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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

GAN-SRAF: Sub-Resolution Assist Feature Generation using Generative Adversarial Networks

Mohamed Baker Alawieh, *Student Member, IEEE*, Yibo Lin, *Member, IEEE*, Zaiwei Zhang, Meng Li, *Member, IEEE*, Qixing Huang, David Z. Pan, *Fellow, IEEE*

***Abstract*—As the integrated circuits (IC) technology continues to scale, resolution enhancement techniques (RETs) are mandatory to obtain high manufacturing quality and yield. Among various RETs, sub-resolution assist feature (SRAF) generation is a key technique to improve the target pattern quality and lithographic process window. While model-based SRAF insertion techniques have demonstrated high accuracy, they usually suffer from high computational cost. Therefore, more efficient techniques that can achieve high accuracy while reducing runtime are in strong demand. In this work, we leverage the recent advancement in machine learning for image generation to tackle the SRAF insertion problem. In particular, we propose a new SRAF insertion framework, GAN-SRAF, which uses generative adversarial networks (GANs) to generate SRAFs directly for any given layout. Our proposed approach incorporates a novel layout to image encoding using multi-channel heatmaps to preserve the layout information and facilitate layout reconstruction. Our experimental results demonstrate $\sim 14.6\times$ reduction in runtime when compared to the previous best machine learning approach for SRAF generation, and $\sim 144\times$ reduction compared to model-based approach, while achieving comparable quality of results.**

***Index Terms*——DFM, Sub-Resolution Assist Feature, Generative Adversarial Learning, Domain Transfer**

I. Introduction

While the integrated circuits technology node continues to scale, the photolithography techniques are supposed to keep up the pace and cope with the ever shrinking feature size. In fact, low image contrast and complex target pattern shapes make it extremely difficult for low$-k1$ lithography, the mainstream technique, to achieve desired lithographic process windows [2], [3]. Hence, resolution enhancement techniques have been the major strategy to improve lithographic process window.

Among these techniques, Sub-Resolution Assist Feature (SRAF) generation is used to improve the lithographic process window of target patterns. These assist features are not actually printed; instead, the SRAF patterns would deliver light to the positions of target patterns at proper phase which can improve the robustness of target printing to lithographic variations [4]. In practice, the process variation band is typically used as a measure of such robustness, where the goal is to achieve the minimum possible band [4].

In literature, different SRAF generation approaches have been proposed and adopted. On one hand, the rule-based approach can achieve acceptable accuracy within short execution time for simple designs and regular target patterns [2], [5], [6]. However, the rule-based approach cannot handle complex shapes as it requires significant pre-processing engineering efforts. On the other hand, two trends of model-based SRAF generation methods have been proposed and these can be categorized based on the lithographic computations involved. The first trend uses simulated aerial images to seed the SRAF generation [2], [3], while the other applies inverse lithography technology (ILT) and computes the image contour to guide the SRAF generation [7]. Despite better lithographic performance compared to the rule-based approach, the model-based SRAF generation is very time-consuming and it is difficult to achieve the same SRAFs around the same layout configuration, i.e., not consistent [4].

Recently, machine learning has been proposed to tackle the problem of SRAF insertion to reduce the computational cost associated with model-based methods [4], [8]. The proposed method relies on SRAF features extraction with local sampling scheme to obtain the optimal SRAF map. The key idea is to use a 2D grid plane where a classification model is trained to predict the probability of existence of SRAF in each grid based on the extracted features. Although this approach has demonstrated significant speedup compared to model-based approaches while achieving comparable results in terms of process variation band, there is still significant room for improvement as we will show in this paper.

Motivated by the recent advancement in the field of image processing in computer vision [9]–[13], we elect in this work to address the SRAF generation problem from a new perspective. In fact, a layout in its essence can be simply viewed as an image; hence, machine learning techniques developed for image related tasks can come in handy. Moreover, it has been shown that using convolutional neural networks has demonstrated superior runtime and performance compared to local decision based approaches when dealing with visual data. Specifically, Generative Adversarial Networks (GANs) have been leveraged to perform a wide-range of domain transfer tasks where image translation is the most infamous. In other words, given related images in two different domains, models can be trained to translate images from one domain to another [9], [10], [14]. Examples of such applications include, among others, image colorization and aerial to map

This paper was presented in part at the Design Automation Conference in 2019 [1]. This work was supported in part by NSF under award 1718570 and Kioxia Corporation.

M. B. Alawieh, M. Li and D. Z. Pan are with The Department of Electrical and Computer Engineering, The University of Texas at Austin, TX, USA.

Y. Lin is with The Computer Science Department, Peking University, Beijing, China.

Z. Zhang and Q. Huang are with The Computer Science Department, The University of Texas at Austin, TX, USA.

and edge to photo translations [10]. In addition, GANs have been recently adopted to tackle different problems in IC design [15] such as enhancing the optical proximity correction manufacturing [16], end-to-end lithography simulation [17], mask modeling [18] and physical design [19], [20].

In this work we propose to use two GAN schemes to address the SRAF generation task. In the first, we propose to use Conditional Generative Adversarial Network (CGAN) for SRAF generation by casting the problem into an image translation task where the two images domains are: (i) original layout and (ii) layout with SRAFs. Hence, generating an SRAF scheme for a particular layout can be seen as translating the layout image from the first domain (i.e., original layout) to the second domain (i.e., layout with SRAFs). Towards this goal, a set of paired images (i.e. original layout with no SRAF paired with the corresponding layout after SRAF insertion) is provided for the network to learn the desired translation.

While this approach is adequate for cases where paired data is available, we also propose an alternative GAN scheme that handles the case where data is available but is not necessarily paired. In practice, the availability of adequate training datasets is one of the major challenges facing machine learning models. Therefore, and knowing that paired data may not be available especially at the early stages in IC technology nodes, we propose an SRAF insertion scheme featuring an unpaired image-to-image translation. Our proposed model, inspired by the Cycle Generative Adversarial architecture (CyGAN) [14] learns simultaneously a two-way image translation using unpaired data. Unlike the CGAN scheme where paired data is used to learn a one-way translation, CyGAN - as the name implies - uses a double cycle scheme to learn two-way translation using unpaired data. In practice, two parallel translation models are trained where the objective is to reconstruct an image after undergoing two translations: (i) from native domain to the other domain, then (ii) back to the native domain. Such reconstruction will be accurate for both domains when both translation tasks are accurate. Hence, this learning scheme uses a cycle translation to learn the mapping using unpaired images.

Both of the proposed GAN schemes require casting the layout information into image format. Therefore, layout files are mapped into images in a novel encoding scheme that captures the layout details. This scheme incorporates a multi-channel heatmap encoding of different layout objects into different layers of an image [21]–[24]. Additionally, this encoding is accompanied by a fast GPU-accelerated decoding scheme to recover layout schemes from images generated by CGAN and CyGAN. With our proposed encoding/decoding framework, CGAN and CyGAN models are trained to generate layouts with SRAF inserted using a paired and unpaired data set respectively. Once trained, the models can take an original layout image as an input and generate a new image with SRAFs inserted. These generated images can be eventually mapped back to layout files.

In this SRAF generation framework, our main contributions are summarized as follows:

- Generative Adversarial Networks are used for the first time for SRAF generation.
- We cast the SRAF generation problem as an image-to-image translation task where the layout is translated from its *original* domain to *layout with SRAFs* domain.
- A CGAN-based SRAF generation scheme is proposed for the scenario with paired dataset.
- Alternatively, a CyGAN-based SRAF generation scheme is proposed for the case of unparaied dataset.
- A novel multi-channel heatmap encoding/decoding scheme is used to map layouts to images suitable for GAN training while preserving the layout details.
- Our proposed framework achieves $\sim 14.6\times$ speed-up with comparable lithographic performance when compared with state-of-art machine learning based approach and $\sim 144\times$ speed-up over the model-based approach in commercial tool Mentor/Calibre [4] while achieving comparable results.

The remainder of this paper is organized as follows. In Section II we review the technical background and then present the two proposed approaches in Section III. Section IV presents numerical results demonstrating the efficacy of our method, and conclusions are presented in Section V.

