DAMO: Deep Agile Mask Optimization for Full Chip Scale

Guojin Chen
Chinese University of Hong Kong
glchen@cse.cuhk.edu.hk

Wanli Chen
Chinese University of Hong Kong
wlchen@cse.cuhk.edu.hk

Yuzhe Ma
Chinese University of Hong Kong
yzma@cse.cuhk.edu.hk

Haoyu Yang
Chinese University of Hong Kong
hyyang@cse.cuhk.edu.hk

Bei Yu
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

Abstract

Continuous scaling of the VLSI system leaves a great challenge on manufacturing, thus optical proximity correction (OPC) is widely applied in conventional design flow for manufacturability optimization. Traditional techniques conduct OPC by leveraging a lithography model but may suffer from prohibitive computational overhead. In addition, most of them focus on optimizing a single and local clip instead of addressing how to tackle the full-chip scale. In this paper, we present DAMO, a high performance and scalable deep learning-enabled OPC system for full-chip scale. It is an end-to-end mask optimization paradigm that contains a deep lithography simulator (DLS) for lithography modeling and a deep mask generator (DMG) for mask pattern generation. Moreover, a novel layout splitting algorithm customized for DAMO is proposed to handle full-chip OPC problem. Extensive experiments show that DAMO outperforms state-of-the-art OPC solutions in both academia and industrial commercial toolkit.

1 Introduction

Continuously shrinking down of the VLSI system has brought inevitable lithograph proximity effects and hence results in a degradation on manufacturing yield [1]. Optical proximity correction (OPC) compensates lithography proximity effects by adding assistant features and moving design edge segments inward or outward [2]. Mainstream OPC solutions include rule-based OPC [3], model-based OPC [4–6], inverse lithography technique (ILT)-based OPC [7, 8], and machine/deep learning-based OPC [9–11].

Kuang *et al.* [4] presented a model-based OPC for faster convergence and achieved good EPE with minor PV Band overhead using multi-stage SRAF insertion and OPC. Gao *et al.* [7] tackled the mask optimization problem by solving an ILT formulation, which descends the gradient of wafer-target error over input masks. The pixel-based optimization of ILT solution makes them robust to process variations. The generality of ILT also enables simultaneous mask optimization and layout decomposition as introduced in [8, 12]. These methods, to some extent, improve OPC from quality, robustness, and efficiency.

ICCAD '20, November 2–5, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415705

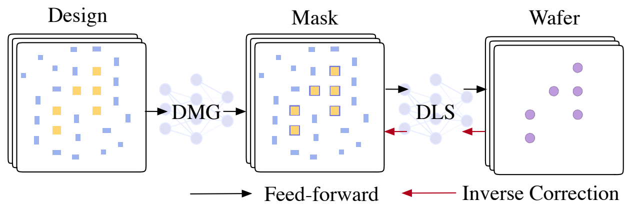

Figure 1: Overview of our proposed DAMO framework, which consists of two deep networks: deep mask generator (DMG) and deep lithography simulator (DLS). The OPC process is completed by utilizing the inverse correction gradient back-propagated from the DLS stage (red arrows).

The great development of machine learning algorithms has demonstrated the potential of applying artificial intelligence to benefit modern OPC flows. On one hand, machine learning-guided mask optimization targets to directly generate masks that are close to an optimal status and only fewer fine-tune steps using traditional OPC engines are required to obtain the final mask. Yang *et al.* [9] proposed GAN-OPC which grasps the advantage of generative machine learning models that can learn a design-to-mask mapping and provides better initialization of the ILT engine. On the other hand, machine learning-based lithography simulation aims to speed-up OPC flows by replacing costly lithography simulation with efficient learning models. Jiang *et al.* [10] applied an XGBoost [13] learning model to predict EPE at certain OPC control points that can guide the adjustment of shape edges. Instead of predicting wafer image errors, Ye *et al.* [14] proposed LithoGAN to build a generative learning model that directly predicts lithography contours. However, LithoGAN only targets a single shape within a clip, which strictly limits its usage in general OPC tasks.

There are several issues in previous methods. Firstly, the model-based/ILT inevitably methods require massive calls of the costly lithography simulation and the mask optimization, both of which are time-consuming. Secondly, all the previous works in machine learning-guided OPC limit the single-clip input layout into a low-resolution such as 256×256 pixel image. They are all exhibiting drawbacks that have still to go through traditional OPC engines in final steps due to the low-resolution limits. Since the resolution loss is intolerable in OPC, the usage scenarios of previous work in machine learning-guide OPC are limited. And worse still, the machine learning-based single-clip OPC is not practical. Thirdly, despite a variety of methods have been proposed, most of them focused on how to optimize a given single clip, and rarely discussed how to tackle the OPC problem in a view of the full-chip scale. For full-chip OPC tasks, the biggest barrier to conventional methods is the runtime overhead. Pang *et al.* [15] presented D2S to create full-chip ILT in a single day with giant GPU/CPU pairs, which consumes a large amount of resources on the handcrafted hardware and software. The learning-based methods, to the best of our knowledge, have not achieved any progress on full-chip mask optimization due to the dataset limitation and the low wafer pattern fidelity.

To address these concerns, we present DAMO, a unified OPC engine that is equipped with high-resolution GANs for full-chip scale. Deep convolution GANs (DCGAN) [16] has been demonstrated to be successful in generating high-resolution images. In DAMO, we designed DCGAN-HD which is customized from DCGAN with a high-resolution generator and multi-scale discriminators with perceptual losses. Then we design a deep lithography simulator (DLS) based on DCGAN-HD that takes the input of mask and generates the lithography contours faster with similar contour quality compared to legacy lithography simulation process. The DLS design also enables a unified neural network-based OPC framework where another deep mask generator (DMG) engine is trained along with the gradient back-propagated from DLS, which allows direct output of optimized high-quality masks (as shown in Figure 1). We further propose a stitchless full-chip splitting algorithm, with which we can perform full-chip OPC tasks efficiently with a few GPU resources. Our main contributions are as follows:

- We design DCGAN-HD, a very competitive high-resolution feature extractor (1024×1024) by redesign the generator and discriminator of DCGAN.
- We build up DLS and DMG based on DCGAN-HD. DLS is expected to conduct high-resolution lithography simulation. By training along with the inverse correction from DLS, DMG can directly generate high-quality masks.
- We develop an efficient stitchless full-chip splitting algorithm to apply DAMO on a layout of any size.
- We compare our proposed framework with state-of-the-art commercial tool Calibre [17]: $5\times$ speed-up in single-clip OPC tasks and $1.3\times$ acceleration in full-chip OPC tasks, while maintaining an even better solution quality.

