This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

Layout Pattern Synthesis for Lithography Optimizations

Pervaiz Kareem, Yonghwi Kwon, and Youngsoo Shin, *Fellow, IEEE*

***Abstract*—A set of comprehensive test patterns is important for a number of lithography applications. Pattern diversity is, however, hard to achieve either from parametric patterns or from actual patterns even though they are carefully extracted and classified. Automatic layout pattern synthesis is proposed in this paper. A generative adversarial network (GAN) is employed to generate a new set of discrete cosine transform (DCT) signals. It is converted to an image format through inverse DCT (IDCT). The image is blurred since output DCT signals from GAN correspond to lower frequency region. Another GAN, this time a conditional GAN (cGAN), is introduced to get sharpened layout pattern. A key in this process is to train the two GANs in such a way that generated patterns are different from existing actual patterns while they are still valid layouts. Experiments indicate that synthetic patterns are less redundant and cover 76% more space in image parameter set space than actual patterns. We choose a machine-learning guided OPC as an example application: when synthetic patterns are included to train OPC model, edge proximity error decreases by 21%. Resist model calibration is chosen as a second example: when synthetic patterns are combined with parametric- and real-patterns, CD RMSE decreases by 10.3%.**

***Index Terms*—Automatic layout synthesis, Layout pattern coverage, Resist modeling, OPC, ML-OPC**

I. Introduction

Comprehensive test patterns are important for a number of lithography applications such as source mask optimization, hotspot library construction, resist model calibration, and more recently building a machine-learning model. Two types of test patterns are puplar: parametric patterns and actual patterns [1]. Parametric patterns are represented by a few geometrical parameters, e.g. width and space. They are easy to compile and analyze, but cannot cover complex shapes. Actual patterns are extracted from sample layouts [2], and are useful for random shapes often found in metal routing. Pattern extraction is important so that only significant ones are extracted; pattern classification is also important to filter out redundant patterns. Both types lack pattern diversity, which we demonstrate in our experiments.

A few approaches have been proposed to generate new patterns. A simple approach is to rotate or flip existing layouts to get new versions [3]; conditionally placing unit patterns on the grid give a number of new layout patterns [4], [5]. However, they are all deterministic methods and so the diversity of generated layouts is still limited. Transforming convolutional auto-encoder has been applied to generate the layout [6]. This method still relies on existing patterns to get a pattern description, which is then altered, and its effectiveness has not been evaluated with actual lithography applications.

Manuscript received January 1, 2020; revised February 26, 2020; accepted March 21, 2020. This work was partly supported by the National Research Foundation of Korea (NRF) of MSIT (No. 2019R1A2C2003402), World Class 300 R&D Project of MSS (No. S2435123) and BK21 Plus Program funded by NRF.

The authors are with the School of Electrical Engineering, KAIST, Daejeon 34101, Korea (e-mail: {pervaiz, yh.kwon}@kaist.ac.kr; youngsoo@ee.kaist.ac.kr).

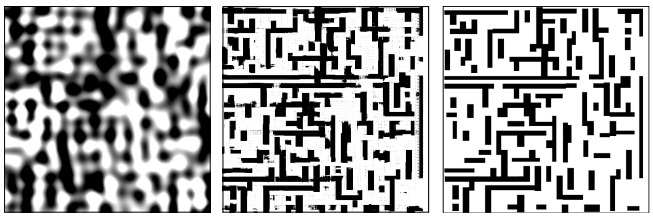

Fig. 1. (a) After IDCT performed on DCT-GAN output, (b) after applying deblurring-cGAN, and (c) final synthesized layout pattern after post processing.

A. Contributions

Automatic layout pattern synthesis is proposed in this paper. We assume that a layout pattern is represented by a number of DCT signals, but only in lower frequency region. A GAN, called DCT-GAN, is employed to generate new DCT signal values. DCT-GAN is trained beforehand to ensure that output DCT signals are new but ultimately correspond to a valid layout. IDCT is applied to the output of DCT-GAN to reconstruct an image. The image is still blurred as shown in Fig. 1(a) since high frequency DCT components are missing. A cGAN, named deblurring-cGAN, is applied to sharpen the image; an example output is shown in Fig. 1(b). It is important to train deblurring-cGAN such that sharpening is performed as it would on layout not on arbitrary image. Post processing is finally performed to remove any noises that usually do not exist in valid layout; Fig. 1(c) shows an example outcome.

We address two sample applications of proposed layout synthesis flow: machine learning guided OPC (ML-OPC) and resist model calibration. For ML-OPC, we run model-based OPC (MB-OPC) with synthesized patterns to obtain reference data. Then it is used as additional training data for the ML-OPC model. Similarly, for resist model calibration, we obtain reference CD data on synthesized layouts; they are used to calibrate resist model along with real layout patterns and conventional test patterns. The proposed layout synthesis flow improves the accuracy of each application by increasing the coverage of the model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

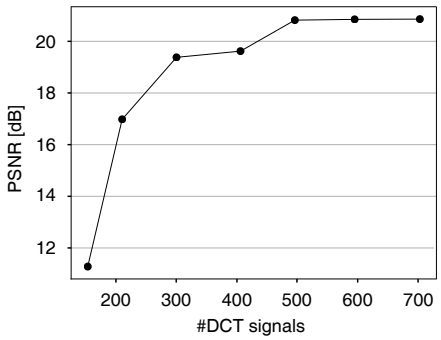

Fig. 2. Average PSNR of 100 sample clips. Each clip is represented by DCT signals, which then go through IDCT and deblurring to reconstruct the original clip. PSNR is the indication of the quality of reconstructed image.

