GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets

Haoyu Yang, Shuhe Li, Zihao Deng, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young

***Abstract*—Mask optimization has been a critical problem in the VLSI design flow due to the mismatch between the lithography system and the continuously shrinking feature sizes. Optical proximity correction (OPC) is one of the prevailing resolution enhancement techniques (RETs) that can significantly improve mask printability. However, in advanced technology nodes, the mask optimization process consumes more and more computational resources. In this paper, we develop a generative adversarial network (GAN) model to achieve better mask optimization performance. We first develop an OPC-oriented GAN flow that can learn target-mask mapping from the improved architecture and objectives, which leads to satisfactory mask optimization results. To facilitate the training process and ensure better convergence, we propose a pre-training scheme that jointly trains the neural network with inverse lithography technique (ILT). We also propose an enhanced generator design with a U-Net architecture and a sub-pixel super-resolution structure that promise a better convergence and a better mask quality, respectively. At convergence, the generative network is able to create quasi-optimal masks for given target circuit patterns and fewer normal OPC steps are required to generate high quality masks. Experimental results show that our flow can facilitate the mask optimization process as well as ensure a better printability.**

I. Introduction

With the VLSI technology node continuously shrinking down, the mask optimization process becomes a great challenge for designers [1]–[4]. Conventional mask optimization process is illustrated in Fig. 1, where optical proximity correction (OPC) aims at compensating lithography proximity effects through correcting mask pattern shapes and inserting assist features. OPC methodologies include model-based techniques [5]–[7], [7]–[11] and inverse lithography-based technique (ILT) [12]–[15].

In model-based OPC flows, pattern edges are fractured into segments which are then shifted/corrected according to mathematical models. A high printability mask can then be obtained with sub-resolution assist features (SRAF) [16]. Awad *et al*. [5] propose a pattern fidelity aware mask optimization algorithm that optimizes core polygons by simultaneously shifting adjacent segmentations. Su *et al*. [7] significantly accelerate the OPC flow by extracting representative process corners while maintaining a good wafer image quality. However, model-based OPC flows are highly restricted by their solution space and hence lacking in reliability for complicated designs. On the other hand, ILTs minimize the error between the wafer image and the target with lithography constraints. Because ILTs conduct pixel-based optimization on layout masks, they are expected to offer better lithography contour quality, which although comes with additional challenges on mask manufacturability problems including manufacturing cost and mask design rules. Recently, Ma *et al*. [14] adopt ILT to simultaneously perform mask optimization and layout decomposition that brings a better solution of multiple patterning mask design. Although the model-based method and the ILT-based method behave well on a variety of designs, they take the wafer image as a mask update criterion in each iteration of the OPC process. In other works, multiple rounds of lithography simulation are indispensable in the optimization flow which is drastically time consuming.

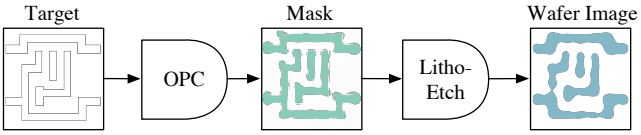


Fig. 1 Conventional OPC flow and lithography process, where OPC is very time consuming.

The explosion of machine learning techniques has dramatically changed the way to solve design for manufacturability problems. Recently, both shallow and deep learning models have been successfully utilized to estimate mask printability accurately and efficiently (e.g. [17]–[20]). There are also several attempts on mask optimization problems that contain more complex regression or classification procedures. Matsunawa *et al*. [21] conduct segment based pattern correction with hierarchical Bayes model. Gu *et al*. [22] introduce discrete cosine transform (DCT) features and linear regression to predict fragment movement. [23], [24] incorporate artificial neural networks to estimate mask patterns. However, existing machine learning models can only perform pixel-wise or segment-wise mask calibration that is not computationally efficient.

Generative adversarial networks (GANs) have shown powerful generality when learning the distribution of a given dataset [25]–[27]. The basic optimization flow of GAN contains two networks interacting with each other. The first one is called **generator** that takes random vectors as input and generates samples which are as much closer to the true dataset distribution as possible. The second one is called **discriminator** that tries to distinguish the true dataset from the generated samples. At convergence, ideally, the generator is expected to generate samples that have the same distribution

The preliminary version has been presented at the Design Automation Conference (DAC) in 2018. This work is supported in part by The Research Grants Council of Hong Kong SAR (Project No. CUHK24209017).

The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, NT, Hong Kong.

as true dataset. Inspired by the generative architecture and the adversarial training strategy, in this work we propose a lithography-guided generative framework that can synthesize quasi-optimal mask with single round forwarding calculation. The quasi-optimal mask can be further refined by few steps of normal OPC engine. It should be noted conventional GAN cannot be directly applied here, due to the following two reasons. (1) Traditional DCGANs [27] are trained to mimic a dataset distribution which is not enough for the target-mask mapping procedure. (2) Compensation patterns or segment movements in the mask are derived based upon a large area of local patterns (e.g. $1000 \times 1000nm^2$) that brings much training pressure on the generator.

In accordance with these problems, we develop customized GAN training strategies for the purpose of mask optimization. Besides, since layout topology types are limited within specific area, we automatically synthesize local topology patterns based on size and spacing rules. The benefits of the artificial patterns are twofold: (1) we avoid to train the neural network with large images and facilitate the training procedure significantly; (2) automatically designed patterns are distributed uniformly and to some extent alleviate the over-fitting problem. Observing that most ILTs update the mask through steepest descent that resembles the training procedure in neural networks, we connect an ILT structure with the generative networks and pre-train the generator through back-propagating the lithography error to neuron weights. With the above pre-training phase, the generative model converges faster than training from random initialized neuron weights. Observe that GANs are typically much deeper than regular neural networks which bring inevitable training challenges. We further enhance the framework with an advanced generator design that integrates the U-Net [28] and the sub-pixel super-resolution (SPSR) structure [29] which are more computationally efficient, promise faster convergence and provide better mask image quality. The main contributions of this paper are listed as follows:

- We synthesize training patterns to enhance the computational efficiency and alleviate the over-fitting problem.
- We propose an ILT-guided pre-training flow to initialize the generator which can effectively facilitate the training procedure.
- We design new objectives of the discriminator to make sure the model is trained toward a target-mask mapping instead of a distribution.
- We enhance the GAN-OPC flow by integrating a U-Net and a sub-pixel super-resultion structure into the generator that promise better model convergence and generated mask quality.
- Experimental results show that our framework can significantly facilitate the mask optimization procedure as well as generating mask that has better printability under nominal condition.

The rest of the paper is organized as follows. Section II lists basic concepts and problem formulation. Section III discusses the details of GAN-OPC framework, ILT-guided training strategies, and enhanced GAN-OPC framework with U-Net and SPSR techniques. Section IV presents experimental results, followed by conclusion in Section V.

TABLE I Symbols and notations used throughout the paper.

Symbols	Description
$\boldsymbol{Z}_t$	Matrix representing the target layout pattern
$\boldsymbol{M}$	Matrix representing the mask pattern
$\boldsymbol{I}$	Matrix representing the aerial image
I_{th}	Threshold used for constant threshold resist model
$\boldsymbol{Z}$	Matrix representing the wafer image
$\boldsymbol{G}(\cdot)$	The generator function
$\boldsymbol{D}(\cdot)$	The discriminator function
$\boldsymbol{h}_k$	The k^{th} convolution kernel of the simple lithography model
w_k	The coefficient associated with $\boldsymbol{h}_k$
$\|\|\cdot\|\|_2$	L_2 norm, calculated with respect to flattened vectors
$\mathbb{E}$	Expectation
$\boldsymbol{x} \sim p$	Random vector $\boldsymbol{x}$ with its elements drawn from distribution p
$\otimes$	Matrix convolution
$\odot$	Entrywise product

II. PRELIMINARIES

In this section, we will discuss some preliminaries of mask optimization and the generative adversarial nets. Major math symbols with their descriptions are listed in TABLE I. In order to avoid confusion, all the norms $||\cdot||$ are calculated with respect to flattened vectors.

Hopkins theory of the partial coherence imaging system has been widely applied to mathematically analyze the mask behavior of lithography [30]. Because the Hopkins diffraction model is complex and not computational-friendly, [31] adopts the singular value decomposition (SVD) to approximate the original model with a weighted summation of coherent systems.