II. Problem Formulation

The objective of the SRAF generation framework is to insert SRAFs on any given layout in a manner that mimics the SRAF scheme generated from model-based techniques. Practically, the input is a layout clip with target patterns only as shown in Fig. 1a, and the expected output is a new layout clip similar to the one shown in Fig. 1b where SRAFs are generated to aid the printing of target patterns. In other words, the objective is to train a GAN to translate images from the target domain, D_{Trgt}, (Fig. 1a) to the SRAF domain, D_{SRAF}, (Fig. 1b).

In the CGAN training phase, each training sample consists of a pair of images representing the original layout in D_{Trgt} and its corresponding layout in D_{SRAF}. Based on the training data, the CGAN model is trained to map images from D_{Trgt} to D_{SRAF}. On the other hand, in the CyGAN training phase, each sample consists of two unpaired images in the two domains and the CyGAN model is trained to perform two-ways mapping; i.e., from D_{Trgt} to D_{SRAF} and from D_{SRAF} to D_{Trgt}. Then, the trained model can be used to generate SRAFs from layouts with target patterns. However, two challenges should be addressed here. The first is that proper image encoding/decoding is needed to aid the GAN training scheme. Secondly, the generated SRAF scheme may violate some of the manufacturing rules; hence, a post-processing step is needed to generate a final layout with SRAFs while abiding by the specified rules.

To evaluate our proposed SRAF generation method, we use two metrics to assess the performance of the mask optimization results: (i) process variation (PV) band and (ii) edge placement error (EPE). These metrics are defined in a way analogous to the definitions used in [4].

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

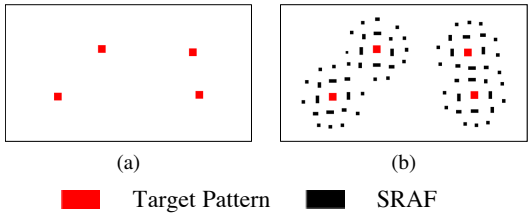


Fig. 1: SRAF generation task can be cast as an image translation problem where layout with target contacts (a) are translated to ones with SRAF generated (b) .

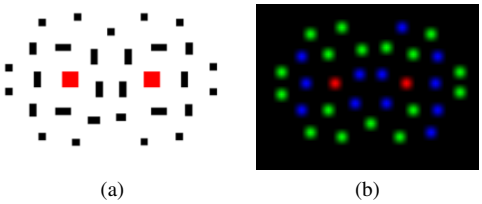


Fig. 2: The multi-channel heatmaps encoding process is demonstrated where (a) shows an original layout representation and (b) shows the encoded representation.

III. SRAF Insertion Using Generative Adversarial Networks

In this section we present our proposed SRAF generation approach using generative adversarial networks. As a first step, the data preparations process, which is a key-enabler for the reformulation of SRAF generation as an image translation task, is introduced. Next, we introduce the two GAN schemes for SRAF generation: (i) CGAN scheme for paired datasets, and (ii) CyGAN scheme for unpaired datasets. Then, a fast GPU-accelerated decoding scheme to map images back to layout files is introduced.

A. Data Preparation using Heatmap Encoding

As shown in Fig 1, the layouts from both domains D_{Trgt} and D_{SRAF} can be treated directly as images. However, this direct image representation is not suitable for the SRAF generation using GANs because the expected output cannot be directly mapped to layout files due to two major limitations. First, the trained GAN is not guaranteed to generate 'clean' rectangular shapes for the SRAFs. In practice, images generated from GANs tend to be blurry and GANs exhibit inherent limitation in detecting sharp edges [13]. In addition, and even under the assumption that the GAN models can generate sharp-edged rectangles for the SRAFs, extracting the SRAF information from the image to be mapped back to the layout file can be prohibitively expensive. Such mapping requires obtaining both SRAF locations and sizes from the image generated by the model. Hence, the direct image representation similar to that shown is Fig. 1 is ill-equipped for SRAF generation using GANs.

With this in mind, we propose using a special encoding scheme, typically used in keypoint estimation [21]–[24], that can overcome the aforementioned limitations. The proposed scheme is based on multi-channel heatmaps which associates each object type with one channel in the image [23], [24]. Specifically, a multi-channel image is a simple representation where the number of channels is equal to that of the object types in the problem. On each particular channel, the first step is to obtain the locations of its corresponding objects in the original image. Next, a Gaussian noise circle is centered at the obtained locations on the channel [23], [24].

To elaborate on this, we consider the example shown in Fig. 2 where an original layout is shown in Fig. 2a and the multi-channel heatmap representation is shown in Fig. 2b. In this example, we limit the number of channels to 3 to visualize the encoded representation through a red-green-blue (RGB) image. These three types are : (i) target patterns (in red), (ii) horizontal SRAFs (in green) and (ii) vertical SRAFs (in blue). Similar encoding can be done for images in D_{Trgt} where only one non-empty channel contains the target patterns.

The representation shown in Fig. 2 has two main advantages. First, learning sharp edges, which is a hard task in GANs, is not needed. Instead, training-friendly Guassian objects are used to encode the objects in the original image. Secondly, and most importantly, with this representation the images generated by the GAN models can be easily mapped back to layout files. In practice, since each channel represents a well-defined type of SRAFs, it suffices to detect the location of excitations in the channel to get the locations of SRAFs of this particular type in the layout file.

Therefore, prior to the model training step, images in the training set are all encoded into multi-channel heatmap representation. In such mapping, the number of channels is equal to $M + 1$ where M is the number of SRAF types and one additional channel is used to encode the target patterns. Here, an SRAF type is simply defined as containing SRAFs with specific dimensions (e.g. [0.04,0.09]).

B. CGAN Model

In their essence, GANs were proposed as generative models that learn a mapping from a random noise vector z to output image y, $G : z \rightarrow y$ [12]. Later, different versions of GANs, tailored towards specific domains and applications, were proposed. Among those are the CGANs which, in contrast with original GANs, learn a mapping from an observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. As in the case of most GAN structure, the architecture of a CGAN is composed of two main components: the *generator* and the *discriminator*. The generator G is trained to produce images in D_{SRAF}, based on an input image in D_{Trgt}, that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator "fakes". The overview of the training procedure is described in Fig. 3 [9], [10]. In this work we adopt the CGAN structure proposed in [10] for the image translation task.

Mathematically, the loss function used for training the CGAN can be given as [9], [10]:

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

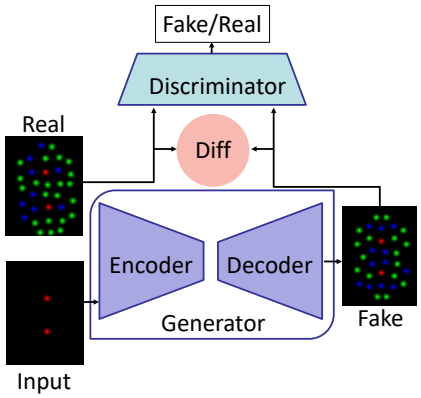


Fig. 3: An overview of the CGAN functionality is shown.

$$\mathcal{L}_{CGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))] + \lambda_{L1}\mathbb{E}_{x,z,y}[||y - G(x,z)||_1]. \quad (1)$$

In (1), the first two terms represent the traditional GAN loss function where G tries to minimize the objective against an adversarial D that tries to maximize it [9], [10], [12]. The third term in the equation affects only the generator whose objective is, not only to fool the discriminator, but also to generate images close to the ground truth. Here, L_1-norm is used because it encourages less blurring when compared to L_2-norm [10].

Hence, the optimal Generator can be obtained by solving for the following objective:

$$G^* = \arg\min_G \max_D \mathcal{L}_{CGAN}(G,D). \quad (2)$$

In practice, experiments shown in [10], [25] have demonstrated that the noise z is typically ignored by the generator. Hence, noise is instead introduced through dropout on several layers of the generator in both the training and inference stages.

In the next subsections, the details of both the generator and discriminator used in image to image translation CGAN are shown in addition to the training and inference process. These implementations are adapted from the deep convolutional generative adversarial networks framework proposed in [26].

1) Generator: The conventional generator in a GAN is basically an encoder-decoder scheme where the input is passed through a series of layers that progressively downsample the input (i.e, encoding), until a bottleneck layer, at which point the process is reversed (i.e, decoding) [9], [10], [12], [26]. This process is shown in Fig. 2a where the gray (white) layers form the encoder (decoder) respectively.