B. Paper Organization

The remainder of this paper is organized as follows. In Section II, layout representation using DCT signals is presented, followed by the overview of two example applications of layout pattern synthesis: ML-OPC and resist model calibration. Section III addresses the overall flow of layout pattern synthesis as well as each synthesis component, namely DCT-GAN, deblurring-cGAN, and layout processing. In Section IV, layout pattern synthesis is experimentally assessed and we evaluate the benefit of synthetic patterns when they are applied to ML-OPC and resist model calibration. The paper is summarized in Section V.

II. Preliminaries

A. Layout Representation with DCT Signals

A variety of layout representation has been tried in lithography applications. A set of local layout densities has been used for OPC applications [7], given that light diffraction and interference are affected by such densities. Topology-based representation has been applied [8] for compact storage of layout data. DFT (discrete Fourier transform) spectrum has been used [1] for layout pattern matching applications; the two patterns, in which one pattern is shifted or reflected from the other, have a similar spectrum and so matching such patterns becomes easier. PFT (polar Fourier transform) signals have been used for OPC applications [9]; they are the basis of light intensity after the patterning process, so using them is a convenient way to capture optical diffraction and interference.

In this paper, we use DCT to represent a layout clip, which is a small layout pattern with particular layout segment of interest at its center as shown in Fig. 3. A layout often contains repeated patterns, and DCT can capture such repetition with a smaller number of frequency signals. A layout clip is an array of binary pixels $l(x, y)$; pixel value is 0 where a pattern exists and 1 otherwise. The DCT of a clip for a basis function of order mn is given by

$$L(m,n) = c_m c_n \sum_{\forall_{x,y}} l(x,y)\omega_{mn}(x,y), \tag{1}$$

where c_m and c_n are scaling factors and

$$\omega_{mn}(x,y) = \cos\left(\frac{\pi(2x+1)m}{2H}\right)\cos\left(\frac{\pi(2y+1)n}{2W}\right). \tag{2}$$

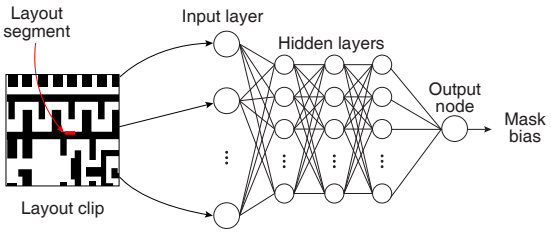

Fig. 3. ML-OPC using MLP network.

The H and W represent the clip size, in height and width, respectively. Transformation from DCT domain back to clip image is performed through IDCT, specifically

$$l(x,y) = \sum_{\forall_{m,n}} c_m c_n L(m,n)\omega_{mn}(x,y). \tag{3}$$

The number of DCT signals should be small for efficiency and flexibility of generating a new clip in our approach, but it also should be large enough to contain enough clip information. To determine the right number of DCT signals, we take 100 sample clips. Each clip is represented by some fixed number of lowest frequency DCT signals. Clip image reconstruction is then performed by IDCT and deblurring (using our cGAN presented in Section III). The quality of reconstructed image is measured by

$$\text{PSNR} = -10\log_{10}\text{MSE}, \tag{4}$$

where MSE is mean squared error and is given by

$$\text{MSE} = \frac{1}{HW}\sum_{x=0}^{H-1}\sum_{y=0}^{W-1}[l(x,y) - l'(x,y)]^2, \tag{5}$$

where $l'(x, y)$ is reconstructed layout clip.

Fig. 2 shows the average PSNR values of 100 sample clips while the number of DCT signals is varied. As the figure indicates, about 500 is a good number since PSNR does not change much beyond this number. We use 496 in all our experiments, specifically $m = 1, 2, \ldots, 31$ and $n = 1, 2, \ldots, 32 - m$.

B. ML-OPC

One of the example applications of layout pattern synthesis is ML-OPC, OPC driven by a machine learning model. Any ML-OPC method can be chosen, but we use the method based on multi-layer perceptron (MLP) network [9]. This method is illustrated in Fig. 3. Layout clip is represented by a number of PFT signals, which are provided to the input nodes of the network. Values propagate through one or more hidden layers, and one output node returns a mask bias value, the amount of shift that should be applied for a layout segment located in the center of clip. The network is trained to determine edge weights and node biases with a number of training clips, while their mask bias values returned by standard MB-OPC are considered as target values that the network should predict.

One of the limitations of ML-OPC is lack of diversity of training clips. Ideally PFT space (the space spanned by the parameters given to input layer) should be full of the points

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

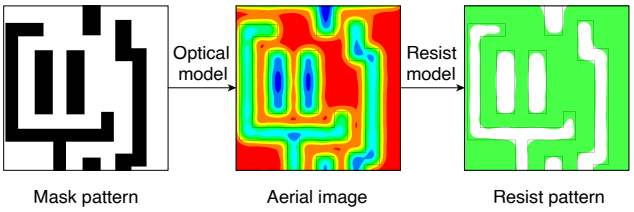

Fig. 4. Model-based lithography simulation.

represented by training clips, which is practically difficult to achieve. So if clips that correspond to empty region of the space are provided to MLP, mask bias prediction becomes inaccurate. For instance, for a clip whose distance to the nearest training clip in PFT space is 10.28, the root-mean-square error (RMSE) of predicted mask bias is 1.25. For a clip that is far away with distance of 68.99, the RMSE becomes 12.26. This limitation may be resolved by adding synthetic patterns in the training process.

