Synthesis of Lithography Test Patterns Using Machine Learning Model

Pervaiz Kareem and Youngsoo Shin, *Fellow, IEEE*

***Abstract*—Diversity of test patterns is important for many lithography applications. It is however difficult to achieve with sample layouts or by using a popular pattern generator. We propose a synthesis method of lithography test patterns. Each pattern is represented by a map of IPS (image parameter space) values, called IPS map. A key in the proposed method is to guarantee that the center of the synthesized pattern corresponds to the IPS values given as input. The synthesis consists of three steps: a new IPS map is generated using an adversarial auto-encoder (AAE) with given IPS values; the IPS map is converted to its corresponding layout through an auto-encoder (AE); the layout goes through the final processing to remove any unrealistic shapes. Both AAE and AE are trained beforehand by using a few sample layouts. The synthesis method is applied to lithography modeling. The RMSE of lithography model is reduced by 30% when the model is calibrated with synthesized patterns, compared to the model based on test patterns from a pattern generator. A machine learning-based lithography simulation is taken as a second application. When the synthesized patterns are used to train the machine learning model, the accuracy of lithography simulation improves by 7%.**

***Index Terms*—Lithography test patterns, pattern synthesis.**

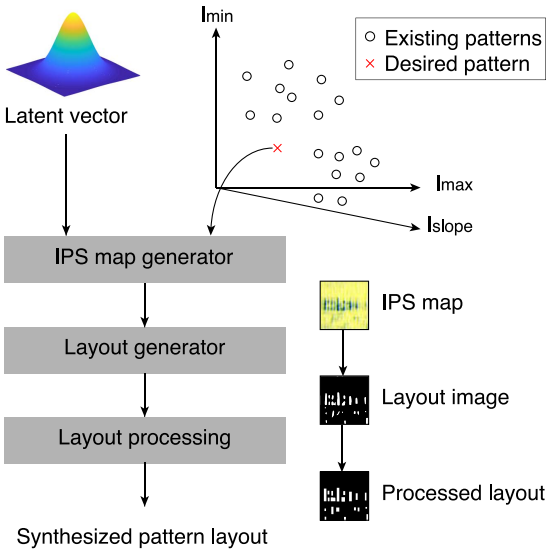

Fig. 1. Overview of the proposed pattern synthesis.

I. Introduction

A SMALL but diverse set of test patterns is essential for a number of lithography applications including calibration of lithography models, source mask optimization, and building hotspot library. A popular method to compile a list of test patterns is to use parameterized patterns, such as the dense line and space (DLS) and line-end to line-end (E2E) patterns. These are simple patterns, so their coverage of actual layouts is not sufficient, in particular for random patterns often found in metal layers. Another method is to prepare some test layouts, extract a number of small regions of layouts called clips, and use them for test patterns. Many clips are redundant or insignificant, and efficient extraction and classification [1] are important. But very often, their coverage is still not sufficient.

Manuscript received August 14, 2020; revised October 30, 2020; accepted January 13, 2021. Date of publication January 18, 2021; date of current version February 3, 2021. This work was supported in part by the National Research Foundation of Korea (NRF) of Ministry of Science and ICT (MSIT), under Grant 2019R1A2C2003402; in part by the World Class 300 Research and Development Project of MSS under Grant S2435123; and in part by the BK21 Plus Program funded by NRF. *(Corresponding author: Pervaiz Kareem.)*

The authors are with the School of Electrical Engineering, KAIST, Daejeon 34141, South Korea (e-mail: pervaiz@kaist.ac.kr; youngsoo@ee.kaist.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSM.2021.3052302.

Digital Object Identifier 10.1109/TSM.2021.3052302

A few methods have been proposed to generate synthetic test patterns. A simple method is to selectively locate predefined unit patterns on a grid to get mixed pattern shapes [2], [3], [4]. Diversity is limited due to a finite number of simple unit patterns. Machine learning techniques have recently been applied for synthesizing test patterns. A transforming convolutional auto-encoder has been employed [5] to generate new patterns from the existing ones while some perturbation is applied to the auto-encoder. Generative adversarial networks (GANs) have been applied [6] while the patterns are represented by a few discrete cosine transform (DCT) signals. A key idea is to use a GAN to generate some new DCT signals, which are then converted to a layout image through inverse DCT process.

A. Motivation and Proposed Method

Machine learning seems promising for test pattern synthesis, but both methods [5], [6] simply generate random patterns without particular directives. So the synthesized patterns should go through sanity check to see which patterns are necessary and which are not. In addition, if particular pattern shape is required, pattern synthesis and sanity check should only be repeated until the pattern is discovered.

0894-6507 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

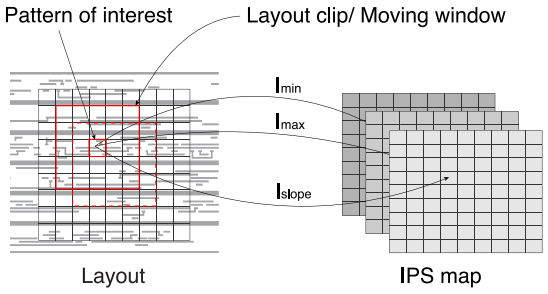

Fig. 2. IPS map representation of layout.

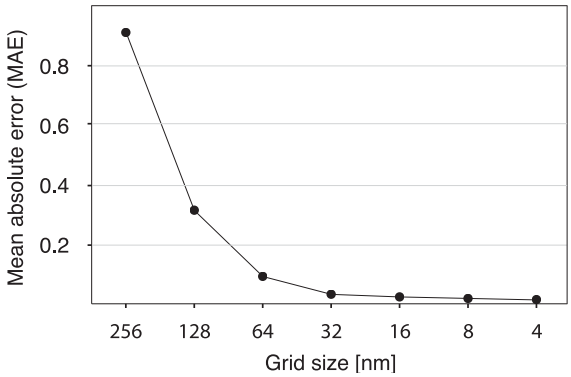

Fig. 3. Average MAE of 100 sample layout clips. Layout is represented by an IPS map with different grid sizes, shown on the x-axis. The y-axis shows the average quality of the layout clips reconstructed by the layout generator.