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

where $\boldsymbol{h}_k$ and w_k are the k^{th} kernel and its weight. As suggested in [13], we pick the N_h^{th} order approximation to the system. Equation (1) becomes,

$$\boldsymbol{I} = \sum_{k=1}^{N_h} w_k |\boldsymbol{M} \otimes \boldsymbol{h}_k|^2. \tag{2}$$

The lithography intensity corresponds to the exposure level on the photo resist that controls the final wafer image. In real design and sign-off flow, it is far from enough to use constant threshold model to analyze resist images, especially for advanced technology node. For the methodology verification purpose only and for simplicity, we still adopt the simple constant threshold resist model throughout the experiments, which is consistent with the ICCAD 2013 CAD contest settings [32]. In the constant threshold resist model [33], only area with intensity greater than a certain threshold will contribute to the final wafer image, as shown in Equation (3).

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

Mask quality is evaluated through the fidelity of its wafer image with respect to the target image. Edge placement

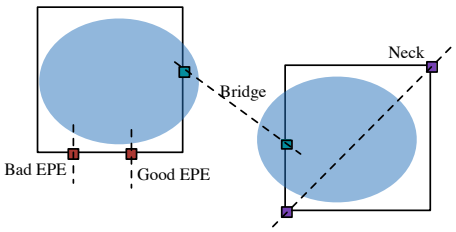


Fig. 2 Different types of defects. Same lithography images result in different EPE violation counts due to different choices of measurement points. Some defects are not detectable through merely checking edge placement errors.

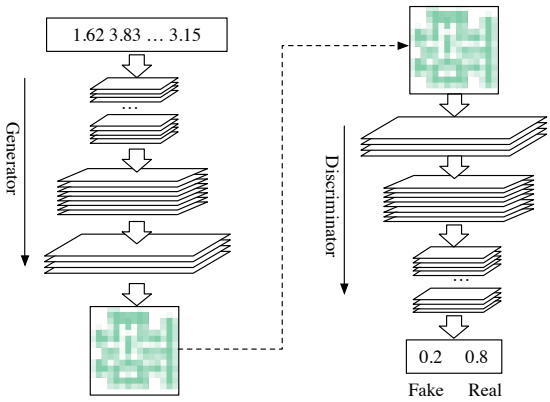


Fig. 3 Conventional GAN architecture.

error (EPE), bridge and neck are three main types of defect detectors that are adopted in a layout printability estimation flow. As shown in Fig. 2, EPE measures horizontal or vertical distances from given points (i.e. EPE measurement sites) on target edges to lithography contours. Neck detector checks the error of critical dimensions of lithography contours compared to target patterns, while bridge detector aims to find unexpected short of wires. Note that unlike EPE violations, bridge and neck defects can appear in any directions. Because EPE violations could happen with good critical dimension and neck or bridge occurs with small EPE, none of these defect types individually can be an ideal representation of mask printability. Considering the objective of mask optimization is to make sure the remaining patterns after lithography process are as close as target patterns, we pick the squared L_2 error as the metric of lithography quality since a smaller L_2 indicates a better wafer image quality.

Definition 1 (Squared L_2 Error)**.** *Let $\boldsymbol{Z}_t$ and $\boldsymbol{Z}$ as target image and wafer image respectively, the squared L_2 error of $\boldsymbol{Z}$ is given by $||\boldsymbol{Z}_t - \boldsymbol{Z}||_2^2$.*

In real manufacturing scenario, lithography conditions (e.g. focus, dose) are usually not fixed as we expected, which results in variations of wafer images. To measure the robustness of the designed mask, process variation (PV) bands are proposed [34]. The mathematical definition can be found as follows.

Definition 2 (PV Bands)**.** *Given the lithography simulation contours under a set of process conditions, the PV Bands is the area among all the contours under these conditions.*

In this work, for simplicity, we use the XOR between the innermost and the outermost images as an approximation of the PV Bands. Following above terminologies, we define the mask optimization problem as follows.

Problem 1 (Mask Optimization)**.** *Given a target image $\boldsymbol{Z}_t$, the objective of the problem is generating the corresponding mask $\boldsymbol{M}$ such that remaining patterns $\boldsymbol{Z}$ after lithography process is as close as $\boldsymbol{Z}_t$ or, in other word, minimizing the squared L_2 error of lithography images.*

III. The Frameworks

A. GAN-OPC

A classical GAN architecture comprises a generator and a discriminator. The generator accepts random vectors $\boldsymbol{z} \sim p_z$ as the input and generates samples $\boldsymbol{G}(\boldsymbol{z}; \boldsymbol{W}_g)$ that follows some distribution p_g, where $\boldsymbol{G}$ is a convolutional neural networks parameterized by $\boldsymbol{W}_g$. The discriminator acts as a classifier that distinguishes $\boldsymbol{G}(\boldsymbol{z}; \boldsymbol{W}_g)$ and the instance drawn from a data distribution p_d. The output $\boldsymbol{D}(\boldsymbol{x}; \boldsymbol{W}_d)$ represents the probabilities of $\boldsymbol{x}$ drawn from p_d and p_g. It should be noted that the original settings are not well suitable for the mask optimization problem. In this section, we will introduce the details of our framework including OPC-oriented GAN architecture and advanced training strategies.

1) Generator Design: From the previous discussion we can notice that the generator learns a distribution of a given dataset, which is originally designed as a mapping function $\boldsymbol{G} : p_z \to p_g$, where p_z is a distribution that input vectors are drawn and p_g denotes the distribution of the training set. The objective of the generator is to generate samples that deceive the discriminator as much as possible, as in Equation (4):

$$\max \mathbb{E}_{\boldsymbol{z} \sim p_z}[\log(\boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z})))], \tag{4}$$

which maximizes the log-likelihood of the discriminator giving predictions that generated samples are real. Correspondingly, the generator comprises a deconvolutional architecture that casts 1D vectors back to 2D images through stacked deconvolution operations, as shown in Fig. 3.

Our framework, however, is expected to perform mask optimization on given target circuit patterns and obviously violates the deconvolutional architecture. To resolve this problem, we design a generator based on auto-encoder [35] which consists of an encoder and a decoder subnets. As depicted in Fig. 4, the encoder is a stacked convolutional architecture that performs hierarchical layout feature abstractions and the decoder operates in an opposite way that predicts the pixel-based mask correction with respect to the target based on key features obtained from the encoder.

2) Discriminator Design: The discriminator is usually an ordinary convolutional neural networks that perform classi-

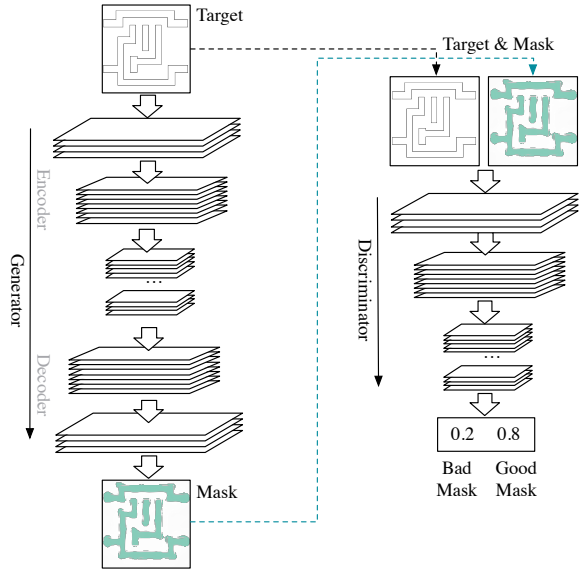


Fig. 4 The proposed GAN-OPC architecture.

fication to distinguish the generated samples from the given data samples as shown in Equation (5):

$$\max \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log(\boldsymbol{D}(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1-\boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z})))]. \quad (5)$$

In this work, the discriminator predicts whether an input instance is the generated mask $\boldsymbol{M}$ or the reference mask $\boldsymbol{M}^*$ which is the ground truth OPC'ed mask generated by a state-of-the-art academic OPC tool [13]. However, the discriminator in Equation (5) is necessary but not sufficient to ensure generator to obtain a high quality mask (Fig. 3). Consider a set of target patterns $\mathcal{Z} = \{\boldsymbol{Z}_{t,i}, i = 1, 2, \ldots, N\}$ and a corresponding reference mask set $\mathcal{M} = \{\boldsymbol{M}_i^*, i = 1, 2, \ldots, N\}$. Without loss of generality, we use $\boldsymbol{Z}_{t,1}$ in the following analysis. Suppose the above GAN structure has enough capacity to be well trained, the generator outputs a mask $\boldsymbol{G}(\boldsymbol{Z}_{t,1})$ that optimizes the objective function as in Equation (4). Observe that $\log(\boldsymbol{D}(\boldsymbol{G}(\boldsymbol{Z}_{t,1})))$ reaches its maximum value as long as

$$\boldsymbol{G}(\boldsymbol{Z}_{t,1}) = \boldsymbol{M}_i^*, \forall i = 1, 2, \ldots, N. \quad (6)$$