C. Resist Model Calibration

Resist model calibration is another example application of our layout pattern synthesis. Fig. 4 illustrates the steps of lithography simulation. Aerial image is obtained by applying optical model to an input mask pattern [10], [11]. Sum of coherent sources approximation [12] is a popular optical model, in which light intensity is described by a number of convolutions between mask pattern and each of a few coherent imaging kernels. To get a resist pattern, we often take a convolution between the aerial image and some Gaussian to simulate chemical reaction. The result is compared to some (constant or variable) threshold to determine whether resist is finally developed [13], [14].

Resist model is constructed to determine such Gaussians and thresholds in a way that the simulated resist pattern and actual measured pattern are similar (usually in terms of CDs) for a number of test layout patterns. The quality of resist model is thus highly dependent on the diversity of test patterns, and so synthetic patterns added on some existing test patterns may improve such quality.

III. Layout Synthesis Flow

The goal of layout synthesis is to yield diverse synthetic layout clips for complex metal layers such as metal 1 (M1). The proposed synthesis flow is shown in Fig. 5. DCT-GAN outputs a few DCT signals. IDCT is applied to construct the corresponding synthetic clip. Since the number of DCT signals is limited to 496 in low frequency range as explained in Section II-A, the resultant clip is blurred. Deblurring-cGAN is applied to sharpen the clip, followed by layout processing to improve the quality of the clip.

A. DCT-GAN

DCT-GAN is responsible for generating new DCT signals that will correspond to a new image (different from actual clip images used to train the network) but the image can still be

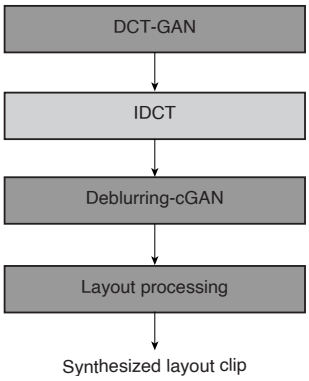

Fig. 5. Proposed flow for layout synthesis.

considered as a valid layout clip. The architecture of DCT-GAN is shown in Fig. 6(a). It consists of two MLP networks called generator (G) and discriminator (D). The generator gets 10 random numbers (z) in [-1, 1] range[1] and outputs 496 DCT signals $(G(z))$. The discriminator is a classifier that is required to train the generator. It receives DCT signals either from the generator $(G(z))$ or from actual training clips (x) and predicts the probability of input being from training clips $(D(.))$. Its objective is to classify input DCT signals in a way that output becomes 1 (predicted probability greater than a threshold) if the network considers the inputs correspond to the training clips and 0 (predicted probability less than a threshold) if the inputs are considered to be synthesized ones. The objective of the generator is to generate samples that can make the output of discriminator 1. The objective function of DCT-GAN can be expressed as

$$\mathcal{L}_{GAN}(D,G) = y_1[\log D(x)] + y_0[\log(1 - D(G(z))], \quad (6)$$

where y_0 and y_1 are binary numbers that indicate that input is from the generator and real training clips, respectively. Generator tries to minimize (6) while discriminator tries to maximize it.

The goal of the training process is to determine the edge weights and the node biases for generator and discriminator networks considering their objectives. A set of training clips with their corresponding DCT signals are grouped into a number of batches, and the following training process is repeated for one batch after the other.

1) A series of 496 DCT signals are provided to the discriminator one by one. Its edge weights and node biases are updated such that classification error at the network output is minimized, i.e. (6) is maximized.
2) A set of DCT signals of one batch size is generated through the generator, and provided to the discriminator. The discriminator network is updated such that classification error is minimized (the expected output, this time, is 0).
3) A set of DCT signals of one batch size is generated through the generator again, and provided to the dis-

[1]The number of generator inputs is determined experimentally. It is observed that MLP works best if the numbers are in [-1, 1] range [15].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

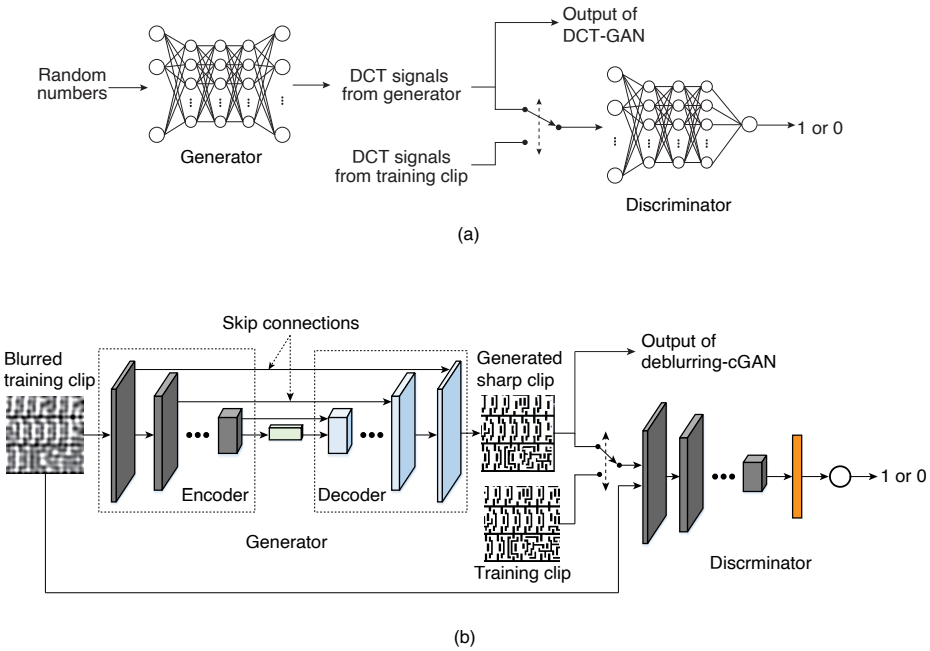

Fig. 6. (a) DCT-GAN and (b) deblurring-cGAN.