The rest of the paper is organized as follows: Section 2 introduces terminologies and evaluation metrics related to this work. Section 3 details the proposed DAMO architecture. Section 4 shows the data preparation and DAMO training procedure, while Section 5 provides the full-chip splitting algorithm. Section 6 details experimental results and followed by conclusion in Section 7.

2 Preliminaries

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

2.1 cGAN Basis

cGAN is the short for conditional Generative Adversarial Networks [18, 19], which resembles classical GANs [20] that consists of a generator and a discriminator. The generator is trained to generate patterns follow some distribution such that the discriminator cannot identify whether these data comes from the generator or the training dataset. cGAN differs from GANs by certain constraints such that inputs and outputs of the generator can have stronger beneath connections. Representative cGAN applications in VLSI include GAN-OPC [9] and LithoGAN [14]. The former is designed for layout mask synthesis and the latter focuses on lithography contour prediction of the single via/contact shapes.

2.2 Problem Formulation

We introduce the following terms and evaluation metrics for the DAMO framework.

Definition 1 (mIoU). Given two shapes P and G, the IoU between P and G is $IoU(P, G) = P \cap G / P \cup G$. The mIoU is mean IoU.

Definition 2 (Pixel Accuracy). Pixel accuracy (pixAcc) is defined as the percentage of pixels that are correctly classified on an image.

Additionally, we have two evaluation metrics to measure mask quality following [9]. The squared L_2 error measures the quality of a mask under nominal process conditions, while PV Band measures the robustness of the generated mask under variations.

Definition 3 (Squared L_2 Error). Let $\boldsymbol{w}$ and $\boldsymbol{y}$ as design image and wafer image respectively, the squared L_2 error is calculated by $||\boldsymbol{w} - \boldsymbol{y}||_2^2$.

Definition 4 (PV Band). Given the lithography simulation contours under a set of process conditions, the PV Bands is the area among all the contours under these conditions.

With these definitions and evaluation metrics, the problem of mask optimization is defined as follows:

Problem 1 (Mask Optimization). Given a design image $\boldsymbol{w}$, the objective of mask optimization is generating the corresponding mask $\boldsymbol{x}$ such that remaining patterns $\boldsymbol{y}$ after lithography process is as close as $\boldsymbol{w}$ or, in other words, minimizing PV Band and squared L_2 error of lithography images.

3 DAMO Framework

The architecture overview of DAMO is illustrated in Figure 2. As the first part of DAMO, DLS aims to conduct an efficient and high-quality lithography simulation with the generative neural network model. Although LithoGAN [14] tries to alleviate the problem by embedding coordinate inputs, the scenario of application is strictly limited for a single via/contact shape, which is not practical in most cases. Therefore, DLS is developed as a customized cGAN for general-purpose lithography contour prediction tasks.

DMG is the second part of DAMO, which shares the identical architecture with DLS. The forward lithography process can be described with the following equation:

$$Z = f(\boldsymbol{M}). \tag{1}$$

The traditional ILT tries to obtain the optimal mask $\boldsymbol{M}_{opt}$ based on the given lithography model, which is presented as:

$$\boldsymbol{M}_{opt} = f^{-1}(Z_t), \tag{2}$$

where Z_t is the design pattern and $\boldsymbol{M}_{opt}$ is the optimized mask with OPC. In DAMO, we regard DLS as f in Equation (1). However, different masks may yield the same result, thus Equation (2) is an ill-posed problem. Previous mask optimizer GAN-OPC [21] generates masks by using cGAN to learn the mapping between the design and the mask pattern. Inspired from conventional ILT, our DMG steps further by not only learning mask patterns from training datasets but also being optimized by gradient back-propagated from the pre-trained DLS. After training, the generator of DMG performs inference to generate the solutions.

3.1 Improving Accuracy by Higher Resolution

Different from synthesizing photo-realistic images in computer vision tasks, the OPC task using generative models has its own properties. Intuitively, the layout in the OPC task has simpler patterns (mostly rectangles) but higher precision demands compared with image translation tasks. Moreover, the inputs of traditional image generation tasks are fixed-size images whose width or height is barely more than 2048 pixels. However, layouts contain thousands of via/contacts or SRAF patterns, whose area can reach more than 100×100 um^2. Previous work GAN-OPC [9] converts 1000×1000 nm^2 layout to 256×256 pixel images, which means 1-pixel shift error will cause an 8 nm shift in the output layout, making the results vulnerable for the industrial OPC tasks. To eliminate image transformation error, we set the input resolution of our model to be 1024×1024 pixels to contain the full 1024×1024 nm^2 layout. Combined with the window splitting algorithm which will be introduced in Section 6, DAMO framework can process input layout of any size, even the large full-chip layouts.

It is known that the adversarial training might be unstable and hard to converge for high-resolution image generation tasks, as mentioned in [16, 22, 23]. Therefore, we present DCGAN-HD, a new conditional GANs model qualified with high-resolution input images, which is the basic architecture of DLS and DMG.

3.2 DCGAN-HD: Solution for High Resolution

Previous work GAN-OPC is a conditional GAN framework for design to mask translation which consists of a generator G and a discriminator D. It adopts U-Net [24] as the generator with the input resolution of 256×256, We tested the GAN-OPC framework directly on high-resolution images and found the training is unstable and the generated results usually became empty. DCGAN [16] is one of the popular and successful network designs for cGAN allowing for higher resolution and deeper models. Based on DCGAN we present DCGAN-HD, a robust high-resolution conditional GAN model consisting of a newly designed generator, multi-scale discriminators, and a novel adversarial loss function. The architecture is illustrated in Figure 2.

3.2.1 *High-resolution Generator for DCGAN-HD.*

The left part of Figure 2 shows the high-resolution generator. In DLS part, the generator of DCGAN-HD resembles lithography simulation which requires mask-to-wafer mapping. In DMG part, with the gradient backpropagated from DLS, the generator focus on synthesizing the mask patterns from design and SRAF pattern groups.