For image translation tasks using CGAN, a significant amount of information is shared between the input and the output, and it would be desirable to shuttle this information directly across the net without passing through the bottleneck layer. Towards this goal, skip connections are added following the general shape of a "U-Net" [10], [27]. As shown in Fig. 2a, skip connections are added between each layer i and layer $L-i$, by simply concatenating all channels at layer i with those at layer $L-i$, where L is the total number of layers.

Layer	Output Dimension	Channels
Input	256x256	3
Conv1	128x128	64
Conv2	64x64	128
Conv3	32x32	256
Conv4	16x16	512
Conv5	8x8	512
Conv6	4x4	512
Conv7	2x2	512
Conv8	1x1	512

TABLE I: The details of the encoder network in the generator are presented.

Layer	Output Dimension	Channels
Input	256x256	6
Conv1	128x128	64
Conv2	64x64	128
Conv3	32x32	256
Conv4	16x16	512
FC	1	1

TABLE II: The details of the discriminator network are presented.

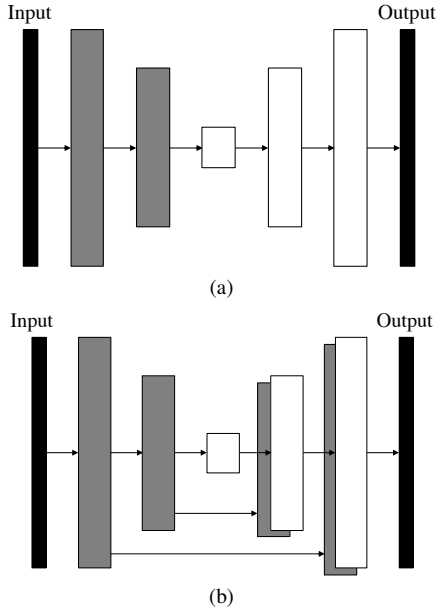


Fig. 4: Two generator network schemes are shown: (a) shows the conventional approach while (b) shows the Unet scheme with skips [10], [27]

Table I lists the characteristics of the layers in the encoder [10]. In all convolutions layers (Conv1-8), (5×5) filters are used with stride 2, and leaky relu is used as an activation function [9], [10], [26]. On the other hand, the decoder is simply a mirrored image of the encoder with deconvolutional layers replacing the convolutional layers. Besides, the downsampling process of the encoder is demonstrated in the network architecture in Fig. 5 where each block shows the tensor properties after undergoing convolution and activation. The notation ch @ $d \times d$ corresponds to tensor with ch channels each with dimensions $[d, d]$. Moreover, the tuple (c, s) above each arrow shows the transition operation corresponding to a convolutional operation with filter $(c \times c)$ and a stride s. On the other hand, the decoder has a mirrored architecture with convolutions replaced by deconvolution at each layer.

2) Discriminator: Practically, the discriminator is a convolutional neural network whose objective is to classify "fake" and "real" images. Hence, its structure differs from that of

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

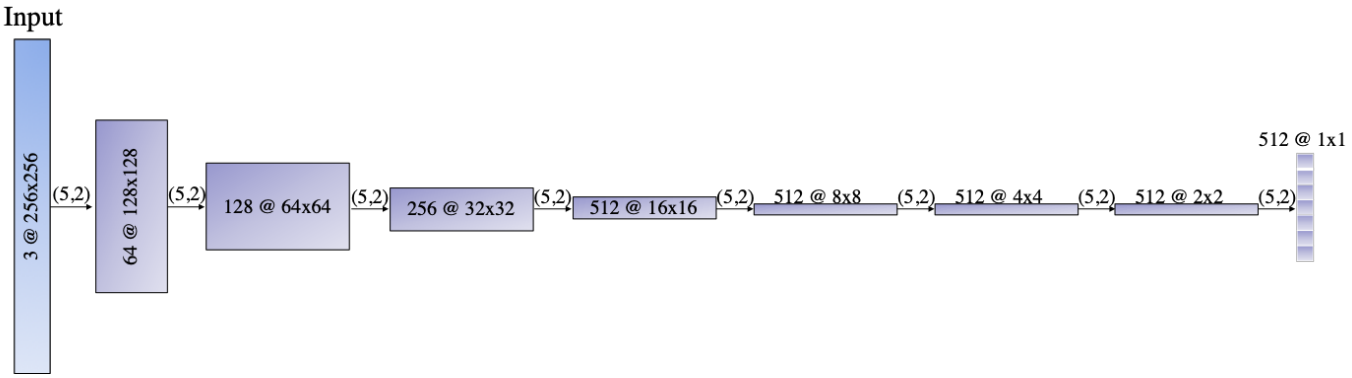


Fig. 5: The network architecture for the encoder of the CGAN model is shown. Each block, denoted with ch @ $d \times d$ represents a tensor with ch channels with dimensions $[d, d]$. Arrows represent the convolution-activation operation with (c, s) denoting the size of the convolutional filter and the stride.

the generator. Table II summarizes the different layers of the discriminator which constitutes of 4 convolutional layers (Conv1-4) and one fully connected layer (FC) whose output is the binary classification results [10]. Similar to the generator, all convolutional filters are of size (5×5) with stride 2.

3) CGAN Training and Inference: Training a CGAN model follows the typical procedure used for training GAN with mini-batch Stochastic Gradient Descent (SGD) and Adam solver [12], [28]. The training alternates between one gradient descent step on the discriminator, and then one step on the generator. During inference, the generator net is used in the same manner as during the training phase. Indeed, the dropout steps introduced in the layers of the decoder are also used during the inference time. Moreover, batch normalization is applied using the statistics of the test batch where a batch size of 4 is used for the SRAF generation task.

C. CyGAN Model

In some scenarios, and especially at the early stages of new technologies, no enough paired samples are available to train a CGAN model. Hence, we propose using the CyGAN approach that does not require paired samples for the training process. Unlike the CGAN model that learns a one-way translation task, the CyGAN model learns a two-way mapping between the two domains [14]. As illustrated in Fig. 6, our model includes two mappings $G_x : y \rightarrow x$ and $G_y : x \rightarrow y$. In addition, we introduce two adversarial discriminators Dis_x and Dis_y , where Dis_x aims to distinguish between images in $\{x\}$ and translated images $G_x(D_x)$; in the same way, Dis_y aims to discriminate between $\{y\}$ and $G_y(D_x)$. This scheme, shown in Fig. 6, can be viewed as training two auto-encoders [14], [29]: we learn one auto-encoder $G_x \circ G_y : x \rightarrow x$ jointly with another $G_y \circ G_x : y \rightarrow y$. However, these auto-encoders each have special internal structures: they map an image to itself via an intermediate representation that is a translation of the image into another domain [14]. In the training process, the objective contains two types of terms: adversarial losses for matching the distribution of generated images to the data distribution in the target domain; and cycle consistency losses to prevent the learned mappings G_x and G_y from contradicting each other [14]. In Fig. 6, the two translation cycles are highlighted in two different colors: the green arrows represent the path $D_{SRAF} \rightarrow D_{Trgt} \rightarrow D_{SRAF}$, and the red arrows represent the other path $D_{Trgt} \rightarrow D_{SRAF} \rightarrow D_{Trgt}$.

Mathematically, the loss adversarial for a single mapping function in CyGAN is similar to that for the CGAN and can be given as [14]:

$$\begin{aligned} \mathcal{L}_{GAN}(G_y, Dis_y) = \mathbb{E}_y[\log Dis_y(y)] \\ + \mathbb{E}_x[\log(1 - Dis_y(G_y(x)))] \end{aligned} \tag{3}$$

A similar loss function is considered for the mapping function $x \rightarrow x$ with G_x and Dis_x [14].