Therefore, a one-to-one mapping between the target and the reference mask cannot be guaranteed with current objectives. To address above concerns, we adopt a classification scheme that predicts positive or negative labels on target-mask pairs that inputs of the discriminator will be either $(\boldsymbol{Z}_t, \boldsymbol{G}(\boldsymbol{Z}_t))$ or $(\boldsymbol{Z}_t, \boldsymbol{M}^*)$, as illustrated in Fig. 4. Claim that $\boldsymbol{G}(\boldsymbol{Z}_t) \approx \boldsymbol{M}^*$ at convergence with new discriminator. We still assume enough model capacity and training time for convergence. The discriminator now performs prediction on target-mask pairs instead of masks. Because only pairs $\{\boldsymbol{Z}_{t,i}, \boldsymbol{M}_i^*\}$ are labeled as data, the generator can deceive the discriminator if and only if $\boldsymbol{G}(\boldsymbol{Z}_{t,i}) \approx \boldsymbol{M}_i^*, \forall i = 1, 2, \ldots, N$, where N is the total number of training instances.

3) GAN-OPC Training: Based on the OPC-oriented GAN architecture in our framework, we tweak the objectives of $\boldsymbol{G}$ as follows,

$$\min_{\boldsymbol{G}} \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}[\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{G}(\boldsymbol{Z}_t)))], \quad (7)$$

and for the discriminator $\boldsymbol{D}$ we have,

$$\begin{aligned}\max_{\boldsymbol{D}} \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}[\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{M}^*))] \\ + \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}[1-\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{G}(\boldsymbol{Z}_t)))].\end{aligned} \quad (8)$$

In addition to facilitate the training procedure, we minimize the differences between generated masks and reference masks when updating the generator as in Equation (9).

$$\min_{\boldsymbol{G}} \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}||\boldsymbol{M}^* - \boldsymbol{G}(\boldsymbol{Z}_t)||_n, \quad (9)$$

where $||\cdot||_n$ denotes the l_n norm. Combining (7), (8) and (9), the objective of our GAN model becomes

$$\begin{aligned}\min_{\boldsymbol{G}} \max_{\boldsymbol{D}} \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}[1-\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{G}(\boldsymbol{Z}_t))) + ||\boldsymbol{M}^* - \boldsymbol{G}(\boldsymbol{Z}_t)||_n^n] \\ + \mathbb{E}_{\boldsymbol{Z}_t\sim\mathcal{Z}}[\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{M}^*))].\end{aligned} \quad (10)$$

Previous analysis shows that the generator and the discriminator have different objectives, therefore the two sub-networks are trained alternatively, as shown in Fig. 5(a) and Algorithm 1. In each training iteration, we sample a mini-batch of target images (line 2); Gradients of both the generator and the discriminator are initialized to zero (line 3); A feed forward calculation is performed on each sampled instances (lines 4–5); The groundtruth mask of each sampled target image is obtained from OPC tools (line 6); We calculate the loss of the generator and the discriminator on each instance in the mini-batch (lines 7–8); We obtain the accumulated gradient of losses with respect to neuron parameters (lines 9–10); Finally the generator and the discriminator are updated by descending their mini-batch gradients (lines 11–12). Note that in Algorithm 1 we convert the min-max problem in Equation (10) into two minimization problems such that gradient ascending operations are no longer required to update neuron weights.

Algorithm 1 GAN-OPC Training

1: **for** number of training iterations **do**
2: Sample m target clips $\mathcal{Z} \leftarrow \{\boldsymbol{Z}_{t,1}, \boldsymbol{Z}_{t,2}, \ldots, \boldsymbol{Z}_{t,m}\}$;
3: $\Delta\boldsymbol{W}_g \leftarrow \boldsymbol{0}, \Delta\boldsymbol{W}_d \leftarrow \boldsymbol{0}$;
4: **for** each $\boldsymbol{Z}_t \in \mathcal{Z}$ **do**
5: $\boldsymbol{M} \leftarrow \boldsymbol{G}(\boldsymbol{Z}_t; \boldsymbol{W}_g)$;
6: $\boldsymbol{M}^* \leftarrow$ Groundtruth mask of $\boldsymbol{Z}_t$;
7: $l_g \leftarrow -\log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{M})) + \alpha||\boldsymbol{M}^* - \boldsymbol{M}||_2^2$;
8: $l_d \leftarrow \log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{M})) - \log(\boldsymbol{D}(\boldsymbol{Z}_t, \boldsymbol{M}^*))$;
9: $\Delta\boldsymbol{W}_g \leftarrow \Delta\boldsymbol{W}_g + \frac{\partial l_g}{\partial \boldsymbol{W}_g}$; $\Delta\boldsymbol{W}_d \leftarrow \Delta\boldsymbol{W}_d + \frac{\partial l_d}{\partial \boldsymbol{W}_g}$;
10: **end for**
11: $\boldsymbol{W}_g \leftarrow \boldsymbol{W}_g - \frac{\lambda}{m}\Delta\boldsymbol{W}_g$; $\boldsymbol{W}_d \leftarrow \boldsymbol{W}_d - \frac{\lambda}{m}\Delta\boldsymbol{W}_d$;
12: **end for**

Algorithm 1 differs from traditional GAN optimization flow on the following aspects. (1) The generator plays as a mapping function from target to mask instead of merely a distribution, therefore the gradient of L_2 loss is back-propagated along with the information from the discriminator. (2) The discriminator functions as an alternative of ILT engine that determines only the quality of generated masks without any calibration operations. Besides, our combined input ensures

that the discriminator will make positive prediction if and only if the generated mask is much close to the ground truth, which also helps train the generator better.

B. ILT-guided Pre-training

Although with OPC-oriented techniques, GAN is able to obtain a fairly good performance and training behavior, it is still a great challenge to train the complicated GAN model with satisfactory convergence. Observing that ILT and neural network training stage share similar gradient descent techniques, we develop an ILT-guided pre-training method to initialize the generator, after which the alternative mini-batch gradient descent is discussed as a training strategy of GAN optimization. The main objective in ILT is minimizing the lithography error through gradient descent.

$$E = ||\boldsymbol{Z}_t - \boldsymbol{Z}||_2^2, \tag{11}$$

where $\boldsymbol{Z}_t$ is the target and $\boldsymbol{Z}$ is the wafer image of a given mask. Because mask and wafer images are regarded as continuously valued matrices in the ILT-based optimization flow, we apply translated sigmoid functions to make the pixel values close to either 0 or 1.

$$\boldsymbol{Z} = \frac{1}{1+\exp[-\alpha \times (\boldsymbol{I} - \boldsymbol{I}_{th})]}, \tag{12}$$

$$\boldsymbol{M}_b = \frac{1}{1+\exp(-\beta \times \boldsymbol{M})}, \tag{13}$$

where $\boldsymbol{I}_{th}$ is the threshold matrix in the constant resist model with all the entries being I_{th}, $\boldsymbol{M}_b$ is the incompletely binarized mask, while α and β control the steepness of relaxed images.

Combine Equations (1)–(3), Equations (11)–(13) and the analysis in [12], we can derive the gradient representation as follows,

$$\begin{aligned}\frac{\partial E}{\partial \boldsymbol{M}} =& 2\alpha\beta \times \boldsymbol{M}_b \odot (1-\boldsymbol{M}_b) \odot \\ & (((\boldsymbol{Z}-\boldsymbol{Z}_t) \odot \boldsymbol{Z} \odot (1-\boldsymbol{Z}) \odot (\boldsymbol{M}_b \otimes \boldsymbol{H}^*)) \otimes \boldsymbol{H} + \\ & ((\boldsymbol{Z}-\boldsymbol{Z}_t) \odot \boldsymbol{Z} \odot (1-\boldsymbol{Z}) \odot (\boldsymbol{M}_b \otimes \boldsymbol{H})) \otimes \boldsymbol{H}^*),\end{aligned} \tag{14}$$

where $\boldsymbol{H}^*$ is the conjugate matrix of the original lithography kernel $\boldsymbol{H}$. In traditional ILT flow, the mask can be optimized through iteratively descending the gradient until E is below a threshold.