UNet++ Backbone. Previous work [9] and [14] adopt traditional UNet [24] for mask generation. Input features are downsampled multiple times. With the decreasing of feature resolution, it is easier for a network to gather high-level features such as context features while low-level information such as the position of each shape becomes harder to collect. However, in OPC tasks, low-level information matters more than in the common computer vision tasks. For example, the shape and relative distance of design or SRAF patterns must remain unchanged after the deep mask optimization or deep lithography process. The number and relative

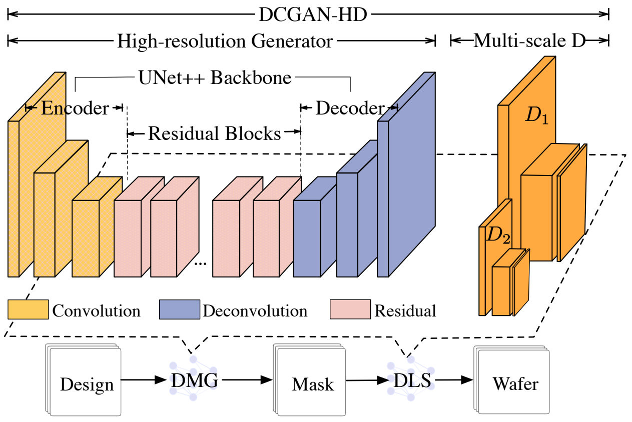

Figure 2: Architecture of DCGAN-HD with high-resolution generator and multi-scale discriminators, used in both DMG and DLS.

distance of via patterns in an input layout have a crucial influence on the result. The features of OPC datasets determine the vital importance of the low-level features. UNet++ [25] is hence proposed for better feature extraction by assembling multiple UNet that have different numbers of downsampling operations. It redesigns the skip pathways to bridge the semantic gap between the encoder and decoder feature maps, contributing to the more accurate low-level feature extraction. The dense skip connections on UNet++ skip pathways improve gradient flow in high-resolution tasks. Although UNet++ has a better performance than UNet, it is not qualified to be the generator of DCGAN-HD. For further improvement, we manipulate the UNet++ backbone with the guidelines suggested in DCGAN [16]. We will show later that our high-resolution generator outperforms UNet and UNet++ by a large margin.

Residual blocks. Most importantly, following Johnson *et al.* [26] settings, a set of residual blocks are added at the bottleneck of UNet++, which has been proven successful in style transfer and high-resolution image synthesis tasks. Since in OPC tasks, most structures are shared in output and input images (design and SRAFs), residual connections make it easy for the network to learn the identity function, which is appealing in the mask generation process. Specifically, we use 9 residual blocks, each of which contains two 3×3 convolution layers and batch normalization layers.

3.2.2 *Multi-scale Discriminators for DCGAN-HD.*

The high-resolution input also imposes a critical challenge to the discriminator design. A simple discriminator that only has three convolutional layers with LeakyReLU [27] and Dropout [28] is presented. Since the patterns in OPC datasets have simple and homogeneous distribution, a deeper discriminator has a higher risk of over-fitting. Therefore, we simplify the discriminator by reducing the depth of the neural network. Meanwhile, a dropout layer is attached after each convolutional layer. We use 3×3 convolution kernels in generator for parameter-saving purposes and 4×4 kernels in discriminator to increase receptive fields.

However, during training, we find that the simple discriminator fails to distinguish between the real and the synthesized images when more via patterns occur in a window. Because when the number of via reaches 5 or 6 in a window, the via patterns will have larger impact on each other and the features become more complicated. Inspired by Wang *et al.* in pix2pixHD [23], we design multi-scale discriminators. Different from pix2pixHD [23] that using three discriminators, our design uses two discriminators that have an identical network structure but operate at different image scales, which is named $D1$, $D2$, as shown in the right part of Figure 2. Specifically, the discriminators $D1$, $D2$ are trained to differentiate real and synthesized images at the two different scales, 1024×1024 and 512×512, respectively, which helps the training of the high-resolution model easier. In our tasks, the multi-scale design also shows its strengths in flexibility. For example, when the training set has only one via in a window, we can use only $D1$ to avoid over-fitting and reduce the training time.

3.2.3 Perceptual Losses. Instead of using per-pixel loss such as L_1 loss or L_2 loss, we adopt the perceptual loss which has been proven successful in style transfer [26], image super-resolution and high-resolution image synthesis [23]. A per-pixel loss function is used as a metric for understanding differences between input and output on a pixel level. While the function is valuable for understanding interpolation on a pixel level, the process has drawbacks. For example, as stated in [26], consider two identical images offset from each other by one pixel; despite their perceptual similarity they would be very different as measured by per-pixel losses. More than that, previous work [16] shows L_2 Loss will cause blur on the output image. Different from per-pixel loss, perceptual loss function in Equation (3) compares ground truth image $\boldsymbol{x}$ with generated image $\hat{\boldsymbol{x}}$ based on high-level representations from pre-trained convolutional neural networks Φ, which is ideal in DAMO framework. In DLS part, since the wafer pattern is not a regular circle, it is meaningless to fit the exact border of a wafer on the pixel level, the ultimate goal is to generate a better mask with higher perceptual quality wafer, reflected in less L_2 error and smaller PV Band.

$$\mathcal{L}_{LP}^{G,\Phi}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \mathcal{L}_{L_1}(\Phi(\boldsymbol{x}),\Phi(\hat{\boldsymbol{x}})) = \mathbb{E}_{\boldsymbol{x},\hat{\boldsymbol{x}}}\left[\|\Phi(\boldsymbol{x})-\Phi(\hat{\boldsymbol{x}})\|_1\right], \tag{3}$$

4 Data preparation and training

In order to collect sufficient data for training, we develop a data generation pipeline that can generate infinite training data, with which our DCGAN-HD can be fully utilized to simulate the lithography process and generate high-quality mask patterns. The overall training procedure of DAMO can be divided into two parts which are depicted in Figure 4.

4.1 Building Training Set from Scratch

It takes five steps to generate a training image, including design generation, SRAF insertion (with design rule checking), OPC, lithography simulation and layout to image transformation.