criminator. The generator network (i.e. its node biases and edge weights) is updated such that classification error at the discriminator output is maximized (i.e. $\log(1 - D(G(z))$ is minimized). This is because the objective of the generator is to yield DCT signals that can be considered as the ones from a real layout clip.

The above process is for a given architecture (the number of layers and the number of nodes in each layer) of generator and discriminator networks. The best architecture is obtained in exhaustive fashion; it corresponds to 128, 256, and 512 nodes in three intermediate layers of the generator and 256, 128, 64, and 32 nodes in four intermediate layers of the discriminator in our experiment. Note that the discriminator is only for the training process, and the output of the generator becomes the output of the DCT-GAN in the end as shown in Fig. 6(a).

B. Deblurring-cGAN

Once DCT signals are generated by the GAN, they go through IDCT so that a corresponding image can be constructed (see Fig. 5). Since we keep only the limited number of 496 signals in low frequency range as explained in Section II-A, the constructed image will be blurred as shown in Fig. 1(a). We choose cGAN to sharpen the image because sharpening should be performed so that the final image looks like layout clips that have been used to train cGAN.

Deblurring-cGAN also consists of a generator (G) and a discriminator (D); its architecture is shown in Fig. 6(b). In cGAN, the generator receives a blurred image (I) of 512×512 pixels and it yields a sharpened image of the same size $(G(I))$. The discriminator is a classifier that requires two inputs: a blurred clip (I) and a sharp clip either from the generator $(G(I))$ or from actual training clips (l). Its goal is to output 1 if the input sharp image is from training clips and corresponds to the given blurred clip, and 0 otherwise. In the training process of deblurring-cGAN, the objective of the discriminator is the same as that of discriminator of DCT-GAN, which is to correctly classify the given samples as much as possible, specifically to maximize

$$\mathcal{L}_{cGAN}(D) = y_1[\log D(I,l)]+y_0[\log(1-D(I,G(I))]. \quad (7)$$

The objective of the generator is different, it not only aims to make the output of discriminator 1 but also tries to generate a pixel by pixel similar image to the reference sharp clip. So its objective is to minimize

$$\mathcal{L}_{cGAN}(G) = y_0[\log(1-D(I,G(I))]+\lambda y_0[l-G(I)], \quad (8)$$

where λ is the weight given to pixel-wise similarity, in our settings its value is 100.

For our problem, we adopt a cGAN architecture similar to [16] due to its ability of the image-to-image transformation. The architecture of deblurring-cGAN is given in Table I. Unlike DCT-GAN that uses MLP networks, deblurring-cGAN uses multiple convolutional and deconvolutional layers that perform downsampling and upsampling, respectively.

The operation of convolutional and deconvolutional layers are shown in Fig. 7. Convolutional layer (see Fig. 7(a)) gets an input image and a matrix of weights (so called kernel) swipes through the input image. For every window (same size with the kernel) that kernel swipes on the input image, the weighted sum of window and kernel is calculated and saved as one pixel of the output image of the convolutional layer. The deconvolutional layer also calculates the output using a kernel. As shown in Fig. 7(b), one pixel of the input image is multiplied to all values in the kernel and saved on the window of the output image. Pixel values are accumulated for overlapping pixels of output image while kernel swipes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

TABLE I
ARCHITECTURE OF DEBLURING-CGAN

Generator				Discriminator	
Encoder		Decoder			
Layer	Output size	Layer	Output size	Layer	Output size
Input	$512 \times 512 \times 1$	Deconv.	$2 \times 2 \times 1024$	Input	$512 \times 512 \times 2$
Conv.	$256 \times 256 \times 64$	Deconv.	$4 \times 4 \times 1024$	Conv.	$256 \times 256 \times 64$
Conv.	$128 \times 128 \times 128$	Deconv.	$8 \times 8 \times 1024$	Conv.	$128 \times 128 \times 128$
Conv.	$64 \times 64 \times 256$	Deconv.	$16 \times 16 \times 1024$	Conv.	$64 \times 64 \times 256$
Conv.	$32 \times 32 \times 512$	Deconv.	$32 \times 32 \times 1024$	Conv.	$32 \times 32 \times 512$
Conv.	$16 \times 16 \times 512$	Deconv.	$64 \times 64 \times 512$	Conv.	$16 \times 16 \times 512$
Conv.	$8 \times 8 \times 512$	Deconv.	$128 \times 128 \times 256$	Conv.	$16 \times 16 \times 1$
Conv.	$4 \times 4 \times 512$	Deconv.	$256 \times 256 \times 128$	Flatten	256
Conv.	$2 \times 2 \times 512$	Deconv.	$512 \times 512 \times 1$	Sigmoid	1
Conv.	$1 \times 1 \times 512$				