The objective of mask optimization problem indicates the generator is the most critical component in GAN. Observing that both ILT and neural network optimization share similar gradient descent procedure, we propose a jointed training algorithm that takes advantages of ILT engine, as depicted in Fig. 5(b). We initialize the generator with lithography-guided pre-training to make it converge well in the GAN optimization flow thereafter. The key step of neural network training is back-propagating the training error from the output layer to the input layer while neural weights are updated as follows,

$$\boldsymbol{W}_g = \boldsymbol{W}_g - \frac{\lambda}{m}\Delta\boldsymbol{W}_g, \tag{15}$$

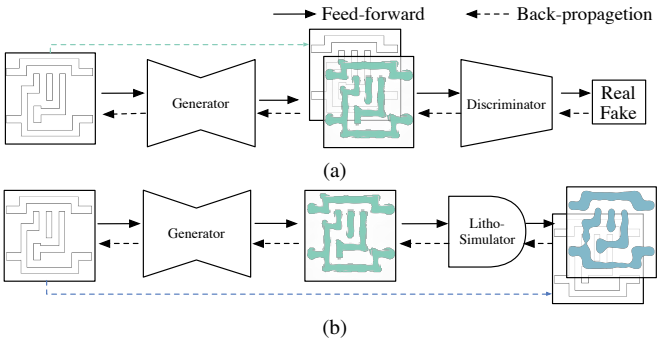


Fig. 5 (a) GAN-OPC training and (b) ILT-guided pre-training.

where $\Delta\boldsymbol{W}_g$ is accumulated gradient of a mini-batch of instances and m is the mini-batch instance count. Because Equation (15) is naturally compatible with ILT, if we create a link between the generator and ILT engine, the wafer image error can be back-propagated directly to the generator as presented in Fig. 5.

The generator pre-training phase is detailed in Algorithm 2. In each pre-training iteration, we sample a mini-batch of target layouts (line 2) and initialize the gradients of the generator $\Delta\boldsymbol{W}_g$ to zero (line 3); The mini-batch is fed into the generator to obtain generated masks (lines 5). Each generated mask is loaded into the lithography engine to obtain a wafer image (line 6); The quality of wafer image is estimated by Equation (11) (lines 7); We calculate the gradient of lithography error E with respect to the neural networks parameter $\boldsymbol{W}_g$ through the chain rule, i.e., $\frac{\partial E}{\partial \boldsymbol{M}}\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{W}_g}$ (line 8) ; Finally, $\boldsymbol{W}_g$ is updated following the gradient descent procedure (line 10).

Algorithm 2 ILT-guided Pre-training

1: **for** number of pre-training iterations **do**
2: Sample m target clips $\mathcal{Z} \leftarrow \{\boldsymbol{Z}_{t,1}, \boldsymbol{Z}_{t,2}, \ldots, \boldsymbol{Z}_{t,m}\}$;
3: $\Delta\boldsymbol{W}_g \leftarrow 0$;
4: **for** each $\boldsymbol{Z}_t \in \mathcal{Z}$ **do**
5: $\boldsymbol{M} \leftarrow \boldsymbol{G}(\boldsymbol{Z}_t; \boldsymbol{W}_g)$;
6: $\boldsymbol{Z} \leftarrow \texttt{LithoSim}(\boldsymbol{M})$ ▷ Equations (2)–(3)
7: $E \leftarrow ||\boldsymbol{Z} - \boldsymbol{Z}_t||_2^2$;
8: $\Delta\boldsymbol{W}_g \leftarrow \Delta\boldsymbol{W}_g + \frac{\partial E}{\partial \boldsymbol{M}}\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{W}_g}$; ▷ Equation (14)
9: **end for**
10: $\boldsymbol{W}_g \leftarrow \boldsymbol{W}_g - \frac{\lambda}{m}\Delta\boldsymbol{W}_g$; ▷ Equation (15)
11: **end for**

Compared to the training towards ground truth (i.e., directly back-propagate the mask error to neuron weights), ILT-guided pre-training provides step-by-step guidance when searching for a solution with high quality, which reduces the possibility of the generator being stuck at local minimum region in an early training stage. Because ILT contains complicated convolutions and matrix multiplications that are computational expensive, we approximate the pre-training stage through back-propagating errors of intermediate masks, which "guides" the generator towards optimality. We only adopt the ILT engine in the pre-training stages and replace it with the discriminator

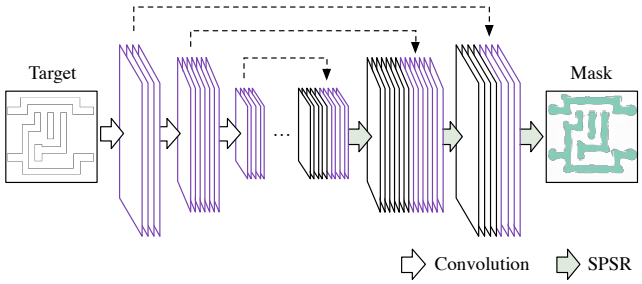


Fig. 6 New generator architecture with concatenation of intermediate feature maps and an SPSR structure.

in the main training stage where the generator is optimized in an adversarial style.

C. Enhanced GAN-OPC Framework

In this section, we will introduce the enhanced GAN-OPC framework, which significantly improves the training efficiency in a more elegant way compared to pretraining with ILT engine. The enhanced GAN-OPC framework includes a U-Net structure that allows gradients to be easily back-propagated to early layers and a sub-pixel super-resolution architecture for better generated mask quality.

1) U-Net: We have noticed the GANs are typically deeper than traditional neural networks, which brings more challenges due to a longer gradient back-propagation path. A common solution is creating shortcut links that allow addition or stacking of feature maps in different layers [28], [36], [37], such that gradients can be more efficiently back-propagated from output layer to early layers. Here we enhance our generator design with a U-Net-like structure where intermediate feature maps in the encoder are stacked at corresponding layers in the decoder, as shown in Fig. 6. Such architecture has two good properties: (1) The inevitable information loss in strided convolution layer can be drastically reduced. (2) The gradient vanishing problem can be alleviated with multiple shortcut links bypassing intermediate feature maps.

2) Sub-Pixel Super-Resolution: In previous designs, low level features in intermediate generator layers are cast back to mask images by standard strided deconvolution operation that can be visualized as in Fig. 7(a). In detail, zeros are inserted among existing pixels such that the output dimension after a convolution operation reaches the desired value. However, such mechanism requires multiple convolution operations on high resolution space which is not computational efficient and might induce additional noises.

SPSR [29] is another upsampling solution that has been widely used in super-resolution tasks. It conducts convolution operations in lower resolution space and generates additional feature maps such that the number of feature map entries matches the desired size of target image, as shown in Fig. 7(b). The major step of SPSR is called periodic shuffling that casts a tensor with shape $H \times W \times r^2C$ into shape $rH \times rW \times C$ as defined in Equation (16).

$$t^{hr}_{i,j,k} = t^{lr}_{i',j',k'}, \tag{16a}$$

$$i' = \lfloor \frac{i}{r} \rfloor, \tag{16b}$$

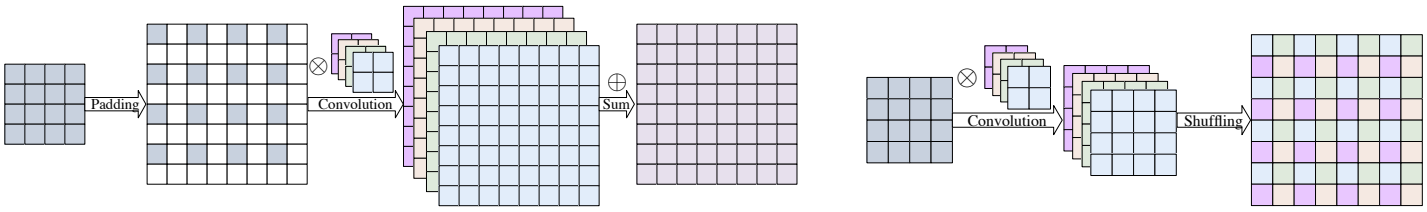
$$j' = \lfloor \frac{j}{r} \rfloor, \tag{16c}$$