Design a design pattern. Via patterns are obtained under the following constraints using a layout pattern generator [29]. Firstly, all via patterns (70×70 nm^2) are restricted in a 1024×1024 nm^2 window. Secondly, by changing the via density we can control the number of via patterns in a single window. The via patterns are grouped evenly by the via numbers for reducing the bias caused by the random distribution of training set.

SRAF insertion and DRC. Mentor Calibre [17] is applied to do the SRAF insertion and design rule checking. Since the design area

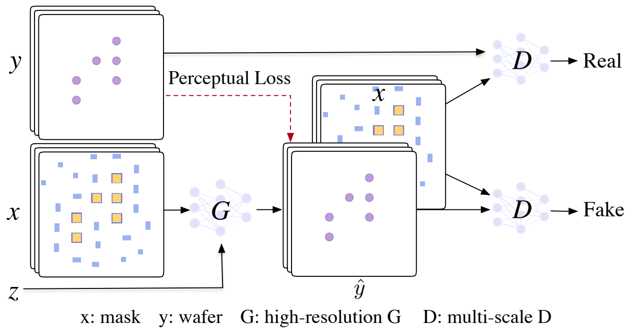

Figure 3: The training details of DLS, where the input images are mask-wafer pairs.

is 1024×1024 nm^2, it is possible that a few of SRAF patterns will be outside the design area when there are more than 2 via patterns. A larger window of 2048×2048 nm^2 will be used to capture all the SRAF patterns, which shares the same center as the design window.

OPC, litho-simulation, and image generation. We use masks and wafer patterns generated by Calibre as ground truth. Two sets of paired data are required for training. Mask-wafer pairs are generated to train DLS. After that, we align design-mask-wafer data for the OPC process. The obtained clips of size 2048×2048 nm^2 are converted into images with 2048×2048 pixels where 1nm represents 1 pixel. All the 2048×2048 pixels images will be centrally cropped into 1024×1024 pixels images where the design window locates before training. After training, the generated 1024 pixels images will be attached at the center of SRAF clip layer to form a 2048×2048 nm^2 layout before testing using Calibre. The crop-then-recover strategy saves the computational cost and improves the accuracy by focusing on the mask generation.

4.2 Training of DLS

Figure 3 shows the training process of our deep lithography simulator. As a customized design of cGAN, DLS is trained in an alternative scheme using paired mask image $\boldsymbol{x}$ and wafer image $\boldsymbol{y}$ obtained from Mentor Calibre. $\boldsymbol{z}$ indicates randomly initialized images.

The objectives of DLS include training the generator G that produces fake wafer images $G(\boldsymbol{x},\boldsymbol{z})$ by learning the feature distribution from $\boldsymbol{x}$–$\boldsymbol{y}$ pairs and training the discriminators D_1, D_2 to identify the paired $(\boldsymbol{x}, G(\boldsymbol{x},\boldsymbol{z}))$ as fake. This motivates the design of DLS loss function. The first part of the loss function comes from vanilla GAN that allows the generator and the discriminator interacting with each other in an adversarial way:

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\log D(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}[\log(1-D(\boldsymbol{x},G(\boldsymbol{x},\boldsymbol{z})))]. \tag{4}$$

Combined with our multi-scale discriminators described in Section 3.2.2, the Equation (4) can be modified as:

$$\begin{aligned}\sum_{k=1,2}\mathcal{L}_{cGAN}\left(G_{DLS},D_{DLS_k}\right) &= \sum_{k=1,2}\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\log D_{DLS_k}(\boldsymbol{x},\boldsymbol{y})] \\ &+ \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}[\log(1-D_{DLS_k}(\boldsymbol{x},G_{DLS}(\boldsymbol{x},\boldsymbol{z}))],\end{aligned} \tag{5}$$

where D_{DLS_k} is the kth discriminator of DLS. In DLS design, the perceptual loss is added to the objective, we denote $\hat{\boldsymbol{y}}$ as $G(\boldsymbol{x},\boldsymbol{z})$

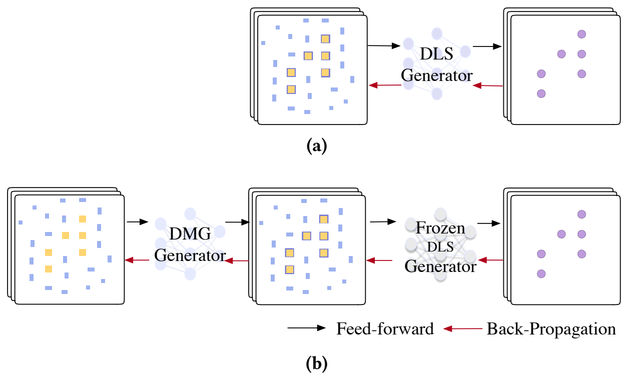

Figure 4: Overall training of DAMO: (a) Training DLS in the first stage; (b) Training DMG with fixed DLS generator in the second stage.

and loss network Φ is a pre-trained VGG19 on ImageNet. The perceptual loss is given by:

$$\begin{aligned}\mathcal{L}_{LP}^{G_{DLS},\Phi}(\boldsymbol{y},\hat{\boldsymbol{y}}) &= \sum_{j=1\dots5}\mathcal{L}_{L_1}(\phi_j(\boldsymbol{y}),\phi_j(\hat{\boldsymbol{y}}))\\ &= \sum_{j=1\dots5}\mathbb{E}_{\boldsymbol{y},\hat{\boldsymbol{y}}}\left[\|\phi_j(\boldsymbol{y})-\phi_j(\hat{\boldsymbol{y}})\|_1\right],\end{aligned} \tag{6}$$

where ϕ_j is the feature representation on j-th layer of the pre-trained VGG19 Φ. By combining Equation (5) and Equation (6):

$$\mathcal{L}_{DLS} = \sum_{k=1,2}\mathcal{L}_{cGAN}(G_{DLS},D_{DLS_k}) + \lambda_0\mathcal{L}_{LP}^{G_{DLS},\Phi}(\boldsymbol{y},\hat{\boldsymbol{y}}). \tag{7}$$

4.3 Training of DAMO

Here we introduce the overall training procedures of the whole framework. The first training step is illustrated in Figure 4(a), which is focusing on DLS. The proposed DLS is expected to predict wafer image with higher precision compared with traditional cGAN. After the training of DLS, all parameters in its generator are frozen.