The proposed test pattern synthesis is illustrated in Fig. 1. We assume that each pattern in the layout is represented by IPS values, so the circles in the figure correspond to the patterns that exist in the sample layouts. Suppose that some values (cross mark) are picked from empty local IPS space. Those values are provided to the IPS map generator, together with a latent vector which models the random neighbors of the target pattern that we want to synthesize. The output of the generator is IPS map, a grid of IPS values, in which the center grid corresponds to the target pattern. We adopt a GAN and an auto-encoder for IPS map generator, which are trained with some sample layouts beforehand. The IPS map is provided to the layout generator; it consists of an auto-encoder network and is also trained beforehand. Note that IPS map and layout are not one-to-one correspondence, which is why a machine learning model is also employed here. The generated layout may contain some unrealistic shapes, which are processed through the layout processing step. A key in the proposed method is to generate the test pattern with given IPS values instead of random pattern, so the test pattern is dictated.

B. Paper Organization

The rest of this article is organized as follows. Section II describes the IPS map. The details of proposed test pattern synthesis are provided in Section III; they include IPS map generator, layout generator, and layout processing. Each component of pattern synthesis is experimentally assessed in Section IV. We address two example applications of test pattern synthesis in Section V: calibration of lithography model and machine learning-based lithography simulation. Section VI concludes the article.

II. Layout Representation Using IPS Map

A number of methods have been studied for layout representation. Optical kernels such as PFT (polar Fourier transform) signals have been used as they capture optical effects very well [7]. In optical proximity correction (OPC) applications, local layout densities are popular representations since they affect the extent of light diffraction and interference [8]. Both optical kernels and local layout densities have been used for etch proximity correction (EPC) applications [9], provided that optical kernels reflect the sidewall angle of photoresist (PR) after etching and the local layout densities provide the information of the number of etching particles. A tensor of DCT (discrete cosine transform) signals has been used to classify hotspot patterns [10], since preserving the spatial information of a layout clip is important in such application.

The focus of this article is to increase the diversity of layout patterns through pattern synthesis. We adopt image parameter space (IPS), which is popular to assess pattern diversity [11]. IPS consists of minimum intensity (I_{min}), maximum intensity (I_{max}), and maximum intensity slope (I_{slope}) measured with target pattern at the center of local aerial image (or light intensity map).

The layout is divided into a number of imaginary grids as shown in Fig. 2. For each grid, a layout clip is assumed with the grid positioned at the center of clip, whose size is 2um $\times$ 2um. The result is three sets of IPS values (called IPS map) as shown in Fig. 2, which can be considered as three channel image. In our machine learning model presented in Section III, IPS map is assumed a size of layout clip, i.e., the IPS map for whole layout is divided into a number of IPS maps with each map being the size of layout clip.

The size of the grid should be large to reduce the spatial correlation between grids, but it should be small enough so that a layout clip can be reconstructed from the IPS map. To decide the optimal size of the grid, we have constructed IPS maps with different grid sizes. For each grid size, 100 layout clips are reconstructed from the IPS map using the layout generator presented in Section III-B. The quality of the reconstructed layout clip is measured by mean absolute error (MAE):

$$MAE = \frac{1}{XY}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\left|l(x,y) - l'(x,y)\right|, \tag{1}$$

where l and l' are reference and reconstructed layout clips, respectively, while X and Y are the height and width of a layout clip, respectively. The result of experiment is shown in Fig. 3. After 32nm the MAE decreases very slowly with reducing grid size, so we assume the size of a single grid as 32nm $\times$ 32nm.

III. Synthesis of Test Patterns

Our proposed pattern synthesis flow, shown in Fig. 1, consists of three components: IPS map generator, layout generator,

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

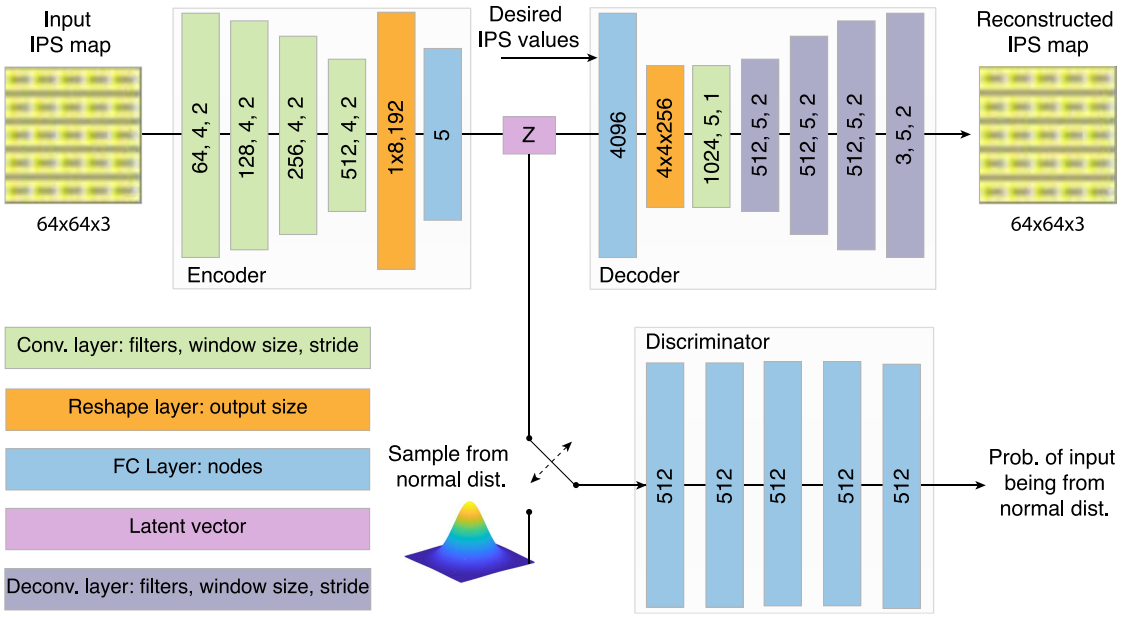

Fig. 4. Architecture of the IPS map generator. The top row network is an autoencoder that reconstructs the input IPS map from a latent vector and a set of IPS values. The discriminator network in the bottom row predicts the probability of input being from the standard normal distribution.

and layout processing. IPS map generator takes two inputs: a latent vector sampled from the standard normal distribution containing characteristics of the neighbor patterns, and a set of IPS values at the center of the layout clip. The output of IPS map generator is an IPS map corresponding to the inputs. The generated IPS map is given to the layout generator, which returns a layout clip corresponding to the map. The generated layout clip then goes through a final layout processing step, where its design rule violations are fixed and then converted to GDS format. In the next subsections, we present components of the proposed flow.