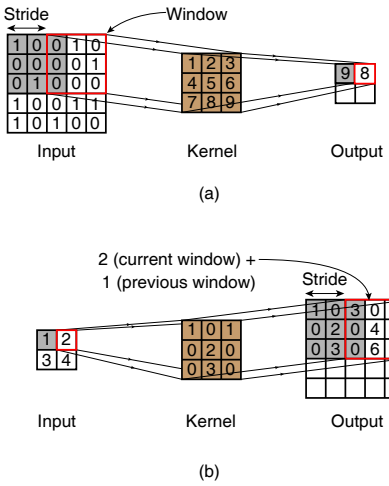

Fig. 7. Operation of (a) convolutional layer and (b) deconvolutional layer.

through input image. The stride of the swiping window on the input of the convolutional layer and output of the deconvolutional layer governs the amount of downsampling and upsampling, respectively. The generator consists of an encoder and decoder network (see Fig. 6(b)). The encoder is a series of convolutional layers that downsamples the input image. The goal of downsampling is to extract only useful features from the input. After the input image is downsampled to a vector, the encoding process is reversed to the decoding process. Decoder consists of deconvolutional layers that upsample the encoder generated features. To make sure that the layers in the decoder can access the lost information in encoding layers, skip connections are used between encoder and decoder. With these connections output of $(n-i)_{th}$ convolutional layer is concatenated with the output of the i_{th} deconvolutional layer, where $1 \leq i < n$ and n is the number of convolutional layers. The window size and stride of all layers are 3×3 and 2, respectively.

The architecture of the discriminator consists of six convolutional layers. The window size and stride of these layers are also 3×3 and 2, respectively. The final prediction is generated by flattening the output of the last convolutional layer and then applying sigmoid ($f(x) = \frac{1}{1+e^{-x}}$) to the sum of all values. For sigmoid values greater than 0.5 discriminator outputs 1, and 0 otherwise.

To decide the weights of convolution and deconvolution kernels a similar three step training procedure is followed as mentioned in Section III-A, while considering the objectives of the discriminator and generator of deblurring-cGAN. Just like DCT-GAN, the discriminator is used only for training the generator. After the training process, the output of the generator becomes the output of deblurring-cGAN as shown in Fig. 6(b).

C. Layout Processing

The output of cGAN is sharpened but is not still a true layout. Besides design rule violations three other main issues are observed: polygons are not perfectly filled with some white pixels inside, some polygon edges are bumpy due to extra black pixels along the edges, and some small black pixels are randomly located in empty space of layout. The layout processing step is completely application dependent. Since in this work our focus is to synthesize layouts for lithography applications so we only perform refinements that make the layouts geometrically feasible.

To remove bumps along the edges on the polygons and randomly located black pixels, we find rectangles of black pixels and only keep those with both dimensions greater than 10nm. As random black spots between polygons and bumps on the edges have very small heights and widths, so they get dropped in the preceding step. To remove small gaps inside and between polygons, we fill the spaces between rectangles that are less than minimum space, which we suppose as 50nm. As the last processing step, we check the minimum width and height of rectangles. We drop rectangles that have height or width less than minimum segment length required for MB-OPC, which is 25nm in our experiments. For every remaining rectangle with any dimension smaller than 50nm, we increase that dimension to 50nm, if this increase does not violate minimum space requirement, otherwise, we drop that rectangle. With these steps, we make sure that all polygons are Manhattan, and the minimum space between polygons and the minimum dimension of a polygon are greater than 50nm. We then convert the processed images into GDS format. The qualitative result of layout processing on Fig. 1(b) is shown in Fig. 1(c).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

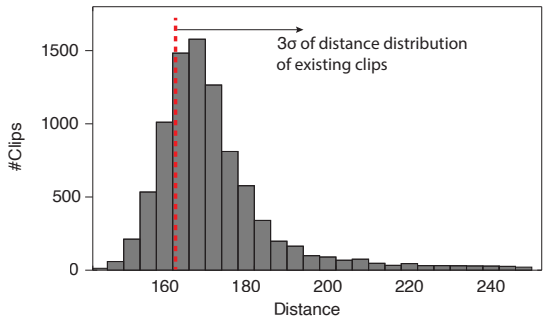

Fig. 8. Distribution of distance of synthetic clips to nearest existing clips.

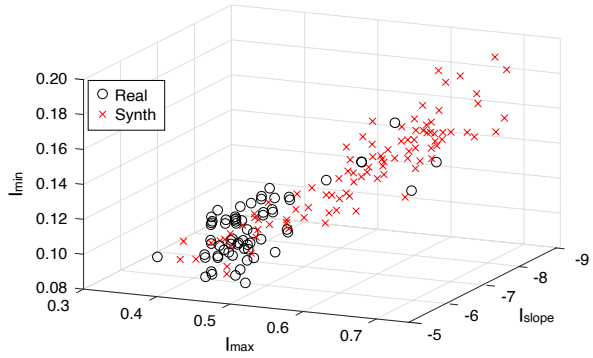

Fig. 9. IPS coverage plot, for real patterns and synthesized patterns after OPC, in I_{max}, I_{min}, and I_{slope}.

IV. Experimental Results

The DCT-GAN and deblurring-cGAN are implemented in Python using Keras [17] with TensorFlow [18] backend. MATLAB is used to perform DCT, IDCT, and layout processing steps. GDS format of layout image is obtained by using Python. Adam optimizer [19] is used to train both GANs, while the learning rate is set to 0.0002. Training takes 7 minutes for DCT-GAN and 2 hours for deblurring-cGAN.