The second training step is illustrated in Figure 4(b), which is focusing on DMG. DMG has the same architecture as DLS developed for DAMO training. In this stage, training data are switched to design-mask-wafer pairs. We use the design-mask to train DMG, obtaining an initial solution. The objective of DMG is shown in Equation (9) where $\boldsymbol{x}$ represents the ground truth mask, $\boldsymbol{w}$ is the corresponding design, and $\boldsymbol{z}_0$ is the image with random values. G_{DMG}, D_{DMG} represents the generator and discriminator of DMG. $\hat{\boldsymbol{x}}$ is the generated mask of G_{DMG}. Here DMG shares the same architecture as DLS, which yields a similar objective as Equation (7),

$$\begin{aligned}\sum_{k=1,2}\mathcal{L}_{cGAN}(G_{DMG},D_{DMG_k}) &= \sum_{k=1,2}\mathbb{E}_{\boldsymbol{w},\boldsymbol{x}}[\log D_{DMG_k}(\boldsymbol{w},\boldsymbol{x})]\\ &+\mathbb{E}_{\boldsymbol{w},\boldsymbol{z}_0}[\log(1-D_{DMG_k}(\boldsymbol{w},G_{DMG}(\boldsymbol{w},\boldsymbol{z}_0))].\end{aligned} \tag{8}$$

$$\mathcal{L}_{DMG} = \sum_{k=1,2}\mathcal{L}_{cGAN}(G_{DMG},D_{DMG_k}) + \lambda_1\mathcal{L}_{LP}^{G_{DMG},\Phi}(\boldsymbol{x},\hat{\boldsymbol{x}}). \tag{9}$$

Then we put the solution into DLS. RGB images instead of binary images are used because we can control the gradient of design, mask, and wafer separately, which is significant for avoiding noise points. Separating the design, mask, and SRAF into different channels makes DAMO more stable and flexible because we can apply different weights on different channels. After that, DLS calculates the perceptual loss between the generated wafer and the ground truth wafer. Finally, the gradient will be back-propagated to DMG to guide mask generation. Combining Equation (7) with Equation (9), the objective function of DAMO can be expressed as Equation (10),

$$\mathcal{L}_{DAMO} = \mathcal{L}_{DMG} + \mathcal{L}_{DLS} + \lambda_2\mathcal{L}_{L_1}(\hat{\boldsymbol{y}},\boldsymbol{w}_r). \tag{10}$$

We denote $\boldsymbol{w}_r$ as the via patterns (without SRAF). The last term in Equation (10) shows the superiority of our architecture, which bridges the gap between the generated wafer ($\hat{\boldsymbol{y}}$) and target design ($\boldsymbol{w}_r$) thus optimizing the mask directly. DAMO controls the whole flow from design to wafer while GAN-OPC relies on the conventional ILT engines.

Thanks to the guidance of DLS, our DAMO framework has a higher solution space than GAN-OPC. The success of our approach is also verified by various experiments. Compared to previous works, there are several advantages of DAMO:

- DLS surpasses LithoGAN [14] by being able to predict lithography contours of a single clip with multiple via patterns, which enables efficient training of DMG.
- DAMO, equipped with DCGAN-HD, can directly output manufacturing friendly masks that avoid further fine-tuning with traditional costly OPC engines.

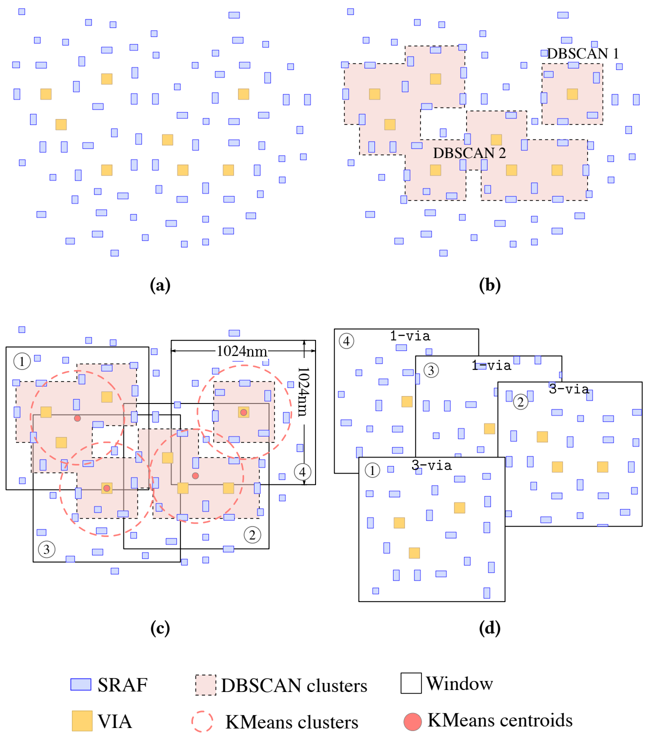

Figure 5: Two-step full-chip splitting algorithm: (a) Part of full-chip; (b) Coarse step: full-chip to DBSCAN clusters; (c) Fine step: run KMeans++ on each DBSCAN cluster to get KMeans clusters, where each KMeans cluster belongs to a $1024 \times 1024nm^2$ **window; (d) The split chips.**

5 Full-chip Splitting Algorithm

DAMO shows advantages on $1024 \times 1024 nm^2$ clips. To further adopt DAMO on full-chip layouts, a coarse-to-fine window splitting algorithm is proposed, in which the two-step clustering enables us to deal with full-chip industrial layouts where via patterns are distributed randomly with different local densities. A portion of one full-chip is shown in Figure 5(a).

Coarse step: DBSCAN. The main concept of the DBSCAN algorithm is to locate the regions of high via density that are separated from other low density regions. Any via neighborhood within a circle of radius Eps(ϵ) from via v will be assigned to the same cluster of v. DBSCAN algorithm is used to initially detect the clusters of via patterns (lines 1–4 in Algorithm 1). After the coarse step, the via patterns in a large layout will be assigned into different DBSCAN clusters, as shown in Figure 5(b).