A. IPS Map Generator

The goal of the IPS map generator is to generate an IPS map with the desired IPS values at its center. To generate an IPS map that can be translated to a layout clip, the desired IPS values at the center are not sufficient but IPS values of neighbor patterns are also required. The neighbor information is provided through a latent vector.

The architecture of the IPS map generator is shown in Fig. 4. We selected this architecture because it is proven to be useful for generating images with certain properties such as a specific handwritten digit [12]. It consists of an encoder, a decoder, and a discriminator (D). In this setting, the encoder maps the input IPS map ($I(x, y)$) to a latent vector (Z). On the other hand, the decoder takes the output of encoder together with the IPS values at the center of the IPS map (I_c) and regenerates the IPS map ($I'(x, y)$). The discriminator network outputs the probability of input being from the standard normal distribution. The encoder and discriminator make a GAN, where encoder acts as the generator (G). The encoder and decoder make an auto-encoder network.

1) Role of GAN: The goal of GAN is to train encoder so that the latent vectors follow the standard normal distribution. Latent vectors need to follow a pre-defined distribution so that the decoder can learn which sample of the distribution corresponds to which type of neighbor patterns.

The individual goals of encoder and discriminator are opposite to each other. The discriminator receives a batch of features from the encoder, and another batch of samples drawn from the standard normal distribution and outputs the probability of its input being from the standard normal distribution. The goal of the discriminator is to correctly predict the probability of as many samples as possible. The goal of the encoder is to map IPS maps to such latent vectors that follow the standard normal distribution, so that the discriminator outputs a high probability for such inputs. Specifically, the cost function of this GAN is

$$\mathcal{L}_{GAN} = y_0\big[log(1 - D(G(I(x, y))))\big] + y_1\big[logD(X)\big], \quad (2)$$

where y_0 and y_1 are two binary numbers indicating that the input of discriminator is from the encoder and from the normal distribution, respectively, and $X \sim \mathcal{N}(0, 1)$. The objective of the discriminator is to maximize (2) while encoder tries to minimize it.

2) Role of Auto-Encoder: The auto-encoder network is trained to minimize the reconstruction error measured as the weighted sum of the root-mean square error (RMSE) between the input of the encoder and output of the decoder, and the RMSE between the desired IPS values and the IPS values at the center of reconstructed image (I'_c). Mathematically the loss function that needs to be minimized is given by

$$\mathcal{L}_{reconstruction} = \big\|I(x, y) - I'(x, y)\big\|_2 + \lambda_1\big\|I_c - I'_c\big\|_2, \quad (3)$$

where λ_1 is the weight given to the accuracy of desired IPS values. As the encoder is also part of the GAN, so besides minimizing reconstruction loss it also tries to generate latent vectors that follow standard normal distribution, so the final

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

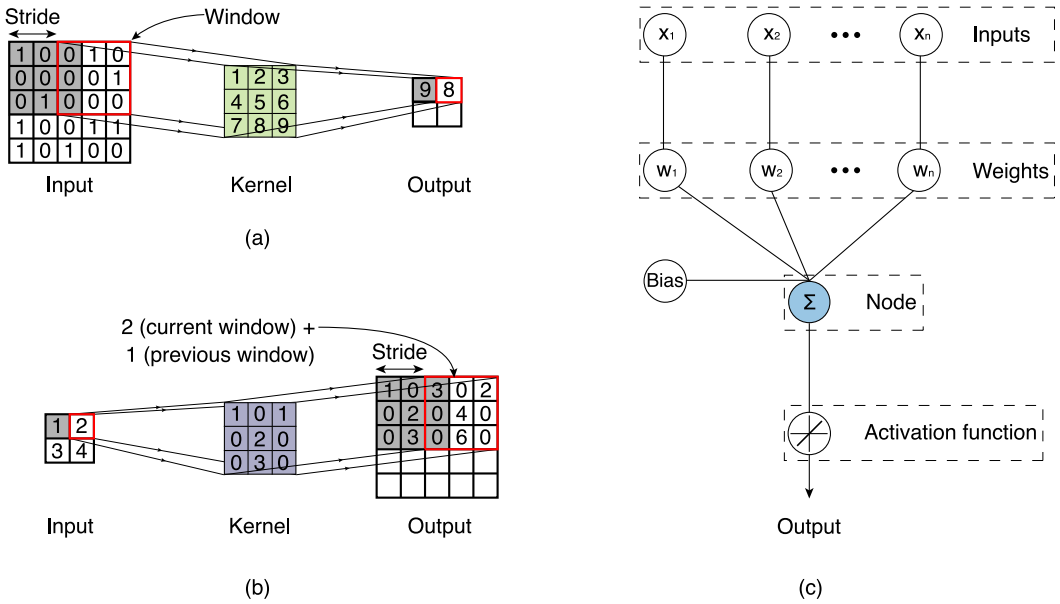

Fig. 5. Operation of (a) convolutional layer, (b) deconvolutional layer and (c) single node of a fully connected layer.

loss of auto-encoder is given by,

$$\mathcal{L}_{AE} = log(1 - D(G(I(x, y)))) + \lambda_2 \mathcal{L}_{reconstruction}, \quad (4)$$

where λ_2 is the weight given to the reconstruction loss.

3) Architecture of IPS Map Generator: The architecture of IPS map generator is shown in Fig. 4, which consists of convolutional, deconvolutional, and fully connected layers.