A. Layout Synthesis

A sample M1 layout in 28nm commercial logic technology is taken. It is divided into 10K clips with clip size of 2.5um × 2.5um. Each clip is represented in a pixel image with 512×512 pixels.

1) DCT Signal Generation: The goal of DCT-GAN is to yield DCT signals that correspond to a new clip different from any of existing clips, while it can still be considered as a valid clip. Somewhat conflicting requirement of uniqueness and similarity is experimentally verified.

For each of 10K existing clips (which are from actual layout), its nearest clip in DCT signal space is identified and the distance is recorded. The average distance turns out to be 132.7 and a standard deviation (or sigma) is 9.8, thus three-sigma is located at 162.1. We then run DCT-GAN 10K times to get the DCT signals for 10K new clips (these are synthetic clips). For each of synthetic clips, its nearest existing clip is identified and the distance is measured. The distribution of such distances is shown in Fig. 8 as a histogram, with three-sigma point at 162.1 being identified. About 71% is beyond three-sigma meaning that the majority of synthetic clips are far away from existing clips.

We now assess whether a synthetic clip can really be considered as a clip rather than a series of random DCT values. For this purpose, we build an MLP classifier that receives 496 DCT signals for one clip at its input and outputs 1 if clip is considered as a real or 0 otherwise. The MLP is trained with 10K existing clips while the output is set to 1, and 10K random clips while the output is set to 0. A random clip contains DCT signals, in which each signal is randomly generated between minimum and maximum values of corresponding signals from existing clips. When synthetic clips are now submitted to the MLP, the output is 1 for 96.9% of time so most of them are considered like existing clips even though they have been synthesized. When we provide random clips to MLP for checking, output is 1 for only 0.8% time so it is assured that MLP has been correctly built for our purpose.

2) Deblurring: As shown in Fig. 5 and presented in Section III, the output of IDCT is blurred clip image because its input is 496 DCT signals corresponding to lower frequency region. The goal of deblurring-cGAN is to sharpen such blurred clip image.

We assess cGAN for its capability of reconstructing the input image. We pick 1000 sample existing clips. For each clip, DCT signals are obtained and provided to IDCT to get blurred version of clip, which is then processed by cGAN. The output of cGAN and original clip are compared pixel by pixel and difference is measured. Only 1.8% pixels are different on average of 1000 clips, while 45.1% pixels are different when cGAN input (which is blurred clip image) is compared against the original.

3) Layout Processing: As shown in Fig. 5, the output of deblurring-cGAN goes through a layout processing step. The goal of this step is to get an image that is more like a valid layout, while the features of original image is mostly preserved. Specifically, the shapes are converted into rectangles, small gaps between some rectangles are filled, and some rectangles with height or width smaller than a certain value are modified or removed (see Section III-C).

To verify that processed images can be considered as valid layout clips, we check the minimum space between polygons and minimum dimension of a polygon using KLayout [20]. Results show that minimum space and minimum dimension are 50nm. The requirement of feature preservation is verified by calculating the distance between the input and output of the layout processing step, in DCT space. The average distance between the input and the output is 25.38, more than five times smaller compares to the average distance between the two closest layout clips, which is 132.7. These results show that layout processing only slightly changes the features of the input image and produces geometrically valid layouts.

4) Coverage Analysis of Synthetic Patterns: IPS (image parameter set) is popular for modeling a layout pattern in a number of lithography applications including resist modeling [21], etch modeling [22], and sub-resolution assist feature printing [23]. IPS consists of maximum intensity (I_{max}), minimum intensity (I_{min}), and maximum intensity slope (I_{slope}), all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2020.2982989, IEEE Transactions on Semiconductor Manufacturing

values extracted from a particular gauge (measurement site) of a layout clip.

We randomly choose 100 out of 10K clips that have been extracted and then perform MB-OPC on these clips. They are shown as circles and labeled Real in Fig. 9. As the plot indicates, a lot of them are localized in IPS space limiting their coverage. We now synthesize 100 clips with our proposed approach and perform MB-OPC on these clips. They are also shown in Fig. 9 with label of Synth. It is visually clear that they are less localized and cover more regions of space. It should be noted that three IPS parameters are not independent, e.g. larger I_{max} is associated with larger I_{min}, so whole IPS space is not full coverage.

To quantitatively analyze the coverage of IPS space, we uniformly divide the IPS space into rectangular cuboids. In this space, I_{max} and I_{min} are limited to $[0, 1]$, and I_{slope} is supposed to be in the range of $[-2, -12]$. Width, height, and depth of a cuboid are 0.0084, 0.003, and 0.108, respectively, decided by the average distance between two closest real layout clips in I_{max}, I_{min} and I_{slope}. To get a quantitative measure, we count the number of cuboids containing at least one layout, specifically

$$c = \sum_{j=1}^{K} 1\{Z \cap RC_j\}, \tag{9}$$

where K is the number of cuboids, RC_j is j_{th} cuboid, Z is set of IPS of layout clips either Real or Synth, and $1\{Z \cap RC_j\}$ is an indicator function which is 0, if the intersection is ϕ, and 1 otherwise. In the case of Real, the c is 52, while it is 91 in the case of Synth. Coverage of Synth is larger than that of Real by 75%, which gives the actual demonstration that even though the extracted patterns used for training are localized, but the synthetic patterns are more diverse and have less redundancy.