In theory, such an adversarial loss helps in learning the mappings G_x and G_y that produce outputs according to the distributions of x and y respectively. However, adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To address this, the learned mapping functions should be *cycle-consistent* as shown in Fig. 6. This means that for each image x from, the image translation cycle should be able to bring x back to the original image, i.e., $x \rightarrow G_y(x) \rightarrow G_x(G_y(x)) \approx x$. Similarly, for each image y , G_x and G_y should also satisfy backward cycle consistency: $y \rightarrow G_x(y) \rightarrow G_y(G_x(y)) \approx y$. Mathematically, this can be expressed as [14]:

$$\begin{aligned} \mathcal{L}_{cyc}(G_y, G_x) = \mathbb{E}_x[||x - G_x(G_y(x))||_1] \\ + \mathbb{E}_y[||y - G_y(G_x(y))||_1] \end{aligned} . \tag{4}$$

Ultimately, the two loss terms in eq. (3) and (4) are combined to train the CyGAN model. In the next subsections, the details of both the generator and discriminator used in image to image translation CyGAN are shown [14].

1) Generator: The core architecture for the CyGAN generator is similar to that of a CGAN in the sense that it comprises an encoder-decoder scheme. Fig. 7 shows the details of the generator architecture where notation follows that used in 5. Beside the encoder and decoder, a set of 9 transformer blocks are used at the core of the domain transfer blocks [14], [30]. These 9 blocks are Resnet blocks where a residue of input is added to the output in each of them [31]. This is done

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

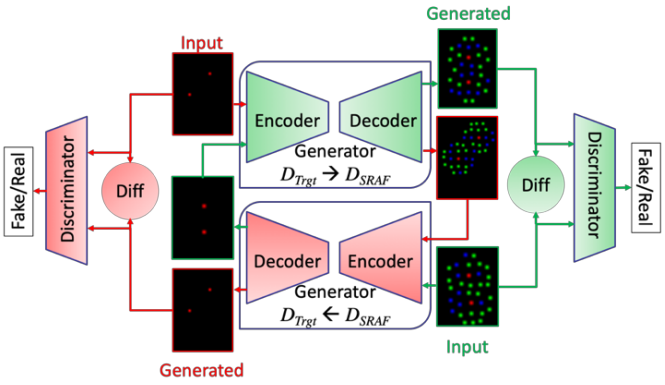


Fig. 6: An overview of the CGAN functionality is shown.

Layer	Output Dimension	Channels
Input	256x256	6
Conv1	128x128	64
Conv2	64x64	128
Conv3	32x32	256
Conv4	16x16	512
Conv5	16x16	1

TABLE III: The details of the discriminator network for the CyGAN model are presented.

to ensure that the properties of input of previous layers are available for later layers as well, so that the their output do not deviate much from original input, otherwise the characteristics of original images will not be retained in the output and results will be very abrupt [14]. Since the primary aim of the task is to retain the native characteristic of original input like the size and location of the objects, residual networks are a great fit for these kind of transformations [14].

2) Discriminator: The discriminartor in the CyGAN model has the same task as that in the CGAN; its objective is to classify "fake" and "real" images. In this model, a 70×70 PatchGAN to classify whether 70×70 images patches are "real" or "fake" [32]. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator and can work on arbitrarily-sized images in a fully convolutional fashion [32]. Table III summarizes the different layers of the discriminator. which constitutes of 5 convolutional layers (Conv1-4) whose output is a 16×16 tensor [14]. The final decision made by the discriminator is the average of all elements in the output tensor. All convolutional filters are of size (4×4) with stride 2, except for the last layer which has a stride of 1.

3) CyGAN Training and Inference: The training process for the CyGAN is similar to that of the CGAN model summarized in Section III-B3.

D. Results Decoding: Heatmap to Layout

The output of the GAN for SRAF generation is a layout image in a multi-channel heatmap representation as in the example of Fig. 2b. Hence, a decoding step is required to extract the SRAF information from the encoded image. Here, it is important to note that the choice of the encoding scheme was made with this SRAF extraction task in mind. The objective of this step is to extract both the types (i.e, sizes) and locations of the SRAF to be generated on the layout. As described in Section III-A, the SRAF scheme is encoded such that each channel of the image contains one type of SRAFs. Hence, for each channel, it suffices to get the locations of excitations on the heatmap to get locations of the SRAFs of a particular type. Therefore, the task reduces to detecting the excitations on each channel.

Towards the goal of parsing the heatmap, we start from conventional methods used in parsing heatmaps in the field of keypoint estimation [21], then tailor these methods to the SRAF generation task at hand. Knowing that the encoding scheme uses a Gaussian circle centered at each keypoint location, it is expected that the exact SRAF location possess the highest magnitude compared to its neighbors. Therefore, on each channel, a fixed size window is swept over all non-zero pixels on the map where the value of the center pixel is compared to that of all other pixels in the window. The center pixel is considered an SRAF location only if it possess the highest magnitude among all pixels in the window. This step constitutes the core of the decoding procedure. In addition, two filtering stages are used to reduce the effect of noise and false alarms.

In the first step, and before performing the window sweeping operation, the image generated from the GAN is passed through a filtering stage with a fixed threshold that sets all pixels with values below the threshold to zero. This step helps reduce the effect of noise present in the generated image. On the other hand, a checking step accompanies the core window screening step mentioned above to discard *isolated* pixels. In other words, if a single pixel in a window has a non-zero value, it is more likely a noise pixel than a legit SRAF location. This is mainly because SRAF locations are expected to be encoded through a Gaussian circle, not a single pixel excitation. When the center pixel in a window possess the highest value compared to other pixels, the number of its immediate non-zero neighbors is examined. If a majority of the neighboring pixels are non-zeros, the pixel is considered a legitimate SRAF; otherwise, the pixel is considered an *isolated* pixel and hence a false alarm. As an illustration, a synthetic example is shown in Figure 8. Figure 8a shows a synthetic heatmap example similar to those produced by the GAN models. In the figure, the scale goes from white to red in an increasing order. Figure 8b shows the results of the thresholding step. The search for excitations in the map results in two candidate locations as shown in Figure 8c; however, only one of the two locations is a valid SRAF location as shown in Figure 8d.

The decoding scheme is summarized in Algorithm 1. The input is a multi-channel heatmap encoded image I generated from the GAN model with the channel carrying the target pattern dropped. Algorithm 1 also requires the size of the scanning window as an input in addition to the two threshold values ϵ_1 and ϵ_2 that are used for the filtering stage and the neighborhood check respectively. As a first step, thresholding is done using ϵ_1 (line 2). Next, for each channel in the image, all pixels are scanned, and at each location three conditions

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

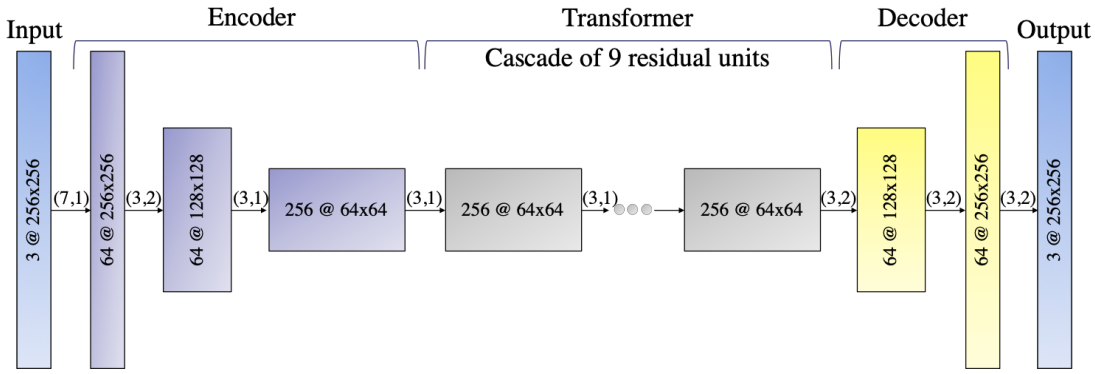


Fig. 7: The network architecture for the generator of the CyGAN model is shown. Each block, denoted with ch @ $d \times d$ represents a tensor with ch channels with dimensions $[d, d]$. Arrows represent the convolution-activation operation with (c, s) denoting the size of the convolutional filter and the stride.