The convolutional layers extract useful features from the input by downsampling it. In each convolutional layer, several kernels, matrices with fixed elements, are convolved with the input [see Fig. 5(a)]. The size of a kernel is much smaller than the size of the input, so the kernel sweeps through the input step by step to generate the output. The step size of the kernel, called stride, controls the amount of downsampling of the input. The number of channels in a kernel is equal to that of the input, so for every window (step), every kernel is convolved with all channels of the input. The convolutional values obtained from a single kernel are accumulated and stored as one element in the output of the convolutional layer. In the output, the values of each kernel are stored as a separate channel. Thus the number of channels in the output of a convolutional layer is equal to the number of kernels in that layer.

The deconvolutional layers upsample the input using a number of kernels. As shown in Fig. 5(b), instead of convolving kernels with the input, all the elements of a kernel are multiplied with one element of the input. For every input element, the result is a matrix of the same size as that of the kernel. A single kernel is multiplied with an element of all channels of the input image, and the resulted matrices are accumulated to get one matrix. The resulting matrix from each kernel is stored in a different channel of the output image. The number of channels in the output of deconvolutional layer is the same as the number of kernels in that layer. Similar to the convolutional layer, the kernel sweeps through the input but this time step size is one. In the output image, the resulting matrices from two successive windows (steps) are placed overlapping, and the overlapping elements are accumulated. The difference between the starting points of two adjacent windows in the output is called stride of a deconvolutional layer that controls the amount of upsampling.

The fully connected layers consist of multiple nodes. An example of a node is shown in Fig. 5(c). Each node has its bias value, and every input is connected to each node of the layer trough an edge with a specific weight. A node takes a sum of weighted inputs (input values multiplied by the edge weights) and then adds node bias to it. The final output of each node is generated by applying an activation function (or threshold) to its value. The output of the convolutional layer in encoder is reshaped to a vector before inputting it to the fully connected layer. On the other hand, the output of fully connected layer in decoder is reshaped into a matrix and then inputted to the convolutional layer.

The optimized AAE network for our flow is obtained exhaustively. It has four convolutional layers and a fully connected layer in the encoder. The decoder has a fully connected layer, a convolutional layer, and four deconvolutional layers. The discriminator has five fully connected intermediate layers with 512 nodes in each layer. The window size of the encoder and decoder layers are 4 and 5, respectively. The stride of the encoder layers is 2. The convolutional and deconvolutional layers in the decoder have the strides of 1 and 2, respectively.

In the actual layout generation, only the decoder is used to generate new IPS maps. The purpose of GAN (encoder and discriminator) is to train the decoder. An IPS map is generated by sampling a latent vector from the standard normal distribution, and it is provided to the trained decoder together with the desired IPS values. In the generation process, the latent vector contains the characteristics of the neighbor patterns. The desired IPS values govern the patterns at the center of the layout clip as well as the height, width, and space parameters of the neighbor patterns. We also demonstrate the effect of latent vector and desired IPS values experimentally.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

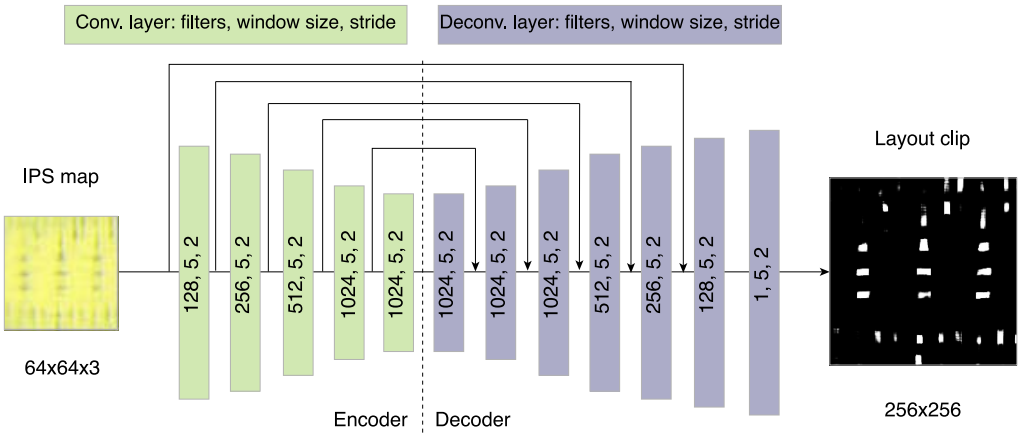

Fig. 6. Architecture of the layout generator. The convolutional layers extract features from the IPS map and the deconvolutional layers generate the layout clip corresponding to the input IPS map using features extracted by convolutional layers.

4) Training Procedure of IPS Map Generator: The values of convolution and deconvolution kernels, and node biases and edge weights of the fully connected layers are determined by a two-phase training procedure as follows.

The first phase is to train the discriminator. In this phase, a batch of inputs is sampled from the standard normal distribution and provided to the discriminator. The edge weights and node biases of the discriminator are updated such as the discriminator outputs a number closer to one, i.e., (2) is maximized. Now a batch of IPS maps is given to the encoder to generate the corresponding batch of latent vectors. The generated latent vectors are then given to the discriminator, and its edge weights and node biases are updated to output a probability closer to zero for such inputs.

The second phase is to train the encoder and decoder. For this purpose, another batch of IPS maps is given to the encoder to generate the corresponding latent vectors that are provided to the discriminator and decoder. The values at the center of IPS maps are also provided to the decoder as desired features. This time the edge weights and node biases of the discriminators are fixed, and the kernel values, node biases, and edge weights of the encoder and the decoder networks are updated such that the (4) is minimized. Both phases are repeated for one batch after the other for our network architecture.

B. Layout Generator

The goal of layout generator is to extract a layout that corresponds to the input IPS map. As IPS map to layout transformation is not one to one, therefore we need to find a layout clip corresponding to a certain IPS map. For this purpose, we use a variant of the U-Net [13], which consists of an encoder and decoder, as shown in Fig. 6. The encoder is a series of convolutional layers that extract information about the characteristics of the corresponding layout from the IPS map. The decoder takes the characteristics extracted by the encoder and upsamples them using deconvolutional layers to generate the corresponding layout clip. We also systematically provide the inputs of convolutional layers to the decoder, so that it can access any information that is lost during the encoding process. As shown in Fig. 6, the input of i_{th} convolutional layer is provided to the $(7-i)_{th}$ deconvolutional layer as well, where $i = 1, 2, \ldots, 5$.