$$k' = C \cdot r \cdot \mathrm{mod}(j, r) + C \cdot \mathrm{mod}(i, r) + k, \tag{16d}$$

where $\lfloor \cdot \rfloor$ is the math floor operator, $\mathrm{mod}(x, y)$ finds the remainder of x divided by y, $t^{hr}_{i,j,k}$ and $t^{lr}_{i',j',k'}$ denotes the (i, j, k) and (i', j', k') entry of high resolution images (or feature maps) and low resolution images (or feature maps) respectively. It should be noted that Equation (16) represents only a reshape operation which is still differentiable as other convolution layers. SPSR has several advantages compared to Fig. 7(a). (1) SPSR is ideally r^2 times faster than the strided deconvolution operation. As shown in Fig. 7, same convolution kernels have to scan over a r^2 larger feature maps in traditional deconvolution layers to achieve same output tensor size as SPSR layers. (2) SPSR layers reduce noises in generated masks by a significant amount, as can be seen in Fig. 8. Such results can be explained by the fact that explicit interpolations are removed in SPSR structure, where the upscaling and rendering are automatically learned during the network training. Traditional deconvolution layers, on the other hand, have to apply padding or zero insertion to increase the feature map size before feeding them into next level convolution layer for rendering, which in turn results in noises (as empty dots) in the generated masks because it is hard for limited number of convolution layers to smooth such noise. On the contrary, SPSR directly organizes the low resolution feature maps into the high resolution space, where every pixels are informative, compared to manually inserted zeros in deconvolution layers.

3) The Enhanced GAN-OPC Architecture: We follow the basic convolutional auto-encoder architecture for the generator design with additional shortcut links for U-Net feature map sharing and SPSR layers for upscaling. The detailed architecture can be found in TABLE II, where column "Layer" includes layer types and layer ID, columns "Filter" and "Stride" list configurations for convolution layers, "Output" lists the output tensor shape of corresponding layers and "Parameter" represents the total number of trainable parameters of a given layer. The proposed generator architecture contains five regular convolution layers for feature extraction and five SPSR layers for mask image construction. It should be noted that the input tensor of the i^{th} SPSR layer has $2\times$ channel numbers as the output tensor of the $(i-1)^{\text{th}}$ SPSR layer, because of the existence of U-Net concatenation.

The discriminator design is detailed in TABLE III. The neural network architecture resembles VGG [38] with more layers and smaller kernels. "repeat2" and "repeat3" indicate two and three consecutive convolution layers with the same configurations. We replace all the pooling layers with strided convolution layers to attain information as much as possible. Three densely connected layers are connected following the last convolution layer for final class prediction. The total



(a) Deconvolution (b) SPSR

Fig. 7 Visualization of (a) standard deconvolution operation and (b) SPSR.

(a) (b)

Fig. 8 Patterns generated from (a) deconvolution layers and (b) SPSR layers.

TABLE II The generator configuration.

Layer	Filter	Stride	Output	Parameter
conv-1	5×5×16	2	128×128×16	400
conv-2	5×5×64	2	64×64×64	25600
conv-3	5×5×128	2	32×32×128	204800
conv-4	5×5×512	2	16×16×512	1638400
conv-5	5×5×1024	2	8×8×1024	13107200
spsr-5	3×3×2048	1	16×16×512	18874368
spsr-4	3×3×512	1	32×32×128	4718952
spsr-3	3×3×256	1	64×64×64	589824
spsr-2	3×3×64	1	128×128×16	73728
spsr-1	3×3×4	1	256×256×1	1152
Summary	-	-	-	39234064

TABLE III The discriminator configuration.

Layer	Filter	Stride	Output	Parameter
repeat2-1	3×3×64	1	256×256×64	38016
conv-1	3×3×64	2	128×128×64	36864
repeat2-2	3×3×128	1	128×128×128	221184
conv-2	3×3×128	2	64×64×128	147456
repeat3-1	3×3×256	1	64×64×256	1474560
conv-3	3×3×256	2	32×32×256	589824
repeat3-2	3×3×512	1	32×32×512	5898240
conv-4	3×3×512	2	16×16×512	2359296
repeat3-3	3×3×512	1	16×16×512	7077888
conv-5	3×3×512	2	8×8×512	2359296
fc-1	-	-	2048	67108864
fc-2	-	-	512	1048576
fc-3	-	-	2	1024
Summary	-	-	-	88361088

number of trainable parameters of the discriminator are intentionally designed much larger than the generator (88M vs. 39M) in case of model collapsing.

D. Framework Summary and Discussion

The proposed GAN-OPC family is summarized in Fig. 9. With the given training data that includes target patterns and

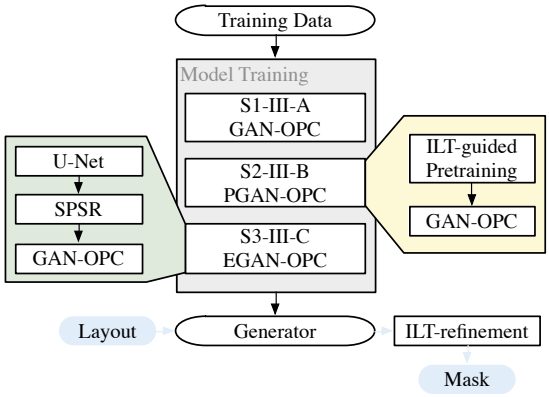


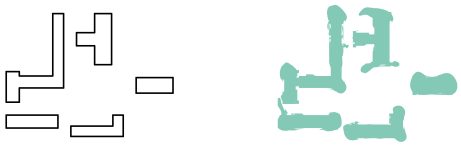
Fig. 9 The framework summary.

mask patterns, we propose three alternative solutions to obtain a trained generator that include direct GAN-OPC proposed in Section III-A, GAN-OPC with ILT-guided pretraining as in Section III-B and the enhanced GAN-OPC solution with U-Net and SPSR techniques as in Section III-C.

Although ILT engines are, to some extent, suffering mask manufacturability (e.g violation of mask notch rule and mask spacing rule [7], [11] which are not considered in this work) and runtime issues compared to traditional model-based OPC, our framework still takes advantage of such methodology with the following reasons. Our framework is built upon the structure of conditional GAN that learns a pixel-to-pixel mapping from the target pattern to the OPCed pattern. The optimization scheme is in a continuous form that compensation patterns can appear in any shapes and any places within the clip. Thus the patterns generated by GAN are inconsistent with the model-based OPC results (e.g. [6], [7]) where compensations are made by moving polygon segments inward or outward. However, we observe that the mask patterns are naturally compatible with the process of ILT, which becomes one reason that we choose ILT for our refinement tasks. As can be seen in previous works [13], [14], ILT is associated with a highly non-convex optimization problems that means the mask initialization affects the final results. The ILT refinement results outperform direct ILT optimization and also experimentally demonstrate the effectiveness of the proposed GAN-OPC framework. Another reason that we choose ILT is that theoretically and intuitively ILT provides a larger solution space in mask optimization problems and tends to offer better mask quality. There are two major advantages of our proposed framework:

TABLE IV The design rules used.

Item	Min Size (nm)
M1 Critical Dimension	80
Pitch	140
Tip to tip distance	60



(a) Target (b) Reference Mask

Fig. 10 An example of a target and a reference mask pair.

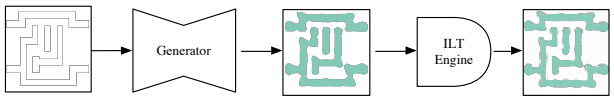


Fig. 11 GAN-OPC flow: generator inference and ILT refinement.

- Compared to ILT itself, the GAN-OPC family offers a better starting point for ILT optimization that promises faster convergence and better mask quality.
- Compared to model-based OPC, the proposed framework attains good properties of ILT, i.e., a lager solution space that has the potential to generate better pattern compensation for better mask printability.

Although we did not consider the mask notch and spacing rule in our framework, it is straightforward to conduct mask manufacturability rule check on the generated masks, and fix the violated region by making minor pixel changes in the generated masks. Actually, if the ground truth masks used for training can meet the mask manufacturability requirements, the GAN-OPC framework is supposed to capture these rules during training, because the discriminator is specifically designed to tell whether the generated masks are good or not. Here a "good" mask refers to the mask that has good printability and good manufacturability.