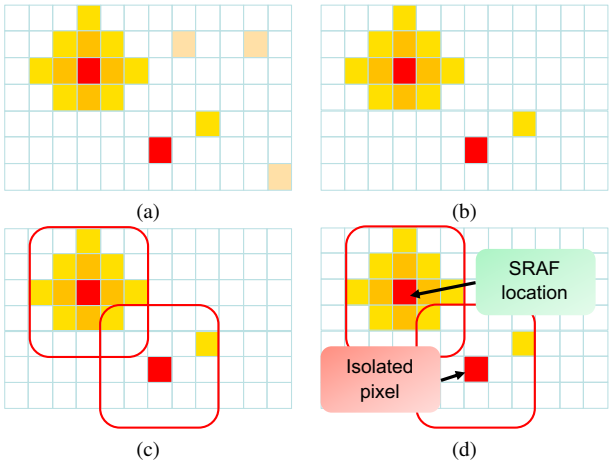


Fig. 8: A synthetic example showing the heatmap decoding process is shown. (a) shows an original heatmap, (b) shows the results after applying thresholding and representation. The map screening results in two tentative SRAF positions as shown in (c). Finally, by checking the neighborhood only one of the locations is considered a valid SRAF location, while the other is discarded.

are checked: (i) if the value of the pixel is nonzero, (ii) if the number of its non-zero (nnz) neighbors is greater than ϵ_2 (line 6) and if its value is the maximum in the window (line 7). If all three conditions are satisfied at a particular (i, j, m) location, the coordinates (i, j) are added to Ω_m. The Algorithm returns the sets $\{\Omega_m : m = 1, \ldots, M\}$, where each set Ω_m contains the locations of SRAFs of type m. These sets contain all necessary information to generate a layout clip similar to the one in Fig. 2a. In practice, Algorithm 1 can be significantly accelerated with massive parallelization, since each pixel can be checked independently. Therefore, we develop a custom CUDA accelerator for this process and integrate it to GAN-SRAF.

Algorithm 1 GAN Results Decoding

Require: An image I with dimensions $(N \times N \times M)$, a widow size w, ϵ_1 and ϵ_2

1: Initialize $\{\Omega_m \leftarrow \emptyset : m = 1, 2, \ldots, M\}$
2: Set all pixels less than ϵ_1 in I to 0
3: **for** $m = 1, 2, \ldots, M$ **do**
4: **for** $i = w, w + 1, \ldots, N - w$ **do**
5: **for** $j = w, w + 1, \ldots, N - w$ **do**
6: **if** $I_{i,j,m} > 0$ and $\text{nnz}(I_{i\mp1,j\mp1,m}) > \epsilon_2$ **then**
7: **if** $I_{i,j,m} = \max I_{i\mp w,j\mp w,m}$ **then**
8: $\Omega_m \leftarrow \Omega_m \cup (i, j)$
9: **end if**
10: **end if**
11: **end for**
12: **end for**
13: **end for**
14: **return** $\{\Omega_m : m = 1, 2, \ldots, M\}$.

E. Post Processing

The decoding procedure in Algorithm 1 generates a properly formatted layout clip. However, this clip is not guaranteed to follow all SRAF manufacturing rules such as minimum spacing [4]. Hence, a final legalization step is employed to ensure that all design rules are satisfied. In this work, we adopt the same greedy simplification scheme proposed in [4] to accommodate the mask manufacturing rules. The key idea is to shrink rectangular SRAF shapes to abide by the rules.

This post processing step is applied to the layout obtained from Algorithm 1 to arrive at the final rules-abiding layout. The step is applied to both SRAF insertion schemes using CGAN and CyGAN.

IV. Experimental Results

In this section, we show the experimental results that demonstrate the efficacy of our proposed SRAF generation approach using generative adversarial learning. First, data preparation using heatmap encoding along with model training are presented. Then, a comparison with state-of-the-art

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

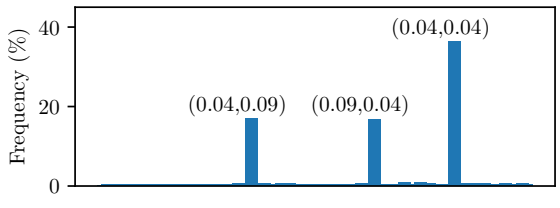


Fig. 9: The distribution of SRAF sizes in the dataset is shown.

approaches is shown which demonstrates the superior performance of out proposed GAN-based approach. The proposed framework for SRAF generation is implemented in Python with the TensorFlow library [33] and validated on a Linux server with 3.3GHz Intel i9 CPU and Nvidia TITAN Xp GPU.

A. Training Data Description and Models Training

To train both the CGAN and CyGAN models, a training dataset containing 1620 layout clips is obtained from a setup that corresponds to memory design and is used for contact generation. In practice, this setup matches closely intel's 14nm process (for P1272-CPU) with minimum spacing and minimum width for contacts set to 70nm (pitch=140nm). The area of the layouts ranges between 0.5 and 10 um^2, and the number of contacts per layout is between 1 and 25. From each clip, two 256×256 images are created using the multi-channel heatmap encoding presented in Section III-A. The first image is in D_T, where only target patterns are present, while the second is in D_S where both target patterns and SRAFs are present. When mapping the layouts to images, all layout clips in the dataset, irrespective of their area, are mapped using the same scale. In other words, the largest clip is scaled to a 256×256 image using a scaling factor which is used universally for all other clips. With such scaling, smaller clips will occupy a portion of the 256×256 image. This is done to ensure that contacts and SRAFs are seen by the model at a uniform scale.

The SRAFs in the layout clips are generated using model-based SRAF generation in Mentor-Calibre with the same setup as [4] and are considered the golden solution when training the GAN models. For the CGAN model, the images are used in pairs while they are randomly used without pairing for the CyGAN setup.

One critical factor in the encoding process is the number of SRAF types to consider since this will decide upon the total number of channels in the encoded images. Here, an SRAF type is defined by the SRAF polygon shape as a tuple of two dimensions. In other words, each unique SRAF shape can be regarded as one type. The encoding scheme in Section III-A depicts that each SRAF type is encoded on one channel of the image; hence, there exists a trade-off between the number of SRAF types to consider and the image size. In our experiments, it was clear that a few SRAF shapes have significantly higher frequency than the remaining ones as shown in Fig. 9 which presents the distribution of SRAF shapes in the training dataset.

Parameter	Learning rate	β_{ADAM}	# of epochs	Batch size	λ_{L1}
value	0.0002	0.5	100	4	100

TABLE IV: A summary of GAN training parameters is shown.

In fact, SRAFs with sizes $(0.04, 0.09)$, $(0.09, 0.04)$, and $(0.04, 0.04)$ are evidently dominant. Therefore, assigning a separate channel to encode each of the total 101 unique shapes will be too expensive and of small return. Hence, it makes more sense to map all shapes to a small set of categories based on the most dominant ones. By examining the distribution in Fig. 9, three major categories can be intuitively obtained based on the three dominant shapes.

However, keeping in mind that the target contact will occupy one channel in the image, we desire to restrict the SRAF types to two for the evident reason that a 3-channel image can be easily visualized. In fact, we observe that the $(0.04, 0.04)$ SRAF shapes were originally either $(0.04, 0.09)$ or $(0.09, 0.04)$; however, due to the legalization step in the post processing phase, these original objects were pruned to the size of $(0.04, 0.04)$ to satisfy mask constraints. Moreover, we observe that even with only $(0.04, 0.09)$ and $(0.09, 0.04)$ SRAF shapes, the final SRAF solutions after post-processing could still contain SRAFs of size $(0.04, 0.04)$ to satisfy the mask constraints. Hence, we choose to map SRAFs in the layout clips to two types (shapes) only: (i) $(0.09, 0.04)$ denoted as horizontal and (ii) $(0.04, 0.09)$ denoted as vertical. All other shapes are mapped to one of these values based on their similarity. For the square patterns, they are mapped to one of the two categories randomly. It is important to note that the proposed encoding scheme can be extended to include more SRAF types at the cost of additional channels in the input image.

In total, three channels are used to encode each clip in the given dataset where one channel is used to encode target patterns and the other two represent the two types of SRAFs. However, note that the proposed approach is general, and the number of channels to be used can be set by the user depending on the data. The GAN models introduced in Section III are trained using the prepared dataset. The details of the training setup for both GAN models are summarized in Table IV.