In the training process, the objective of this network is to minimize the RMSE between the reference image and the generated layout image. Specifically, to minimize $\|l - l'\|_2$, where l is the reference layout clip and l' is the generated layout clip from the input IPS map. The kernel values of convolutional and deconvolutional layers, that fulfill this objective, are determined by training a network with IPS maps with their known reference layout clips.

The architecture of the layout generator, shown in Fig. 6, is also decided in exhaustive fashion. Its encoder and decoder consist of 5 convolutional layers and 7 deconvolutional layers, respectively. The window size and stride in all layers are 5 and 2, respectively. The input image size is $64\times64\times3$, and the output image size is 256×256.

C. Layout Processing

When IPS maps are converted to layout clips, then the shapes are not Manhattan, and some of them are very small or very close to each other. In this step, we modify the layouts such that they contain only Manhattan shapes that follow the minimum spacing and minimum dimension rules.

Our layout processing algorithm takes the output of the layout generator and finds the horizontal and vertical edges in it. The approximate polygons are then generated from the edges. Each polygon is converted to its minimum bounding rectangle, and those rectangles with any dimension smaller than the minimum allowed dimension are dropped. After these steps, we obtain Manhattan shapes that follow minimum dimension rules. In the next steps, we fix the minimum spacing violations. First, the rectangle pairs with space less than the minimum allowed space are identified. If any of the rectangles in each pair has more than the minimum space in the opposite direction to the space violation, then it is moved in that direction. If this movement does not resolve the violation, then rectangles are shrunk in opposite to the violation direction until the violation is resolved or rectangles reach the minimum dimension limit. If these steps do not resolve the space violation, then the original rectangle pair is restored, and the rectangle with

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

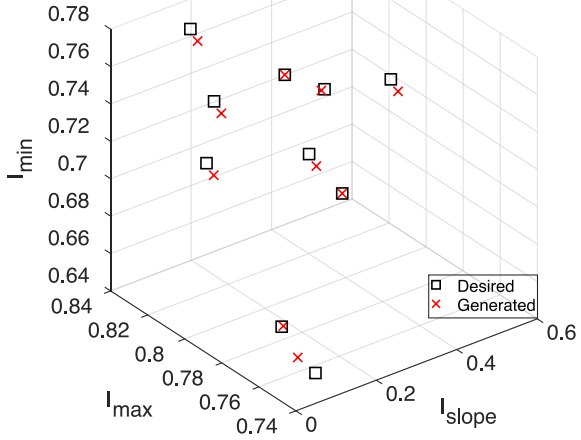

Fig. 7. Desired IPS values compared to the IPS values at the center of generated IPS map.

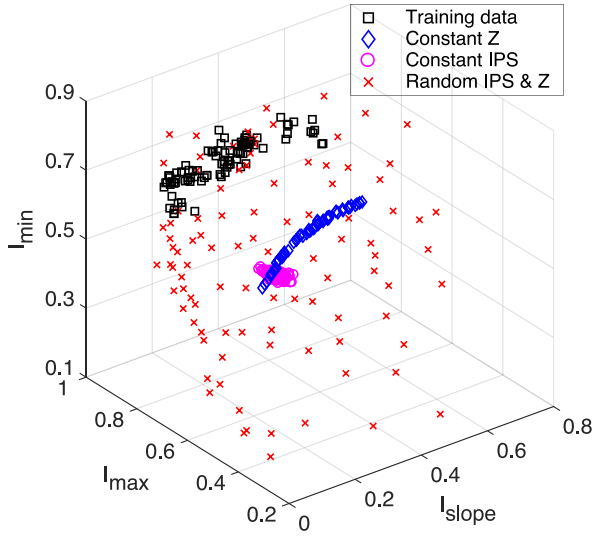

Fig. 8. IPS coverage plot, for training parametric patterns and synthetic patterns generated with different settings of IPS map generator.

the smaller area is dropped. After these steps, the layout is converted into the GDS format as a 2um $\times$ 2um clip.

IV. Assessment of Proposed Synthesis Method

The proposed pattern synthesis is implemented in Python using Keras [14] with TensorFlow [15] backend. An aerial image is generated through Proteus [16]. Adam optimizer [17] is used to train both the IPS map generator and the layout generator. The learning rate of the two generators is set to 0.0002 and 0.001, respectively. The value of λ_1 in (3) is 10 and that of λ_2 in (4) is 100. The size of latent vector is set to 5 in our experiments.

For training of the two generators, 5K parametric layout clips are prepared using a commercial pattern generator [18]. Each layout clip is 2um $\times$ 2um; it is stored as 256$\times$256 binary image. Each clip is represented by IPS map, which is three channel image with each channel being 64$\times$64. So, one set of IPS values correspond to 4$\times$4 region of layout clip image. The training of the layout generator and the IPS map generator take 1 and 5 hours, respectively. To assess the two generators, another 5K layout clips are extracted, but this time from a contact layer of an actual N10 memory circuit.

A. IPS Map Generation

To generate an IPS map with given IPS values, the IPS map generator is trained with the IPS maps of the training set. We evaluate the trained IPS map generator in the following four settings.

First, we check the capability of the IPS map generator by generating unseen IPS maps from their latent vectors and desired IPS values, while reference IPS maps being known. Instead of providing random latent vectors, we extract latent vectors of the test set IPS maps using the encoder of the IPS map generator. These latent vectors together with the IPS values of the corresponding layout clip (desired IPS values) are provided to the decoder. We compare the generated IPS maps and reference IPS maps pixel by pixel. Results show that the average Euclidean distance between a pixel of a generated IPS map to the corresponding pixel of the reference IPS map is only 0.012. We choose ten random samples from IPS space as shown in Fig. 7. Note that both generated and desired (reference) layout clips are very close in IPS. We also check the average Euclidean distance between desired and generated features in IPS, which is only 0.0068. The pixel at the center of generated IPS maps and reference IPS maps are even closer to each other compared to the pixels at other regions because of extra weight given to this pixel during training. These results demonstrate that the IPS map generator returns the layout that exactly matches the provided IPS values with the pattern at the layout center.