Fine step: KMeans++. After DBSCAN clustering, every via pattern is assigned to a coarse cluster d which contains V via patterns. Then we search every coarse cluster and run KMeans++ algorithm to find the best splitting windows, where the max number of via patterns in a window is set to K (lines 5–27 in Algorithm 1). Note that every KMeans cluster belongs to a $1024 \times 1024 nm^2$ window, whose center locates at the centroid of the KMeans cluster, as shown in Figure 5(c).

Algorithm 1 Full-chip splitting algorithm.

```
Input: Full-chip, DBSCAN parameter $\epsilon$;
Output: Best full-chip splitting windows;
 1: $\mathcal{V} \leftarrow$ collection of all via patterns;            ▷ DBSCAN starts.
 2: $MinPts \leftarrow 1$;
 3: Run DBSCAN on $\mathcal{V}$ with parameters $\epsilon$ and $MinPts$;
 4: $\mathcal{D} \leftarrow$ collection of DBSCAN clusters.               ▷ DBSCAN ends.
 5: $\mathcal{S} \leftarrow$ empty collection of best splitting windows;  ▷ KMeans++ starts.
 6: $K \leftarrow$ max via number in a window;
 7: $H \leftarrow$ width and height of a window;
 8: for each $d \in \mathcal{D}$ do
 9:     $V \leftarrow$ via number in DBSCAN cluster $d$;
10:     for $\forall k < V$ do
11:         Run KMeans++ in cluster $d$ with $k$ centroids;
12:         $\mathcal{C} \leftarrow$ collection of KMeans clusters in DBSCAN cluster $d$;
13:         Create $H \times Hnm^2$ split windows centered at $k$ centroids;
14:         $BestSplitting \leftarrow$ True;
15:         for each KMeans cluster $c \in \mathcal{C}$ do
16:             $v_c \leftarrow$ via number of KMeans cluster $c$;
17:             if $v_c > K$ or via in $c$ is not in $k$ split windows then
18:                 $BestSplitting \leftarrow$ False;
19:                 Break;
20:             end if
21:         end for
22:         if $BestSplitting$ is True then
23:             Add the $k$ split windows to $\mathcal{S}$;
24:         end if
25:     end for
26: end for
27: return collection of best splitting windows $\mathcal{S}$;          ▷ KMeans++ ends.
```

After the coarse-to-fine clustering, the design will be split into many 1024×1024 nm^2 windows (see Figure 5(d)). Our coarse-to-fine splitting algorithm has many advantages. Firstly, it is extremely fast because DBSCAN only needs to scan the via patterns once and it also skips the empty areas. Secondly, the typical window-sliding method is hard to handle overlapping situations and stitching errors. In our algorithm, the overlapping situations and stitching errors will not occur, since every design pattern belongs to a fine cluster. Thirdly, because the window locates at the centroids of the clusters, the via patterns are all placed near the center of the windows, which reduces the search space of the machine learning model to a large extent, resulting in less training data and training time.

6 Experimental Results

Many experiments are carried out to evaluate our proposed framework. Firstly, we evaluate the effectiveness of our DLS by testing the mIoU and pixAcc of generated wafer patterns. Secondly, the superiority of our proposed DAMO is also validated by thorough experiments. Lastly, we test our model using the full-chip layout in ISPD 2019 contest [30], which is generated by an open-source router [31].

6.1 Dataset

Our training set and validation set. As described in Section 4.1, two sets of 2048×2048 pixels RGB images are generated for training purpose: one mask-wafer paired for DLS, while another one design-mask-wafer paired for DMG. To obtain fine-grained models, we divide our data depending on the via number with a window, and six groups marked as `1-via`, `2-via`, ... `6-via` are generated. For instance, the `1-via` group contains all cases with only one via in a window. Each group has 2000 training images and 500 validation images.

ISPD 2019 large full-chip test set. We use another real benchmark coming from ISPD 2019 Contest on Initial Detailed Routing, We take the layer 40 of `ispd19_test1` [30] as our design layer ($100 \times 100um$). After the SRAF insertion, OPC, and lithography process via Calibre, we extract the design, SRAF, mask, wafer layers and merge them to be the ground truth. Then, using our coarse-to-fine full-chip splitting algorithm, the full-chip layout is split to lots of $1024 \times 1024nm^2$ layout windows. According to the design rule, we set the DBSCAN radius Eps (ϵ) to be $400nm$. The hyper-parameters K in KMeans++ fine step is set to 5, because the images containing more than 5 design patterns only account for 0.5% in the total windows. The `ispd19_test1` benchmark contains 16035 design patterns which are split to 11649 windows. 6116 split windows marked as `ISPD-1-via` has only one via in a window, accounting for 52.5%. The detailed distribution of different windows is illustrated in Figure 6.

6.2 Implementation Details

The proposed DAMO is implemented in Python with PyTorch library [32]. Adam optimizer [33] is adopted, where we set base learning rate and momentum parameters to 0.0002 and (0.5, 0.999). In the LeakyReLU, the slope of the leak is set to 0.2 in all models. We set the batch size to be 4, and the maximum training epoch is 100. The weight parameters of λ_0, λ_1, and λ_2 are set to 100, 100, and 50, respectively. After training, the generated mask layer will be converted into GDSII layout file then fed into Mentor Calibre for lithography simulation validation. We use four Nvidia TITAN Xp GPUs for training and one for testing. The evaluation metrics

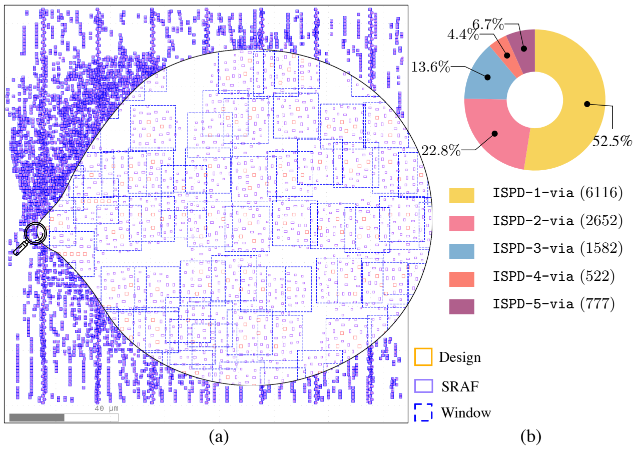

Figure 6: (a) ISPD 2019 large full-chip layout and splitting windows; (b)via window distribution in `ispd19_test1`[30].

we adopt are mIoU, pixAcc, L_2 error, and PV Band. Here the PV Band is calculated by Calibre.