IV. Experimental Results

The generative adversarial network for mask optimization is implemented based on `Tensorflow` [39] library and tested on single Nvidia Titan X. The lithography engine is based on the `lithosim_v4` package from ICCAD 2013 CAD Contest [32], which also provides ten industrial M1 designs on $32nm$ design node. We pick $N_h = 24, \alpha = 50$ and $\beta = 4$ for the lithography simulation procedure. The ILT refinement will be stopped if the average gradient per pixel as calculated in Equation (14) is smaller than 5×10^{-4}. Related parameters are chosen according to the experimental results on one test case. The OPC framework applies $8nm$ resolution during the initial mask generation stage and $1nm$ resolution for refinement.

A. Synthesizing Training Data

As a type of deep neural networks, GAN can be hardly well trained with only ten instances. To verify our framework, we synthesize a training layout library with 4000 instances based on the design specifications from existing $32nm$ M1 layout topologies. 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 simple design rules, as detailed in TABLE IV. An example of such synthesized target-reference mask pair can be found in Fig. 10. In addition, most generative models have shown obvious weakness in image details, which makes it extremely hard to optimize images with size 2048×2048. Therefore, we perform 8×8 average pooling on layout images before feeding them into the neural networks. In the generation stage we adopt simple linear interpolation to convert the layout images back to their original resolution.

B. Evaluation of GAN-OPC and PGAN-OPC

The proposed GAN-OPC flow is illustrated in Fig. 11, where we first feed target patterns into the generator and obtain the quasi-optimal masks, followed by refinement through an ILT engine. In the first experiment, to verify the effectiveness of ILT-guided pre-training algorithm, we record training behaviors of two GANs which are denoted by GAN-OPC and PGAN-OPC. Here "GAN-OPC" and "PGAN-OPC" denote GAN-OPC flow without generator pre-training and GAN-OPC flow with ILT-guided pre-training, respectively. "ILT" corresponds to `MOSAIC_fast` in [13]. The training procedure is depicted in Fig. 12, where x-axis indicates training steps and y-axis is L_2 loss between generator outputs and ground truth masks, as in Equation (9).

The training time for both GAN and PGAN are around 10 hours on our platform. Although L_2 loss of GAN-OPC drops slightly faster before 3000 iterations, the training curve shows that PGAN-OPC is a more stable training procedure and converges to a lower loss. Besides, it takes much more efforts for GAN-OPC to search a direction to descending the gradient fast, while the training loss of PGAN-OPC drops smoothly and converges at a lower L_2 loss than GAN-OPC, which indicates ILT-guided pre-training indeed facilitates mask-optimization-oriented GAN training flow. We will also show that PGAN-OPC exhibits better mask optimization results in the following section.

In the second experiment, we optimize the ten layout masks in ICCAD 2013 contest benchmark [32] and compare the results with previous work, as listed in TABLE V. Here the wafer images are calculated from the simulation tool (`lithosim_v4`) in the contest [32]. Note that all the GAN-OPC and PGAN-OPC results are refined by an ILT engine which generates final masks to obtain wafer images. Column "L_2" is the squared L_2 error between the wafer image and the target image under nominal condition. Column "PVB" denotes the contour area variations under $\pm$ 2% dose error and defocus range of $\pm25nm$ settings as in the contest. It is notable that GAN-OPC significantly reduces squared L_2

TABLE V Comparison with state-of-the-art.

Benchmarks	ILT [13]		Model-Based [6]		GAN-OPC		PGAN-OPC		EGAN-OPC	
	L2	PVB (nm^2)	L2	PVB (nm^2)	L2	PVB (nm^2)	L2	PVB (nm^2)	L2	PVB (nm^2)
1	49893	65534	53816	66218	54970	64163	52570	56267	55425	58043
2	50369	48230	41382	53434	46445	56731	42253	50822	40211	53020
3	81007	108608	79255	146776	88899	84308	83663	94498	93090	75644
4	20044	28285	21717	33266	18290	29245	19965	28957	22877	26401
5	44656	58835	48858	65631	42835	59727	44733	59328	42650	59765
6	57375	48739	46320	62068	44313	52627	46062	52845	39776	54878
7	37221	43490	31898	51069	24481	47652	26438	47981	22761	49156
8	19782	22846	23312	25898	17399	23769	17690	23564	16296	24441
9	55399	66331	55684	75387	53637	66766	56125	65417	52157	66492
10	24381	18097	19722	18536	9677	20693	9990	19893	9765	21338
Average	44012.7	50899.5	42196.4	59828.3	40094.6	50568.1	39948.9	49957.2	39500.8	48917.8
Ratio	1.0	1.0	0.959	1.175	0.911	0.993	0.908	0.981	**0.898**	**0.961**

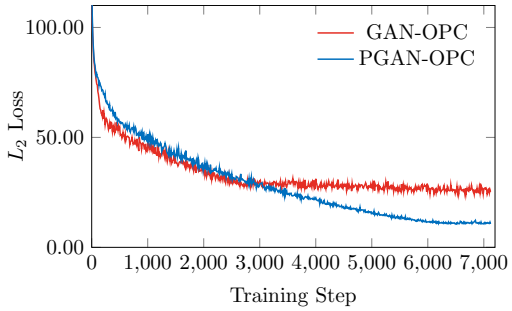


Fig. 12 Training curves of GAN-OPC and PGAN-OPC.

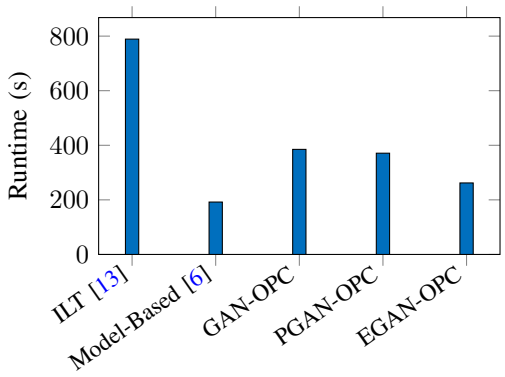


Fig. 13 Average runtime comparison of different methods.

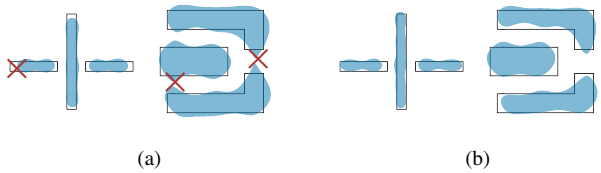


Fig. 14 Some wafer image details of (a) ILT [13] and (b) PGAN-OPC.

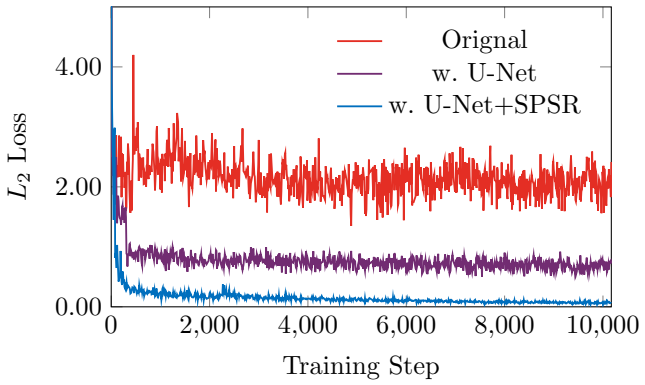


Fig. 15 Training behavior of the EGAN-OPC framework with faster and better convergence.

error of wafer images under the nominal condition by 9% and with the ILT-guided pre-training, squared L_2 error is slightly improved and PVB is further reduced by 1%. We also compared our work with one academic state-of-the-art Model-based OPC engine [6], which exhibits larger L_2 error (42196.4) and worse PVB (59828.3nm^2) compared to GAN-OPCs.

Because we only focus on the optimization flow under the nominal condition and no PVB factors are considered, our method only achieves comparable PVB areas among ten test cases. Additionally, feed-forward computation of GAN only takes $0.2s$ for each image which is ignorable, therefore runtime of our flow is almost determined by ILT refinements. Runtime of different frameworks are illustrated in Fig. 13. Items "ILT", "Model-Based""GAN-OPC", "PGAN-OPC" list the average mask optimization time of [13], [6], GAN-OPC and PGAN-OPC, respectively. For most benchmark cases, GAN-OPC and PGAN-OPC show a earlier stop at a smaller L_2 error and, on average, reduce the optimization runtime by more than 50%. We also observe that model-based OPC engine shows advantages on execution time at the cost of wafer image quality as well as PVB area, as shown in TABLE V. For most test cases, [13] exhibits a smaller PV band area possibly because the printed images are more likely to have large wafer image CD and shorter wire length, which makes masks suffer less proximity effects while inducing bridge or line-end pull back defects, as shown in Fig. 14.