In the next experiment, we show the effect of the latent vector on generated layouts. We pick just one latent vector from the test set of patterns and change the desired IPS values. The generated IPS maps are marked as Constant Z in Fig. 8. It can be seen that although the latent vector is the same, the IPS values of the generated layouts are different because of the different desired IPS values. We show the layouts generated from these IPS maps in Fig. 9(a). It can be observed from these layouts that the basic shape of the pattern, which is a row of contacts at the center of the clip, is the same because of the constant latent vector. However, the parameters such as the width and height of contacts, and space between them are changing based on the desired IPS values.

Next, we fix the desired IPS values and randomly sample 100 latent vectors from the standard normal distribution, to show that we can generate different patterns with the same IPS values. The results of this experiment are marked as Constant IPS in Fig. 8. Although the generated IPS maps can not be mapped to a single point in IPS, they still located in a very small span. This small difference between desired and generated IPS values is mainly because for some patterns it is impossible to have the exact desired IPS values due to the neighbor pattern characteristics provided by the latent vector. The qualitative results of this experiment are shown in

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

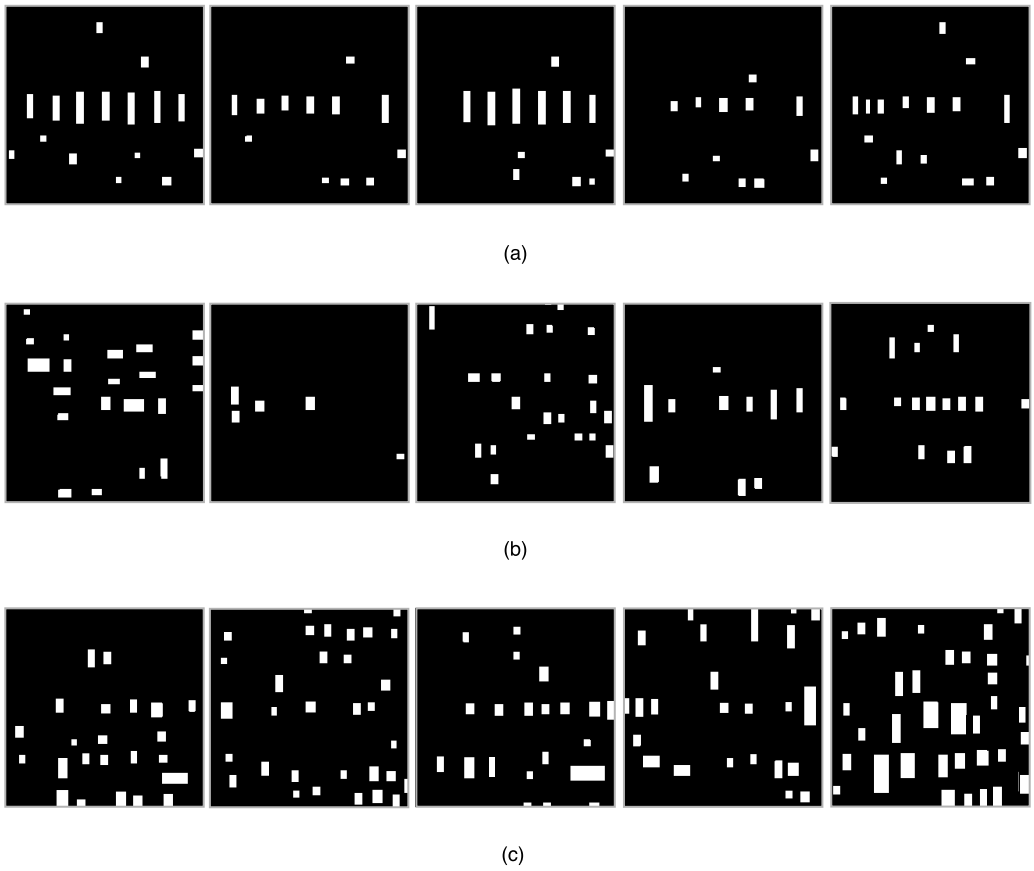

Fig. 9. Generated layouts, (a) when the latent vector is kept constant and input IPS values are changed systematically, (b) when input IPS values are kept constant but the latent vector is changed randomly and (c) when both the input IPS values and the latent vector are randomly changed.

Fig. 9(b). It can be seen that in this case, the neighbor pattern shapes are changing due to different latent vectors but the width and height of the contacts in the central region are similar due to the same desired IPS values.

As the last experiment for the IPS map generator, we give 100 sets of desired IPS values randomly sampled from the IPS. The latent vectors are also randomly drawn from the standard normal distribution. The resulted IPS maps are marked as Random IPS & Z in Fig. 8. Notice that 100 layouts generated with these settings cover more space compared to randomly sampled 100 samples from the training data that are marked as Training data in Fig. 8. We further show the qualitative results of this experiment in Fig. 9(c). The neighbor pattern shapes of generated layouts are different due to different latent vectors, and height, width, and space parameters at the center of the clips are also different because of desired IPS values.

B. IPS Map to Layout Transformation

As shown in Fig. 1 and presented in Section III, the generated IPS maps are converted to the corresponding layout clips using the layout generator. We train the layout generator using IPS maps and layout clips from the training set. Once the layout generator is trained, we use IPS maps from the test set to generate corresponding layouts. Now we compare the generated layouts with the reference layout clips. Only 4.03% of the pixels are different in a generated layout compared to its reference layout clip, on average. This result validates that our network can generate the correct corresponding layout clip for a given input IPS map.

C. Layout Processing

When the layout generator transforms IPS maps into layout clips then some of them may not follow basic design rules. The objective of layout processing is to fix design rule violations while preserving the features of the input layout. In this step, the shapes are converted to rectangles. Those with heights and widths smaller than a certain threshold are removed. Also, the rectangles with space less than the minimum desired distance from their neighbors are moved or shrunk.