B. Testing Data Description

The testing data set contains 404 layout clips from which the input layout images are obtained using the multi-channel heatmap encoding presented in Section III-A. These layouts are generated using the same setup as that for the training dataset presented in Section IV-A. The histograms in Figs. 10 and 11 show the distribution of the layout area (in um^2) and the contacts count per layout for the 404 layouts used in the testing dataset.

C. SRAF Generation

To demonstrate the efficacy of our proposed approach, we compare the layouts generated from our CGAN and CyGAN

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/TCAD.2020.2995338, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

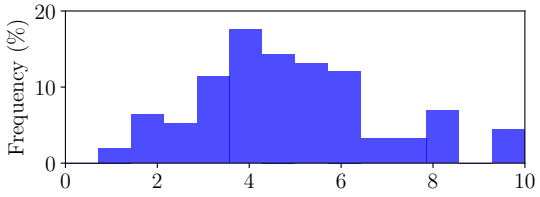


Fig. 10: The distribution of layout area in the testing dataset in um^2.

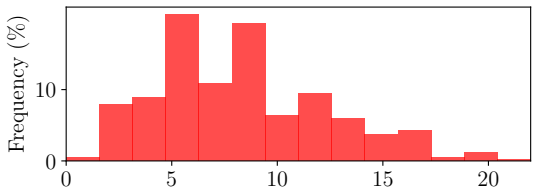


Fig. 11: The distribution of contact count per layout in the testing dataset.

with (i) those obtained from the local sampling scheme approach presented in [4] where Support Vector Machine is used as the classification model (denoted LS_SVM) and (ii) those obtained from model-based SRAF generation (denoted MB). The performance of the four different methods, under the same setup, can be visualized in Fig. 12 which shows the result for SRAF generation using MB,LS_SVM, CGAN and CyGAN for two layout clips in the testing dataset. For each clip, the first column shows the layout with no SRAF (denoted Target) and the results for SRAF generation using MB approach. The other three columns show the results for LS_SVM, CGAN, and CyGAN respectively where the first row shows the result before the legalization step described in III-E; i.e., direct output of the models. The SRAF schemes after legalization are shown in the second row.

By examining the results in Fig. 12, and recalling that CGAN, CyGAN and LS_SVM are trained against the model-based SRAF results as the golden solution, one can easily conclude that the generated SRAFs from CGAN and CyGAN mimic the MB results much better than the SRAFs from LS_SVM even before legalization. Moreover, comparing the legalized SRAFs obtained from the GAN models and the MB results shows that, although the results do not match exactly, both CGAN and CyGAN have captured the systematic way of generating the SRAFs in a model-based approach. In addition, one can notice a relatively small difference between the predicted SRAFs and final SRAFs for the case of CGAN and CyGAN when compared to those of LV_SVM. This is due to the fact that the GAN generated SRAF schemes are very close to the legal layout; hence, minimal changes are needed in the post processing stage. On the other hand, LS_SVM generates a clip of clustered SRAFs which requires intensive post-processing before obtaining a legal layout.

D. Lithography Compliance Check

To validate the lithography compliance of our proposed approach, we integrate the SRAF generation with a complete mask optimization flow using Mentor Calibre. The four different generation schemes are compared in terms of PV band and EPE. For each contact, the PV band value is measured, and the EPE value at the center of the edges at nominal conditions is considered. Table V summarizes the mean absolute values of the two metrics. The table also includes the PV band and EPE evaluations with no SRAFs to better demonstrate the performance gain achieved through SRAF. In practice, the most important metric of evaluation is the PV band (smaller is better) [4], and as demonstrated by the results, both CGAN and CyGAN can achieve better PV band when compared to LS_SVM [4] which demonstrates a superior performance in terms of SRAF insertion quality. Moreover, the table shows that the CyGAN scheme can achieve a PV band value which is even better than the MB approach. This is in fact consistent with our observation in Section IV-C where SRAF schemes generated from the GAN models can better mimic the MB generated schemes. On the other hand, despite the fact that LS_SVM can achieve better results in terms of EPE, EPE can be further improved with better OPC [34]; hence, it is not the best metric used to judge upon the quality of SRAF generation [4]. In general, the four SRAF generation schemes -MB, LS_SVM, CGAN and CyGAN- achieve comparable results in terms of lithography evaluation metrics with CGAN demonstrating superior results compared to LS_SVM, and CyGAN achieving better PV results compared to MB . This can be better seen when examining the histograms in Figs. 13 and 14 showing the distribution of EPE and PV band respectively across all clips in the testing dataset. The figures clearly show that the GAN performance is comparable to that of LS_SVM and MB, and at the same time, there exists a significant difference in the metrics between the case with no SRAF and the those with SRAF generated.

	No SRAF	MB	LS_SVM	CGAN	CyGAN
PV band (*0.001 um^2)	3.354	2.845	3.009	2.916	**2.773**
PV band ratio	1	0.848	0.897	0.869	**0.827**
EPE (nm)	3.9287	0.5270	**0.5067**	0.5410	0.5721
EPE ratio	1	0.134	**0.129**	0.138	0.146
Runtime (sec)	-	6910	700	**48**	**45**

TABLE V: The comparison of evaluation metrics and run time across different SRAF generation schemes is shown.

E. Runtime

Most importantly, considering the overall runtime for generating SRAFs (including post processing time) for all clips in Table V, GAN models can achieve $\sim 14.6\times$ runtime reduction when compared to LS_SVM while achieving better PV band, and $\sim 144\times$ when compared to model based approach while achieving comparable results when compared to the model based approach and the state-of-the-art machine learning based approach in [4].

F. CGAN vs CyGAN

In this section, we present an analysis of the two proposed models, CGAN and CyGAN, in terms of performance, runtime and robustness.

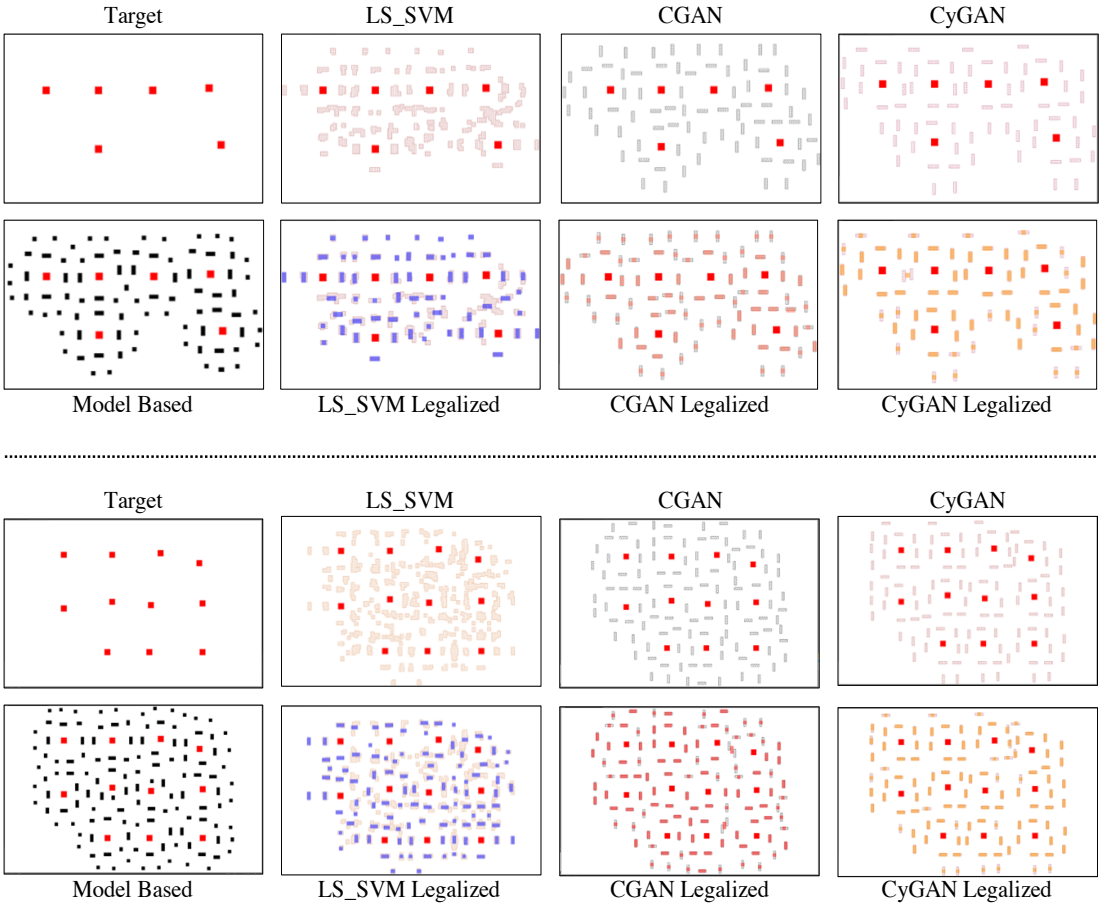


Fig. 12: The results of SRAF generation using the four approaches for two sample clips in the test dataset is shown. For each clip, the first column contains the target layout and the layout with SRAF inserted using the MB approach. For the other three columns, the first row shows the direct results from LS_SVM, CGAN, and CyGAN. The second row contains the legalized layouts for these three approaches.