6.3 Effectiveness of DLS

Before training DAMO, it is of great importance to construct a high-performance DLS. Since our DLS model is based on the cGAN framework, we set up an ablation experiment to illustrate the advantages of our generator and discriminators. The results shown in Table 1 is the average of 6 groups of validation set. Firstly, cGAN (used in LithoGAN) provides a baseline mIoU of 94.16% which is far away from practical application. Then, UNet++ is used to replace the UNet generator in cGAN for better performance. However, the original UNet++ is not qualified to be a generator of a cGAN and the mIoU is reduced to 93.98% (as shown in Table 1).

Following DCGAN, we made some amendments in UNet++ (as discussed in Section 3.2.1) and high resolution generator is adopted in our DLS model. After applying our high resolution generator, mIoU is improved to 97.63%, which outperforms UNet and UNet++ generators by a large margin when using the same discriminator. The huge gain in mIoU implies that our developed high resolution generator is a strong candidate for DLS. Next, the newly designed multi-scale discriminators (introduced in Section 3.2.2) are used to replace the original cGAN discriminator. Results in Table 1 show that mIoU is further boosted to 97.63%.

Lastly, we replace the L_1 loss with the perceptual loss proposed in Section 3.2.3 and the mIoU reaches 98.68%. Additionally, DLS can handle multiple vias in a single clip, which overcomes the limitation of LithoGAN [14].

6.4 Performance of DAMO

We test DAMO on the six groups of validation sets to verify the performance. Every generated mask will be pushed into Calibre for lithography simulation. After that, we apply L_2 and PV Band measurements to test the performance of different mask optimization methods. Note that since GAN-OPC fails to train on high-resolution input, the 1024×1024 input images are down-sampled to 256×256 pixels to train the model. After the inference

Table 1: Results of DLS

Generator	Discriminator	Loss	mIoU (%)	pixAcc (%)
UNet (cGAN)	D (cGAN)	L_1	94.16	97.12
UNet++	D (cGAN)	L_1	93.98	96.74
G (Our)	D (cGAN)	L_1	96.23	97.50
G (Our)	D (Our)	L_1	97.63	98.76
G (Our)	D (Our)	Our	**98.68**	**99.50**

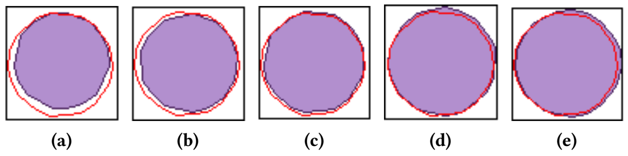

Figure 7: Visualization of DAMO model advancement on via layer: (a) Epoch 20; (b) Epoch 40; (c) Epoch 60; (d) Epoch 80; (e) Epoch 100.

process, the results are upsampled to the original size for L_2 and PV Band testing. Table 2 shows that on the validation set, DAMO has 2.7× less L_2 error and 1.3× less PV Band compared with GAN-OPC. In addition, DAMO outperforms Calibre in both L_2 and PV Band metrics, meanwhile achieving 4× speed-up. The L_2, the PV Band, and the runtime performance of DAMO are better than Calibre and GAN-OPC in all cases, which demonstrates that the stability of DAMO can be guaranteed.

The mask optimization process of DAMO is visualized in Figure 7. All the wafer images are generated using Calibre lithography simulation. The red contours represent wafer patterns on masks produced by Calibre while the purple wafers are on masks generated by DAMO. We sample DAMO results after 20/40/60/80/100 training epochs for the illustration. Initially, the wafer patterns of DAMO have lower quality compared with Calibre (as shown in Figure 7(a) and Figure 7(b)). Along with the increase of training epochs, the results of DAMO and Calibre are getting closer (Figure 7(c)). Figure 7(d) and Figure 7(e) show that the performance of DAMO surpasses Calibre after iterative optimization.

6.5 Results on ISPD 2019 Full-chip Layout

For ISPD 2019 large full-chip layout, the experiment has two stages. In the first stage, we test DAMO on the 11649 split windows, as listed in Table 3. We compare GAN-OPC, Calibre, and DAMO under metrics of L2, PV Band, and runtime. DAMO shows better performance against Calibre and GAN-OPC, on all metrics of L_2, PV Band, and runtime.

In the second stage, we recover all the split windows into the original 100×100 um^2 large full-chip layout with DAMO generated masks. Still, we use Calibre to test the L_2 error and PV Band of the large layout results. Figure 8 shows the sum of L2 error and PV band on split windows are very close to the results of full-chip layouts owing to our efficient splitting algorithm. As shown in Figure 8(a) and Figure 8(b), DAMO still has better performance than Calibre. For the runtime of the large full-chip layout (see Figure 8(c)), we separate runtime of DAMO to preparation time

Table 2: Comparison with State-of-the-art on validation set

Bench	case#	GAN-OPC			Calibre			DAMO		
		L_2 (nm^2)	PV Band (nm^2)	runtime (s)	L_2 (nm^2)	PV Band (nm^2)	runtime (s)	L_2 (nm^2)	PV Band (nm^2)	runtime (s)
`1-via`	500	1464	3064	321	1084	2918	1417	**1080**	**2917**	**284**
`2-via`	500	4447	6964	336	2161	5595	1406	**2129**	**5576**	**281**
`3-via`	500	8171	11426	317	3350	8286	1435	**3244**	**8271**	**285**
`4-via`	500	11659	14958	327	4331	10975	1477	**4263**	**10946**	**291**
`5-via`	500	15773	18976	318	5410	13663	1423	**5396**	**13640**	**279**
`6-via`	500	18904	22371	320	6647	15572	1419	**5981**	**15543**	**284**
Average		10069	12960	323	3831	9502	1430	**3682**	**9482**	**284**
Ratio		2.735	1.367	1.138	1.040	1.002	4.427	**1.00**	**1.00**	**1.00**

