *IEEE/ACM International Conference on Computer-aided Design (ICCAD'12)*. 349–350.

[28] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 16000–16009.

[29] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI, 2018.

[30] Zekai Chen, Devansh Agarwal, Kshitij Aggarwal, Wiem Safta, Mariann Micsinai Balan, and Kevin Brown. 2023. Masked image modeling advances 3D medical image analysis. In *Winter Conference on Applications of Computer Vision (WACV'23)*. 1970–1980.

[31] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'19)*. 4171–4186.

[32] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: A simple framework for masked image modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 9653–9663.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Annual Conference on Neural Information Processing Systems (NIPS'17)*. 5998–6008.

[34] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. n.d.. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE International Conference on Computer Vision (ICCV'21)*. 10012–10022.

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 2117–2125.

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 2881–2890.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[39] Shayak Banerjee, Zhuo Li, and Sani R. Nassif. 2013. ICCAD-2013 CAD contest in mask optimization and benchmark suite. In *IEEE/ACM International Conference on Computer-aided Design (ICCAD'13)*.

[40] Su Zheng, Haoyu Yang, Binwu Zhu, Bei Yu, and Martin Wong. 2024. LithoBench: Benchmarking AI computational lithography for semiconductor manufacturing. *Annual Conference on Neural Information Processing Systems (NeurIPS)* 36 (2024).

[41] Haoyu Yang, Jing Su, Yi Zou, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young. 2019. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 38, 6 (2019), 1175–1187.

[42] Binwu Zhu, Ran Chen, Xinyun Zhang, Fan Yang, Xuan Zeng, Bei Yu, and Martin D. F. Wong. 2021. Hotspot detection via multi-task learning and transformer encoder. In *IEEE/ACM International Conference on Computer-aided Design (ICCAD'21)*. IEEE, 1–8.

[43] Jhih-Rong Gao, Xiaoqing Xu, Bei Yu, and David Z. Pan. 2014. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *ACM/IEEE Design Automation Conference (DAC'14)*.

[44] Ziyang Yu, Peiyu Liao, Yuzhe Ma, Bei Yu, and Martin D. F. Wong. 2023. CTM-SRAF: Continuous transmission mask-based constraint-aware sub resolution assist feature generation. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)* 42, 10 (2023), 3402–3411.

Received 1 November 2024; revised 24 March 2025; accepted 28 March 2025