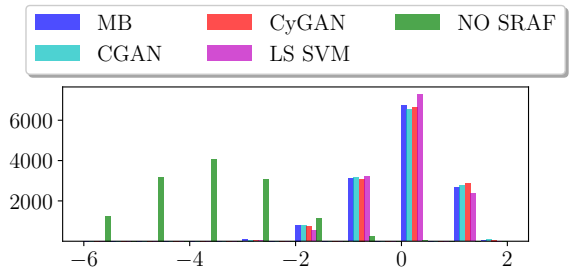


Fig. 13: The comparison of EPE distribution across different SRAF generation schemes is shown.

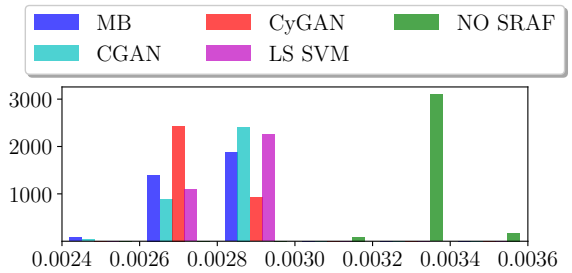


Fig. 14: The comparison of PV band distribution across different SRAF generation schemes is shown.

1) Performance: By examining the normalized results of PV band and EPE in Table V, one can clearly notice that the four different SRAF insertion approaches achieve comparable results when compared to the case with no SRAF. However, results also show better results for CyGAN in terms of PV band.

Based on the model description in Section III-C, it is clear that the CyGAN model is a more complex one with multiple residual blocks at the bottleneck layer which give more learning capacity for the model when compared to the vector latent space representation in CGAN. In fact, image translation models targeting high definition images, such as *pix2pixHD* [35], have adopted such residual block scheme due to its better performance in capturing spatial information. Besides, the cycle consistency loss can also be viewed as a form of regularization. By enforcing cycle consistency, CycleGAN framework prevents generators from excessive hallucinations and mode collapse, which typically cause loss of information and thus increase in cycle consistency loss [36]. Hence, these facts contribute to better generality of

CyGAN and the superior performance demonstrated in Table V.

On the other hand, both Table V and the histogram shown in Fig. 13 indicate that, when taking the no SRAF case as a reference, the four SRAF insertion approaches achieve comparable EPE results. Here again, LS_SVM shows a slight advantage in terms of EPE. However, it is important to note that EPE is not the best metric used to judge upon the quality of SRAF generation. While it is still an important issue to address, one can rely on other techniques to improve EPE after SRAF insertion such as OPC. This can be also accelerated with machine learning based techniques such as GAN-OPC [16].

2) Runtime: The runtime values reported in Table V show a slight advantage for CyGAN over CGAN. It is important to note here that this runtime includes both the model inference time and post-processing time including the decoding process described in Section III-D and writing out GDSII files. In practice, the inference time for both models does not exceed 30% of the overall time reported in Table V with 10 and 12.5 secs for CGAN and CyGAN, respectively. This difference can be attributed to the difference in model complexities where CyGAN has a more complicated structure.

Despite the slight advantage of CGAN in terms of inference time, its overall runtime, which is still governed by the post-processing steps, is higher than that of CyGAN. This is mainly due to the fact that, unlike the inference time, the post-processing time is data dependent. In other words, the number of SRAFs generated by the model affects the process of mapping to GDSII layout file. For the testing dataset at hand, CGAN generated 7% more SRAF objects than CyGAN which resulted in slightly longer post-processing time, and hence overall time.

However, when comparing to both baseline approaches LS_SVM and MB, the runtime values of both CGAN and CyGAN are in the same order and achieve similar speedup with $14-15\times$ and $144-150\times$ comparing to LS_SVM and MB, respectively.

3) Robustness: In general, CGAN is regarded as a more stable model, from a training perspective, due to its more strict supervision when compared to the cycle consistency loss in CyGAN which can result in issues related to training convergence and generalization such as information hiding [37]. But such issues were not encountered in our experiments.

In practice, the path from D_S to D_T is the one prone to information hiding. However, generated images in D_T throughout our experiments had excitations on the red channel only, with some occasional random noise on other channels at the edges of the red excitations. This is consistent with the encoding scheme, and it reflects the stability of learning for both paths in the cycle. Moreover, any information hiding can be detected when testing the SRAF insertion model. In the test mode, only one path of the cycle is used from D_T to D_S where the input images in D_T are golden images and not generated by the model; hence, they cannot carry any hidden information. Therefore, if the desired path was relying on hidden information embedded in the generated D_T images during the cycle training, it should fail when such information is not present during testing. However, as shown by our experimental results above, the model was generalizing well to the testing data where the cycle is not complete.

On the other hand, the concern of non-unique mapping between input and output is always present in supervised learning schemes and can be an issue for CGAN training. This can happen when the same sample in D_T has several possible MB results in D_S which are present in the training data. In our experiments, the training data was randomly generated with multiple levels of random variations (e.g., the size of layout, number of contacts, contact grid locations, random shift from the grid for each contact, etc.) which make the probability of having the exact same design repeated very low. In an ideal case, the mapping should be 1-to-1 in the training dataset. In our setup, and while no pre-screening was done, the probability of having exactly identical masks is very low such that it does not jeopardize the training process. The main concern here is related to the convergence of the model. When non-unique mappings exist, their frequency plays a major role in assessing their impact. In the low frequency case, the model tends to adopt the mapping that is best aligned with the other samples in the training data while ignoring the other redundant mapping since this would result in minimizing the training loss function. However, as the frequency of such case increases, the model might face a convergence issue since the mini-batch gradient descent method can result in oscillation between the different mappings if they are of equal influence. Therefore, in general, it is important to account for this factor when preparing the training dataset, since the uniqueness of mapping is always desired to avoid convergence issues.

V. Conclusion

In this paper, a novel SRAF generation framework, *GAN-SRAF*, is presented based on generative adversarial neural networks. The proposed approach casts the SRAF generation problem into an image to image translation task where GAN has demonstrated impressive capabilities. Two GAN schemes, namely CGAN and CyGAN, were presented to tackle SRAF insertion with paired and unpaired data respectively. We propose an effective encoding scheme to represent the layout information using a multi-channel heatmaps and a GPU-accelerated decoding scheme for extraction of SRAF solutions. The experimental results demonstrate that GAN-SRAF achieves $\sim 14.6\times$ reduction in computational cost compared to state-of-art machine learning SRAF generation approaches, and $\sim 144\times$ when compared to model-based approach, while achieving comparable quality.

VI. Acknowledgement

The authors would like to thank Dr. Xiaoqing Xu (Arm Inc) and Ahmed Omran (Synopsys Inc) for the helpful discussions.

References

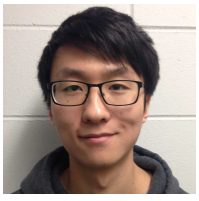
[1] M. B. Alawieh, Y. Lin, Z. Zhang, M. Li, Q. Huang, and D. Z. Pan, "GAN-SRAF: Sub-resolution assist feature generation using conditional generativeadversarial networks," in *Proc. DAC*, 2019.