Table 3: Comparison on ISPD 2019 full-chip splitting windows

Bench	case#	GAN-OPC			Calibre			DAMO		
		L_2 (nm^2)	PV Band (nm^2)	runtime (s)	L_2 (nm^2)	PV Band (nm^2)	runtime (s)	L_2 (nm^2)	PV Band (nm^2)	runtime (s)
`ISPD-1-via`	6116	2367	3492	3963	1073	2857	18959	**1056**	**2848**	**3669**
`ISPD-2-via`	2652	5412	7126	1742	2232	5670	7537	**2172**	**5654**	**1591**
`ISPD-3-via`	1582	8792	13047	1021	3602	8276	4494	**3196**	**8127**	**949**
`ISPD-4-via`	522	12395	15015	341	4395	11051	1692	**4361**	**10987**	**313**
`ISPD-5-via`	777	16526	19147	495	5526	12305	2537	**4542**	**12251**	**466**
Average		9098	11565	1512	3365	8031	7043	**3065**	**7973**	**1397**
Ratio		2.968	1.451	1.082	1.098	1.007	5.041	**1.00**	**1.00**	**1.00**

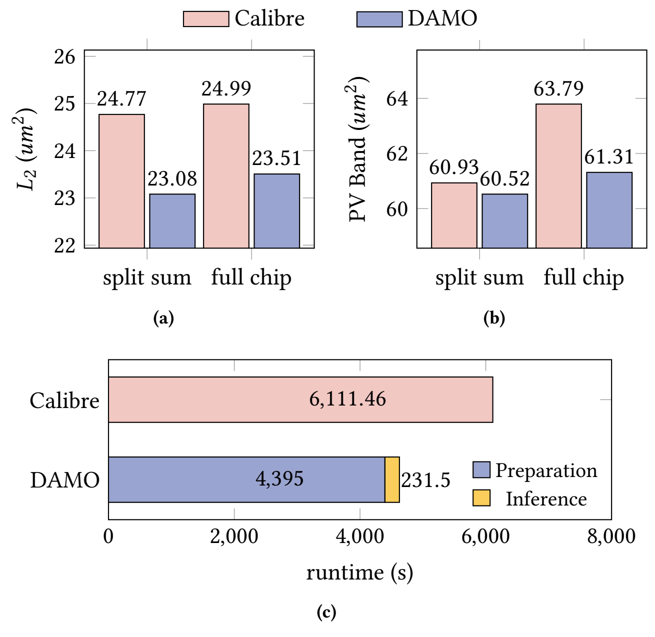

Figure 8: Comparison with Calibre on ISPD 2019 full-chip layout in terms of (a) L2; (b) PV Band; (c) runtime.

(4395s) and inference time (231.5s). The inference time takes only 5% of the total by parallel using four GPUs. Preparation includes the full-chip splitting, split layouts to images, generated images to layouts, and the split windows to full-chip recovering. All these preparation processes are running on a single CPU, which means the preparation time can be easily reduced when using multi CPUs in parallel.

7 Conclusion

In this paper, we present DAMO, an end-to-end framework targeting full-chip mask optimization with high-resolution generative machine learning models. The framework comes with DLS that offers precise lithography prediction benefiting from the proposed DCGAN-HD. The high-quality DLS also enables efficient training of DMG which hence promises to generate manufacturing friendly masks without further costly fine-tuning. The advantage of the proposed framework over the representative industrial and academic state-of-the-art demonstrates the possibility of deep neural networks as an alternative solution to many layout and mask optimization problems. Our future research includes the deployment of the framework to more complicated designs (such as metal layers) and the transfer-ability as technology node advances.

8 Acknowledgment

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14209420).

References

[1] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 10, pp. 1453–1472, 2013.

[2] H. Yang, W. Zhong, Y. Ma, H. Geng, R. Chen, W. Chen, and B. Yu, "VLSI mask optimization: From shallow to deep learning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2020, pp. 434–439.

[3] 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.

[4] J. Kuang, W.-K. Chow, and E. F. Y. Young, "A robust approach for process variation aware mask optimization," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2015, pp. 1591–1594.

[5] 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.

[6] T. Matsunawa, B. Yu, and D. Z. Pan, "Optical proximity correction with hierarchical bayes model," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 15, no. 2, p. 021009, 2016.

[7] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 52:1–52:6.

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

[9] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 131:1–131:6.

[10] B. Jiang, H. Zhang, J. Yang, and E. F. Young, "A fast machine learning-based mask printability predictor for OPC acceleration," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2019, pp. 412–419.

[11] H. Geng, W. Zhong, H. Yang, Y. Ma, J. Mitra, and B. Yu, "Sraf insertion via supervised dictionary learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[12] W. Zhong, S. Hu, Y. Ma, H. Yang, X. Ma, and B. Yu, "Deep learning-driven simultaneous layout decomposition and mask optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* ACM, 2016, pp. 785–794.

[14] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 107:1–107:6.

[15] L. Pang, E. V. Russell, B. Baggenstoss, M. Lee, J. Digaum, M.-C. Yang, P. J. Ungar, A. Bouaricha, K. Wang, B. Su *et al.*, "Study of mask and wafer co-design that utilizes a new extreme simd approach to computing in memory manufacturing: full-chip curvilinear ilt in a day," in *Photomask Technology 2019*, vol. 11148. International Society for Optics and Photonics, 2019, p. 111480U.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[17] Mentor Graphics, "Calibre verification user's manual," 2008.

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

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

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[21] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Y. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[22] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[23] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention.* Springer, 2015, pp. 234–241.

[25] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support.* Springer, 2018, pp. 3–11.

[26] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision.* Springer, 2016, pp. 694–711.

[27] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[29] H. Yang, W. Chen, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Automatic layout generation with applications in machine learning engine evaluation," *arXiv preprint arXiv:1912.05796*, 2019.

[30] "ISPD 2019 Contest on Initial Detailed Routing," http://www.ispd.cc/contests/19/#benchmarks.

[31] H. Li, G. Chen, B. Jiang, J. Chen, and E. F. Young, "Dr. cu 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–7.

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Workshop*, 2017.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.