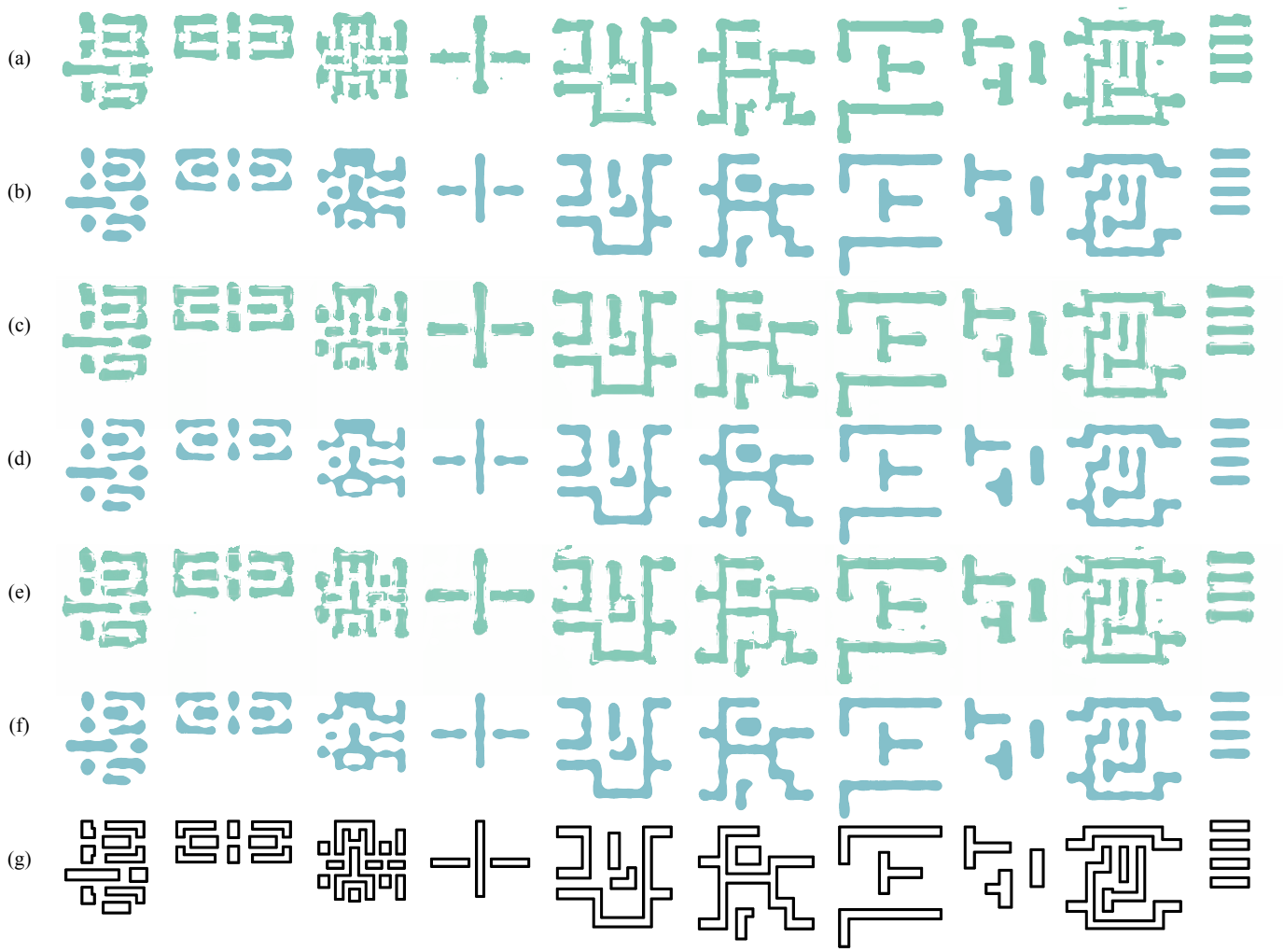


Fig. 16 Result visualization of PGAN-OPC, Enhanced GAN-OPC and ILT. Columns correspond to ten test cases from ICCAD 2013 CAD contest. Rows from top to bottom are: (a) masks of [13]; (b) wafer images by masks of [13]; (c) masks of PGAN-OPC; (d) wafer images by masks of PGAN-OPC; (e) masks of Enhanced GAN-OPC (f) wafer images by masks of Enhanced GAN-OPC; (g) target patterns.

C. Evaluation of Enhanced GAN-OPC

Here we show the effectiveness and the efficiency of the Enhanced GAN-OPC (EGAN-OPC) framework. In the first experiment, we illustrate the training behavior of PGAN-OPC and the EGAN-OPC frameworks as shown in Fig. 15. Red curve stands for the original PGAN-OPC model, which fluctuates fiercely around a large value. Dark curve refers to the results with U-Net generator. Blue curve represents the complete version of enhanced GAN model with both U-net structure and the embedded SPSR structure. It's encouraging to see that U-net alone can already ensure a good convergence in terms of L_2 loss. As we have pointed out in algorithm section, such structure attains the neural network capacity with significantly lower computational cost, which is consistent with the trends of L_2 error during training.

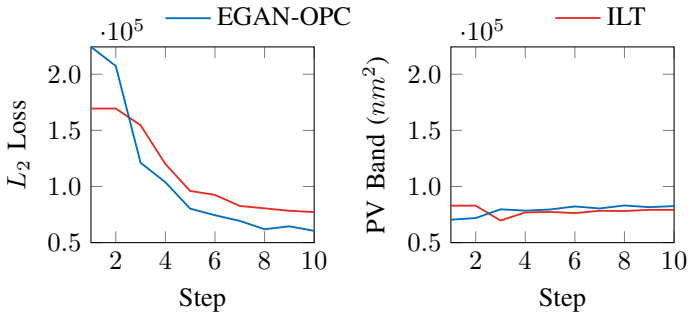
In the second experiment, we compare the mask optimization results of the EGAN-OPC with original GAN-OPC and PGAN-OPC, as depicted in Fig. 16. The quantitative results can also be found in column "EGAN-OPC" of TABLE V. EGAN-OPC outperforms PGAN-OPC and GAN-OPC on most test cases with better L_2 error (39500 vs. 39948) and smaller PVB area ($48917nm^2$ vs. $49957nm^2$) with only 70% average runtime of PGAN-OPC (see Fig. 13), which demonstrates the efficiency of EGAN-OPC framework. It should be also noted that EGAN-OPC can be trained end-to-end without any interaction with the lithography engine which induces a large amount of computational cost in PGAN-OPC.

D. On the Scalability of GAN-OPC Family

In order to verify the scalability of our frameworks, we conduct further experiments on 10 additional testcases that contain more patterns and larger total pattern areas. Similar to [7], these 10 testcases are created from the original IBM benchmarks with additional geometries. The results of one example can be found in Fig. 17. It can be seen that our framework generalizes to more complex patterns. We also visualize the ILT convergence in terms of different mask initialization in Fig. 18. Here we use testcase 18 as an

(a) (b) (c)

Fig. 17 A larger-case example of (a) a mask pattern, (b) its wafer image and (c) the corresponding target pattern.



(a) L2 (b) PV Band

Fig. 18 Visualization of convergence during ILT refinement.

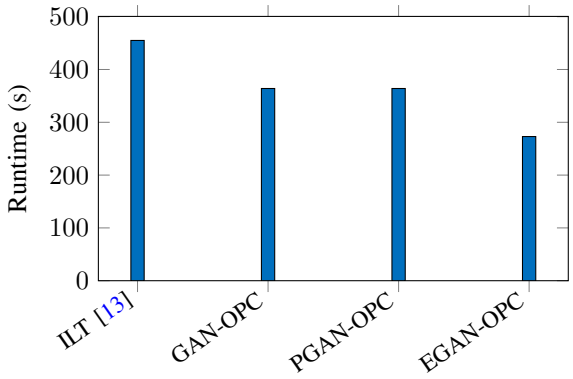


Fig. 19 Average runtime comparison on larger benchmarks.

example. It can be seen that ILT converges much faster when using the mask initialized by EGAN-OPC as input, with only ignorable PV Band penalty. We did not compare the performance with model-based OPC as the binary release in [6] encounters unknown failure on the new benchmarks.