B. Application to ML-OPC

ML-OPC is set up while ArF immersion lithography (1.35NA) with a quasar illumination source is assumed. MLP network consists of 30 input nodes (for 30 PFT signals extracted from a layout clip) and 3 hidden layers with 128, 64, 32 nodes in each layer. A test circuit named b22 is taken from ITC99 benchmark [24] and it is synthesized in 28nm technology library while a set of standard cells marked with Train in Table II is assumed. Its M1 layout is taken and 80K layout clips are extracted. A commercial MB-OPC tool [25] is set up with etch retarget option being enabled, the damping factor is set to 0.6 for the first iteration and 0.8 for the remaining iterations, and minimum segment length for layout dissection is set to 25nm. All the extracted clips are provided to MB-OPC tool to get their reference mask bias values after 15 OPC iterations. The 80K clips with their mask bias values are now used to train the MLP network.

To test the accuracy of the MLP network, the same b22 circuit is synthesized but using a slightly different set of standard cells marked with Test in Table II. This is in an effort to extract layout clips that are different from those that have been used to train the network. Similar to training, a circuit is synthesized, M1 layout is taken, and 80K layout clips are extracted. The clips are provided to MLP network and predicted mask bias values are obtained at the output; they are compared to reference values when the same clips are submitted to MB-OPC. RMSE is shown in the second column of Table III. Layout clips are now corrected with the mask bias values from ML-OPC; they go through lithography simulation; simulated contours are now compared to target layout; average EPE is shown in Table III.

TABLE II
STANDARD CELLS USED FOR SYNTHESIZING LAYOUTS

	Standard cells used for synthesis
Train	INV, BUF X1-X3, DFF X1-X4, OR, NOR, IMUX (with various cell sizes)
Test	INV, BUF X4-X16, DFF X1-X4, OR X1-X3, AND, NAND, XOR, AOI, MUX (with various cell sizes)

TABLE III
COMPARISON OF ML-OPC ACCURACY

	Trained with 80K real clips	Trained with 40K synthetic + 40K real clips
RMSE	3.31nm	2.60nm
Average EPE	5.21nm	4.09nm
Standard deviation of EPE	7.83nm	6.27nm

To assess our layout pattern synthesis, a half of 80K clips that have been used to train the first MLP network (real clips) are chosen. We then generate 55K layout clips (synthetic clips). They are compared to real clips in PFT space, the space spanned by 30 PFT signals. If a synthetic clip is too close to some real clip, it is considered to be redundant and is dropped. Finally we take the remaining 40K synthetic clips. They, together with 40K real clips, are used to train the MLP network again. The new network is tested with the same 80K clips, which have been used to test the first network, and the results are shown in the last column of Table III. RMSE, average EPE, and standard deviation of EPE are reduced by 21.4%, 21.3%, and 19.9%, respectively. This illustrates the benefit of synthetic clips when they are systematically used together with real clips in ML-OPC process.

C. Application to Resist Model Calibration

The goal of resist model calibration is to determine a few Gaussians to simulate chemical reaction and the threshold values to determine resist development using some test patterns (see Section II-C). The quality of resist model is assessed through CD comparison between simulated resist patterns and corresponding patterns from actual measurement or from rigorous simulation [26].

We first use 1000 parametric patterns from commercial periodic test pattern (TP) generator [27]. The quality of resist model is shown in Fig. 10 with label of 1000 TP: a white bar is when the same 1000 patterns are used to assess the quality, and a black bar is for randomly chosen 1000 patterns from a circuit b22 when it is synthesized with Test in Table II. Clearly, RMSE becomes higher for patterns that have not been used for resist model calibration.

We build a new resist model, this time using two sets of randomly selected patterns, 500 from TP patterns and 500 patterns from b22 when it is synthesized with Train in Table II

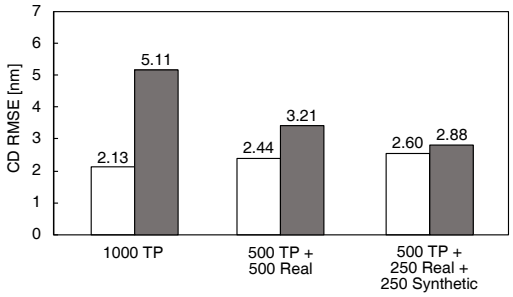

Fig. 10. Resist modeling with three different sets of patterns. White is CD RMSE for modeling patterns and black is for patterns unknown in modeling.

(marked as 500 TP + 500 Real in Fig. 10). Patterns are now more diverse than those in 1000 TP, which explains why RMSE slightly increases in a white bar. However, when the same 1000 patterns from b22 (with Test library) are used to assess in a black bar, RMSE decreases because resist model is less over-fitted than the one with 1000 TP. It is also known that the coverage of resist model increases when some real patterns are included [28].

The final resist model is calibrated with randomly picked 500 and 250 patterns from TP and b22, respectively, and 250 synthetic patterns. The result is shown with the last two bars Fig. 10. The coverage of resist model increases even more with decreased RMSE in black bar. This is expected from larger coverage of synthetic patterns in Fig. 9 as well as from more rigorous machine learning model in ML-OPC experiment.