To verify that processed layout clips follow basic design rules, we check the minimum dimension of polygons and minimum space between two polygons using KLayout [19]. Results show that none of the dimensions and spaces are less than 24nm and 56nm, respectively, which we suppose as the minimum requirements in the layout processing step. We confirm the feature preservation property of the layout processing step by comparing the IPS values of layout clips before and after the layout processing. IPS maps generated from randomly sampled latent vectors, and random input features (last experiment of Section IV-A) are provided to the layout generator to obtain layout clips. These layout clips are processed in the layout processing step. Now we compute the IPS values at the

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

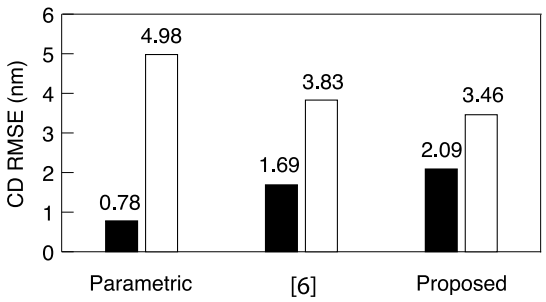

Fig. 10. Results of lithography model calibration using three different set of test patterns. Black bar is CD RMSE for calibration patterns and white bar is CD RMSE for patterns unknown in calibration.

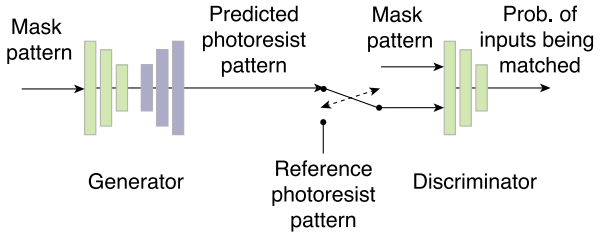

Fig. 11. Architecture of GAN-based lithography simulation.

center of the processed layout clips and compare them with those of the corresponding unprocessed layout clips. The average Euclidean distance between the IPS values of processed and unprocessed layout clips is 0.0107 only. Hence, we can conclude from these results that processed layout clips follow basic design rules, and most of the features are preserved in this processing step.

V. Applications of Test Pattern Synthesis

As sample applications of proposed test pattern synthesis, we choose calibration of lithography model and lithography simulation using machine learning model.

A. Calibration of Lithography Model

Lithography model consists of optical model to reflect light exposure and resist model to reflect photoresist development. It is an empirical model, so test patterns are used to calibrate a number of model parameters. In resist model, in particular, some Gaussian kernels are convolved with the aerial image to get the photoresist pattern. The result of convolution is compared to some (constant or variable) threshold to decide whether photoresist is fully developed [20], [21]. Model calibration determines the parameters of the kernels and threshold to accurately simulate the resist pattern. The model accuracy is assessed through CD comparison between reference and simulated pattern. In our experiments, the reference CD data is obtained from rigorous simulations [22].

We first use 100 parametric patterns from a commercial layout pattern generator [18] to build a model. The model accuracy is shown in Fig. 10 with heading of Parametric. The black bar shows the RMSE of model when the same 100 parametric patterns are used to assess the quality of model. The white bar is the RMSE of model when randomly chosen 1000 patterns from an actual N10 memory circuit are used to assess this model. Clearly, the model is accurate for the patterns that have been used to build the model but RMSE of the model for real patterns is much higher. This indicates that the parametric patterns are not diverse enough to represent all the patterns that can arise in real layouts.

Next, the lithography model is built with 100 test patterns from pattern synthesis flow proposed in [6]. The results of this model are labeled as [6] in Fig. 10. The black bar shows the RMSE of model when tested with the same calibration patterns. Patterns are now diverse than the parametric patterns that explains why the calibration error is increased. However, when tested with the real patterns from the N10 memory circuit, the same patterns that are used to assess the previous model, the RMSE is reduced by 23.1% compared to the preceding model.

We finally employ our proposed pattern synthesis to build the lithography model. We randomly sample 100 latent vectors from the standard normal distribution. For desired IPS values we uniformly sample 100 sets of values from the IPS. The results of the model calibrated with these patterns are marked as Proposed in Fig. 10. The black bar shows that the diversity of patterns is now increased, which helped to reduce the RMSE of the white bar. If we compare black bars in Fig. 10, then the diversity of patterns generated from the proposed flow is higher than that of the patterns generated with [6]. This increase in diversity is thanks to controlling the features of the patterns. By calibrating model with layout patterns generated from the proposed flow, the RMSE of unseen patterns is reduced by 30.5% and 9.7% compared to models calibrated with parametric patterns and layout patterns from [6], respectively.

B. Lithography Simulation Using GAN

Recently, GAN has been used as a lithography simulation [23]. Since the accuracy of GAN depends on the test patterns, we choose this simulation method as our next application. An image to image transforming conditional GAN (cGAN), shown in Fig. 11, is used to predict the photoresist pattern with mask pattern given at the input. The network architecture of cGAN is adopted from [24]. It consists of an auto-encoder-based generator and a convolutional neural network-based discriminator. The discriminator is only used in the training phase. Once the cGAN is trained we only use the generator to get the photoresist pattern for the input mask pattern. The reference photoresist patterns of training and test sets are obtained from a conventional lithography simulation model [16].

We first train a network by providing mask patterns from the 5K parametric patterns, generated from a commercial pattern generator [18], together with their reference photoresist images. The trained network is then assessed by providing 10K mask patterns from an actual N10 memory circuit together with parametric tip to tip and tip to line patterns, 1K each. These parametric patterns are included to make sure that test set contains patterns that are important for evaluating a lithography model. We compute pixel-wise similarity in the central

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF PHOTORESIST PATTERN PREDICTION ACCURACY

	Trained with parametric patterns	Trained with generated patterns
Pixel accuracy	87%	94%
Average EDE	3.64nm	3.31nm
Standard deviation of EDE	2.32nm	2.17nm

grid of 64nm $\times$ 64nm in predicted and reference photoresist images; we also check the average edge displacement error (EDE) in this region. The results are shown in the second column of Table I.