We list the detailed optimization results in TABLE VI, where columns are defined exactly the same as TABLE V. It can be seen that GAN-OPC exhibits trade-offs on nominal image quality and PVB compared to pure ILT, while both PGAN-OPC and EGAN-OPC show significant advantages on L_2 error (86105.7 v.s. 90486.3) with similar or slightly better PVB (108690.7nm^2 v.s.109842.7nm^2). Besides, competitive results of our framework are also achieved with shorter optimization time thanks to the good initialization offered by the generator, as shown in Fig. 19.

V. Conclusion

In this paper, we have proposed a GAN-based mask optimization flow that takes target circuit patterns as input and generates quasi-optimal masks for further ILT refinement. We analyze the specialty of mask optimization problem and design OPC-oriented training objectives of GAN. Inspired by the observation that ILT procedure resembles gradient descent in back-propagation, we develop an ILT-guided pre-training algorithm that initializes the generator with intermediate ILT results, which significantly facilitates the training procedure. We also enhance the GAN-OPC flow by integrating U-Net and SPSR layers in the generator that ensures better model convergence and mask quality. Experimental results show that our framework not only accelerates ILT but also has the potential to generate better masks through offering better starting points in ILT flow.

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] B. Yu, X. Xu, S. Roy, Y. Lin, J. Ou, and D. Z. Pan, "Design for manufacturability and reliability in extreme-scaling VLSI," *Science China Information Sciences*, pp. 1–23, 2016.

[3] "ITRS," http://www.itrs.net.

[4] X. Xu, T. Matsunawa, S. Nojima, C. Kodama, T. Kotani, and D. Z. Pan, "A machine learning based framework for sub-resolution assist feature generation," in *ACM International Symposium on Physical Design (ISPD)*, 2016, pp. 161–168.

[5] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process variation and pattern fidelity aware mask optimization algorithm," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 238–245.

[6] 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.

[7] 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.

[8] P. Yu, S. X. Shi, and D. Z. Pan, "Process variation aware OPC with variational lithography modeling," in *ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 785–790.

[9] 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 μm ASIC," in *IEEE International Symposium on Quality Electronic Design (ISQED)*, 2000, pp. 81–85.

[10] P. Yu, S. X. Shi, and D. Z. Pan, "True process variation aware optical proximity correction with variational lithography modeling and model calibration," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 6, no. 3, pp. 031 004–031 004, 2007.

[11] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process-variation-aware mask optimization algorithm with a novel intensity modeling," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 25, no. 3, pp. 998–1011, 2017.

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

[13] 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.

[14] 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.

[15] W. Xiong, J. Zhang, Y. Wang, Z. Yu, and M.-C. Tsai, "A gradient-based inverse lithography technology for double-dipole lithography," in *International Conference on Simulation of Semiconductor Processes and Devices*. IEEE, 2009, pp. 1–4.

[16] R. Viswanathan, J. T. Azpiroz, and P. Selvam, "Process optimization through model based SRAF printing prediction," in *SPIE Advanced Lithography*, vol. 8326, 2012.

TABLE VI Experiments on larger benchmarks.

Benchmarks	ILT [13]		GAN-OPC		PGAN-OPC		EGAN-OPC	
	L2	PVB (nm^2)	L2	PVB (nm^2)	L2	PVB (nm^2)	L2	PVB (nm^2)
11	94792	125578	90547	137107	89229	121276	93704	119928
12	94604	128306	97969	132929	93454	127551	96634	122060
13	136861	160327	137981	175390	134397	155511	131220	153527
14	69090	79337	62582	94562	58776	85644	57329	85560
15	96961	116039	102759	126157	99830	116728	99926	113971
16	98159	115107	98070	123707	96335	113981	93755	111186
17	79192	91989	76807	98744	71522	94841	70864	94877
18	65572	81503	63573	93219	60372	83718	58383	83568
19	107095	121922	103753	136493	105973	122770	102994	122371
20	62537	78319	61524	90514	57086	81285	56248	79859
Average	90486.3	109842.7	89556.5	120882.2	86697.4	110330.5	86105.7	108690.7
Ratio	1.00	1.00	0.990	1.101	0.958	1.004	**0.952**	**0.990**

[17] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in *Proceedings of SPIE*, vol. 9427, 2015.

[18] H. Zhang, B. Yu, and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 47:1–47:8.

[19] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: a deep learning approach," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 16, no. 3, p. 033504, 2017.

[20] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2017, pp. 62:1–62:6.

[21] 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.

[22] A. Gu and A. Zakhor, "Optical proximity correction with linear regression," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 21, no. 2, pp. 263–271, 2008.

[23] R. Luo, "Optical proximity correction using a multilayer perceptron neural network," *Journal of Optics*, vol. 15, no. 7, p. 075708, 2013.

[24] S. Choi, S. Shim, and Y. Shin, "Machine learning (ML)-guided OPC using basis functions of polar fourier transform," in *Proceedings of SPIE*, vol. 9780, 2016.

[25] 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.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 214–223.

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

[28] 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 (MICCAI)*, 2015, pp. 234–241.

[29] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874–1883.

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

[31] N. B. Cobb, "Fast optical and process proximity correction algorithms for integrated circuit manufacturing," Ph.D. dissertation, University of California at Berkeley, 1998.

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

[33] W.-C. Huang, C.-H. Lin, C.-C. Kuo, C. Huang, J. Lin, J.-H. Chen, R.-G. Liu, Y. C. Ku, and B.-J. Lin, "Two threshold resist models for optical proximity correction," in *Optical Microlithography XVII*, vol. 5377. International Society for Optics and Photonics, 2004, pp. 1536–1544.

[34] J. A. T. Robles, "Integrated circuit layout design methodology with process variation bands," Aug. 5 2014, uS Patent 8,799,830.

[35] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks (ICANN)*, 2011, pp. 52–59.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[39] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, "TensorFlow: A system for large-scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

Haoyu Yang received his B.E. degree from Qiushi Honors College, Tianjin University in 2015. He is currently pursuing his Ph.D. Degree at the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has interned at ASML, San Jose, CA and Cadence Design Systems, San Jose, CA. He received the 2019 Nick Cobb Scholarship by SPIE and Mentor Graphics. His research interests include machine learning and VLSI design and sign-off.

Shuhe Li received his B.Sc Degree from The Chinese University of Hong Kong in 2019. He is currently pursuing his M. Sc degree in Computer Science at The Chinese University of Hong Kong.

Zihao Deng received his B.Sc. degree in Computer Science with first class honour from The Chinese University of Hong Kong in 2019. He is currently pursuing his Ph.d. degree at Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include machine learning algorithms, deep neural networks, and information theory.

Yuzhe Ma received his B.E. degree from the Department of Microelectronics, Sun Yat-sen University in 2016. He is currently pursuing his Ph.D. degree at the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has interned at Cadence Design Systems, San Jose, CA, USA and NVIDIA Research, Austin, TX, USA. His research interests include VLSI design for manufacturing, physical design and machine learning on chips.

Bei Yu (S'11–M'14) received the Ph.D. degree from The University of Texas at Austin in 2014. He is currently an Assistant Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is Editor-of-Chief of IEEE TCCPS Newsletter. He received five Best Paper Awards from Integration, the VLSI Journal in 2018, International Symposium on Physical Design 2017, SPIE Advanced Lithography Conference 2016, International Conference on Computer Aided Design 2013, and Asia and South Pacific Design Automation Conference 2012, and five ICCAD/ISPD contest awards.

Evangeline F.Y. Young received her B.Sc. degree in Computer Science from The Chinese University of Hong Kong (CUHK). She received her Ph.D. degree from The University of Texas at Austin in 1999. She is currently a professor in the Department of Computer Science and Engineering in CUHK. Her research interests include Optimization, algorithms and VLSI CAD. She works actively on floorplanning, placement, routing, DFM and EDA on Physical Design in general. Dr. Young has served on the organization committees of ISPD, ARC and FPT and on the program committees of conferences including DAC, ICCAD, ISPD, ASP-DAC, SLIP, DATE and GLSVLSI. She also served on the editorial boards of IEEE TCAD, ACM TODAES and Integration, the VLSI Journal. Besides, her research group has won several championships and prizes in renown EDA contests, including the 2016, 2015, 2013 and 2012 CAD Contests at ICCAD, DAC 2012 and ISPD 2011 Routability-driven Placement Contests and ISPD 2010 High Performance Clock Network Synthesis Contest.