V. Conclusions

A layout pattern synthesis has been proposed. It relies on two GANs: one to generate new DCT signals and another to get corresponding realistic layout pattern. A key is to train the two GANs such that synthesized patterns are different from actual patterns that have been used to train while they are still valid layouts like actual ones. The coverage of synthetic patterns has been demonstrated in IPS space: it is larger than that of actual patterns by 76%. Pattern diversity is very important for good machine-learning model. When both synthetic and real patterns are applied to build such model for ML-OPC, prediction error in EPE decreases by about 21% compared to the model built using real patterns alone. Resist model calibration has been chosen as another example application. When synthetic patterns are used together with parametric- and real-patterns to calibrate the resist model, CD RMSE decreases by 43.6% compared to resist model built with parametric patterns alone and by 10.3% compared to resist model built with parametric- and real-patterns.

References

[1] S. Shim and Y. Shin, "Topology-oriented pattern extraction and classification for synthesizing lithography test patterns," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 14, no. 1, pp. 1–12 (013 503, Jan. 2015.

[2] T. Matsunawa, S. Maeda, H. Ichikawa, S. Nojima, S. Tanaka, S. Mimotogi, H. Nosato, H. Sakanashi, M. Murakawa, and E. Takahashi, "Generator of predictive verification pattern using vision system based on higher-order local autocorrelation," in *Proc. SPIE Advanced Lithography*, vol. 8326, Mar. 2012, pp. 1–8 (832 615).

[3] W. Ye, Y. Lin, M. Li, Q. Liu, and D. Z. Pan, "LithoROC: lithography hotspot detection with explicit ROC optimization," in *Proc. Asia and South Pacific Design Automation Conference*, Jan. 2019, pp. 292–298.

[4] 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 Advanced Lithography*, vol. 6607, May 2007, pp. 1–11 (66 070M).

[5] A. Hamouda, M. Bahnas, D. Schumacher, I. Graur, A. Chen, K. Madkour, H. Ali, J. Meiring, N. Lafferty, and C. McGinty, "Enhanced OPC recipe coverage and early hotspot detection through automated layout generation and analysis," in *Proc. SPIE Advanced Lithography*, vol. 10147, Feb. 2017, pp. 1–9 (101 470R).

[6] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "DeePattern: Layout pattern generation with transforming convolutional auto-encoder," in *Proc. Design Automation Conference*, Jun. 2019, pp. 148–153.

[7] A. Gu and A. Zakhor, "Optical proximity correction with linear regression," *IEEE Trans. on Semiconductor Manufacturing*, vol. 21, no. 2, pp. 263–271, May 2008.

[8] E. Teoh, V. Dai, L. Capodieci, Y.-C. Lai, and F. Gennari, "Systematic data mining using a pattern database to accelerate yield ramp," in *Proc. SPIE Advanced Lithography*, vol. 9053, Mar. 2014, pp. 1–13 (905 306).

[9] S. Choi, S. Shim, and Y. Shin, "Neural network classifier-based OPC with imbalanced training data," *IEEE Trans. on CAD*, vol. 38, no. 5, pp. 938–948, May 2019.

[10] H. Gamo, "Chapter 3: Matrix treatment of partial coherence," in *Progress in Optics*. Elsevier, 1964, vol. 3, pp. 187–332.

[11] Y. Pati and T. Kailath, "Phase-shifting masks for microlithography: automated design and mask requirements," *Journal of the Optical Society of America A*, vol. 11, no. 9, pp. 2438–2452, Sep. 1994.

[12] N. B. Cobb, A. Zakhor, and E. A. Miloslavsky, "Mathematical and CAD framework for proximity correction," in *Proc. SPIE Advanced Lithography*, vol. 2726, Jun. 1996, pp. 208–222.

[13] T. A. Brunner and R. A. Ferguson, "Approximate models for resist processing effects," in *Proc. SPIE Advanced Lithography*, vol. 3679, Jun. 1996, pp. 176–182.

[14] J. Randall, K. G. Ronse, T. Marschner, A.-M. Goethals, and M. Ercken, "Variable-threshold resist models for lithography simulation," in *Proc. SPIE Advanced Lithography*, vol. 3679, Jul. 1999, pp. 176–182.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 1125–1134.

[17] F. Chollet, "Keras," https://keras.io, 2015.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G.Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX Symp. on Operating Systems Design and Implementation*, Nov. 2016, pp. 265–283.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs.LG]*, Dec. 2014.

[20] M. Köfferlein, "KLayout–High performance layout viewer and editor," http://www.klayout.de, 2020.

[21] T. Roessler, B. Frankowsky, and O. Toublan, "Improvement of empirical OPC model robustness using full-chip aerial image analysis," in *Proc. SPIE Photomask Technology*, vol. 5256, Dec. 2003, pp. 222–229.

[22] Y. Kim, S. Jung, D. Kwak, V. Liubich, and G. Fenger, "Predictable etch model using machine learning," in *Proc. SPIE Advanced Lithography*, vol. 10961, Mar. 2019, pp. 1–11 (1 096 106).

[23] C.-Y. Hung, L. Zhang, and Q. Liu, "A novel approach for full-chip SRAF printability check," in *Proc. SPIE Advanced Lithography*, vol. 6154, Feb. 2006, pp. 1–7 (615 438).

[24] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, Jul. 2000.

[25] Synopsys, "Proteus," Jun. 2014.

[26] ——, "Sentaurus Lithography," Jun. 2018.

[27] ——, "Proteus Metrokit," jun 2018.

[28] W. A. Tawfic, M. Al-Imam, K. Madkour, R. Fathy, I. Kusnadi, and G. E. Bailey, "Feedback flow to improve model-based OPC calibration test patterns," in *Proc. SPIE Advanced Lithography*, vol. 6521, Mar. 2007, pp. 1–9 (65 211J).