We now generate 5K layout clips through our pattern synthesis with desired IPS values uniformly sampled from IPS space and latent vectors randomly drawn from the standard normal distribution. The layout clips are applied to train the cGAN. Lithography simulation with new cGAN is assessed with the same test set that is used to assess the previous model. The last column of Table I shows the results of this model. The pixel-wise accuracy, average EDE, and standard deviation of EDE are improved by 7%, 9.1%, and 6.5%, respectively. The patterns are now diverse than those generated by the commercial tool, therefore, the current model performs better than the previous model for the unseen patterns. These results are evident that the layouts generated from our proposed flow can be used for machine learning-based applications that require diverse patterns.

VI. CONCLUSION

Test pattern synthesis using machine learning model has been proposed. It returns a pattern layout with its center corresponding to given IPS values, which is a key difference compared to related works. It relies on two auto-encoders: an adversarial and a U-Net-based. The adversarial auto-encoder receives a latent vector and a vector of desired IPS values and outputs an IPS map. The U-Net-based auto-encoder converts an IPS map to a corresponding layout image. The layout image is further processed to make it design rule clean and to convert it to GDS format. It has been shown through experiments that the proposed method indeed generates the layout with desired features. The synthesized test patterns have been used to calibrate a lithography model parameters and also to train a GAN for lithography simulation, as sample applications. Results have shown that the error of lithography model is reduced by 30.5% when the model is built with the test patterns from the proposed method as opposed to the model with popular parametric patterns. The accuracy of lithography simulation is improved by 7% when GAN is trained with our test patterns instead of parametric patterns.

REFERENCES

[1] S. Shim and Y. Shin, "Topology-oriented pattern extraction and classification for synthesizing lithography test patterns," *J. Micro/Nanolithography MEMS MOEMS*, vol. 14, no. 1, pp. 1–12, Jan. 2015. [Online]. Available: https://doi.org/10.1117/1.JMM.14.1.013503

[2] S. Maeda, R. Ogawa, S. Shibazaki, and T. Nakajima, "Novel method for quality assurance of two-dimensional pattern fidelity and its validation," in *Proc. SPIE Photomask Next Gener. Lithography Mask Technol. XIV*, vol. 6607, May 2007, pp. 191–201. [Online]. Available: https://doi.org/10.1117/12.728936

[3] A. Hamouda *et al.*, "Enhanced OPC recipe coverage and early hotspot detection through automated layout generation and analysis," in *Proc. SPIE Opt. Microlithography XXX*, vol. 10147, Mar. 2017, pp. 223–231. [Online]. Available: https://doi.org/10.1117/12.2260769

[4] G. R. Reddy, M.-M. Bidmeshki, and Y. Makris, "VIPER: A verstile and intuitive pattern generator for early design space exploration," in *Proc. Int. Test Conf.*, Nov. 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1109/ITC44170.2019.9000169

[5] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "DeePattern: Layout pattern generation with transforming convolutional auto-encoder," in *Proc. Design Autom. Conf.*, Jun. 2019, pp. 148–153. [Online]. Available: https://doi.org/10.1145/3316781.3317795

[6] P. Kareem, Y. Kwon, and Y. Shin, "Layout pattern synthesis for lithography optimizations," *IEEE Trans. Semicond. Manuf.*, vol. 33, no. 2, pp. 283–290, May 2020. [Online]. Available: https://doi.org/10.1109/TSM.2020.2982989

[7] S. Choi, S. Shim, and Y. Shin, "Neural network classifier-based OPC with imbalanced training data," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 5, pp. 938–948, May 2019. [Online]. Available: https://doi.org/10.1109/TCAD.2018.2824255

[8] A. Gu and A. Zakhor, "Optical proximity correction with linear regression," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 263–271, May 2008. [Online]. Available: https://doi.org/10.1109/TSM.2008.2000283

[9] S. Shim and Y. Shin, "Machine learning-guided etch proximity correction," *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 1, pp. 1–7, Nov. 2017. [Online]. Available: https://doi.org/10.1109/TSM.2016.2626304

[10] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1175–1187, Jun. 2019. [Online]. Available: https://doi.org/10.1109/TCAD.2018.2837078

[11] P. Filitchkin, T. Do, I. Kusnadi, J. L. Sturtevant, P. de Bisschop, and J. V. de Kerkhove, "Contour quality assessment for OPC model calibration," in *Proc. SPIE Metrol. Inspection Process Control Microlithography XXIII*, vol. 7272, Mar. 2009, pp. 841–847. [Online]. Available: https://doi.org/10.1117/12.814662

[12] A. Makhzani, J. Shlens, N. Jaitly, I. J. Goodfellow, and B. Frey, "Adversarial autoencoders," May 2015. [Online]. Available: arXiv:1511.05644.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.*, Oct. 2015, pp. 234–241. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28

[14] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[15] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX Symp. Oper. Syst. Design Implement.*, Nov. 2016, pp. 265–283.

[16] *Proteus*, Synopsys, Mountain View, CA, USA, Jun. 2014.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014. [Online]. Available: arXiv:1412.6980.

[18] *Proteus Metrokit*, Synopsys, Mountain View, CA, USA, Jun. 2018.

[19] M. Köfferlein. (2020). *KLayout—HIgh Performance Layout Viewer and Editor*. [Online]. Available: http://www.klayout.de

[20] T. A. Brunner and R. A. Ferguson, "Approximate models for resist processing effects," in *Proc. SPIE Opt. Microlithography IX*, vol. 2726, Jun. 1996, pp. 198–207. [Online]. Available: https://doi.org/10.1117/12.240906

[21] J. Randall, K. G. Ronse, T. Marschner, A.-M. Goethals, and M. Ercken, "Variable-threshold resist models for lithography simulation," in *Proc. SPIE Opt. Microlithography XII*, vol. 3679, Jul. 1999, pp. 176–182. [Online]. Available: https://doi.org/10.1117/12.354329

[22] *Sentaurus Lithography*, Synopsys, Mountain View, CA, USA, Jun. 2018.

[23] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *Proc. Design Autom. Conf.*, Jun. 2019, p. 107. [Online]. Available: https://doi.org/10.1145/3316781.3317852

[24] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1125–1134.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on February 10,2021 at 15:01:41 UTC from IEEE Xplore. Restrictions apply.