[2] R. Viswanathan, J. T. Azpiroz, and P. Selvam, "Process optimization through model based SRAF printing prediction," in *Proc. SPIE*, 2012, pp. 83 261A–83 261A.
[3] K. Sakajiri, A. Tritchkov, and Y. Granik, "Model-based SRAF insertion through pixel-based mask optimization at 32nm and beyond," in *Proc. SPIE*, 2008, pp. 702 811–702 811.
[4] X. Xu, Y. Lin, M. Li, T. Matsunawa, S. Nojima, C. Kodama, T. Kotani, and D. Z. Pan, "Sub-Resolution Assist Feature Generation with Supervised Data Learning," *IEEE TCAD*, vol. PP, no. 99, 2017.
[5] J.-H. Jun, M. Park, C. Park, H. Yang, D. Yim, M. Do, D. Lee, T. Kim, J. Choi, G. Luk-Pat *et al.*, "Layout optimization with assist features placement by model based rule tables for 2x node random contact," in *Proc. SPIE*, 2015, pp. 94 270D–94 270D.
[6] C. Kodama, T. Kotani, S. Nojima, and S. Mimotogi, "Sub-resolution assist feature arranging method and computer program product and manufacturing method of semiconductor device," Aug. 19 2014, US Patent 8,809,072.
[7] B.-S. Kim, Y.-H. Kim, S.-H. Lee, S.-I. Kim, S.-R. Ha, J. Kim, and A. Tritchkov, "Pixel-based SRAF implementation for 32nm lithography process," in *Proc. SPIE*, 2008, pp. 71 220T–71 220T.
[8] Y. Lin, M. B. Alawieh, W. Ye, and D. Pan, "Machine learning for yield learning and optimization," in *Proc. ITC*, 2018.
[9] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.
[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.
[13] B. Wu, H. Duan, Z. Liu, and G. Sun, "SRPGAN: perceptual generative adversarial network for single image super resolution," *CoRR*, vol. abs/1712.05927, 2017.
[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
[15] M. B. Alawieh, Y. Lin, W. Ye, and D. Z. Pan, "Generative learning in VLSI Design for Manufacturability: Current status and future directions," *Journal of Microelectronic Manufacturing*, vol. 2, no. 4, 2019.
[16] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "Gan-opc: mask optimization with lithography-guided generative adversarial nets," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 131.
[17] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *Proc. DAC*, 2019.
[18] W. Ye, M. B. Alawieh, Y. Watanabe, S. Nojima, Y. Lin, and D. Z. Pan, "TEMPO: Fast mask topography effect modeling with deep learning," in *Proc. ISPD*, 2020.
[19] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. Iyer, and D. Z. Pan, "High-definition routing congestion prediction for large-scale FPGAs," in *Proc. ASPDAC*, 2020.
[20] B. Xu, Y. Lin, X. Tang, S. Li, L. Shen, N. Sun, and D. Z. Pan, "WellGAN:generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Proc. DAC*, 2019.
[21] X. Zhou, A. Karpur, C. Gan, L. Luo, and Q. Huang, "Unsupervised domain adaptation for 3d keypoint prediction from a single depth scan," *CoRR*.
[22] S. Tulsiani and J. Malik, "Viewpoints and keypoints," *Proc. CVPR*, pp. 1510–1519, 2015.
[23] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.
[24] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *NIPS*, 2014.
[25] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *CoRR*, 2015.
[26] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*.
[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.
[29] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.
[30] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, 2016, pp. 694–711.
[31] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.
[32] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 702–716.
[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." vol. 16, 2016, pp. 265–283.
[34] C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*. John Wiley & Sons, 2008.
[35] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "pix2pixhd: High-resolution image synthesis and semantic manipulation with conditional gans."
[36] T. Wang and Y. Lin, "Cyclegan with better cycles."
[37] C. Chu, A. Zhmoginov, and M. Sandler, "Cyclegan, a master of steganography," *arXiv preprint arXiv:1712.02950*, 2017.

Mohamed Baker Alawieh (S'16) received the B.S. degree in electrical and computer engineering from the American University of Beirut, Beirut, Lebanon, in 2014 and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2016. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Texas at Austin, Austin, TX, USA. He is a collaborator at Stroke Thrombectomy and Aneurysm Registry (STAR) as part of the STAR-AI team at the Medical University of South Carolina.

His current research interests include machine learning, VLSI computer-aided design and medical data analytics.

Yibo Lin (S'16–M'19) received the B.S. degree in microelectronics from Shanghai Jiaotong University in 2013, and his Ph.D. degree from the Electrical and Computer Engineering Department of the University of Texas at Austin in 2018. He is currently an assistant professor in the Computer Science Department associated with the Center for Energy-Efficient Computing and Applications at Peking University, China. His research interests include physical design, machine learning applications, GPU acceleration, and hardware security. He has received 3 Best Paper Awards at premier venues (DAC 2019, VLSI Integration 2018, and SPIE 2016). He has also served in the Technical Program Committees of many major conferences, including ICCAD, ICCD, ISPD, and DAC.



Zaiwei Zhang received the B.S. degree in Computer Engineering from Purdue University West Lafayette, IN, USA in 2015. He is currently pursuing the Ph.D. degree Computer Science at the University of Texas at Austin, Austin, TX, USA. His research interests include computer graphics and machine learning. He is particularly interested in applying machine learning, especially deep learning, to graphics.

Meng Li (S'15–M'18) received the Ph.D. degree from the University of Texas at Austin, Austin, Tx, USA under the supervision of Dr. D. Z. Pan in 2018. He is currently an AI research scientist in Facebook. His research interests include AI Software/Hardware Co-design, hardware-oriented security and reliability. Dr. Li was the recipient of the UT Austin Margarida Jacome Outstanding Dissertation Prize in 2019, the EDAA Outstanding Dissertation Award in 2019, the First Place in the Grand Final of ACM Student Research Competition in 2018, the Best Poster Award in ASPDAC Ph.D. forum in 2018, the UT Austin University Graduate Fellowship in 2013, as well as the Best Paper Award in HOST'2017 and GLSVLSI'2018.

Qixing Huang received the Ph.D. degree in Computer Science from Stanford University, Stanford, CA, USA. He is currently an assistant professor of Computer Science at the University of Texas at Austin. He was a research assistant professor at Toyota Technological Institute at Chicago before joining UT Austin.

Dr. Huang's research spans the fields of computer vision, computer graphics, and machine learning, and publishes extensively in venues such as SIGGRAPH, CVPR, ICCV, ECCV, NeuriPS, ICML, and etc. In particular, his recent focus is on developing machine learning algorithms (particularly deep learning) that leverage Big Data to solve core problems in computer vision, computer graphics and computational biology. He is also interested in statistical data analysis, compressive sensing, low-rank matrix recovery, and large-scale optimization, which provides theoretical foundation for his research. He also received the best paper award at the Symposium on Geometry Processing 2013, the best dataset award at the Symposium on Geometry Processing 2018, and the most cited paper award of Computer-Aided Geometric Design in 2010 and 2011.

David Z Pan (S'97-M'00-SM'06-F'14)received his B.S. degree from Peking University, and his M.S. and Ph.D. degrees from University of California, Los Angeles (UCLA). From 2000 to 2003, he was a Research Staff Member with IBM T. J. Watson Research Center. He is currently Engineering Foundation Professor at the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, machine learning and hardware acceleration, design/CAD for analog/mixed signal designs and emerging technologies. He has published over 375 journal articles and refereed conference papers, and is the holder of 8 U.S. patents.

He has received a number of prestigious awards for his research contributions, including the SRC Technical Excellence Award in 2013, DAC Top 10 Author in Fifth Decade, DAC Prolific Author Award, ASP-DAC Frequently Cited Author Award, 18 Best Paper Awards at premier venues (ASPDAC 2020, DAC 2019, GLSVLSI 2018, VLSI Integration 2018, HOST 2017, SPIE 2016, ISPD 2014, ICCAD 2013, ASPDAC 2012, ISPD 2011, IBM Research 2010 Pat Goldberg Memorial Best Paper Award, ASPDAC 2010, DATE 2009, ICICDT 2009, SRC Techcon in 1998, 2007, 2012 and 2015) and 15 additional Best Paper Award finalists, Communications of the ACM Research Highlights (2014), UT Austin RAISE Faculty Excellence Award (2014), and many international CAD contest awards, among others.