BeGAN: Power Grid Benchmark Generation Using a Process-portable GAN-based Methodology

Vidya A. Chhabria*, Kishor Kunal*, Masoud Zabihi*, and Sachin S. Sapatnekar
University of Minnesota, Minneapolis, MN 55455

***Abstract*—Evaluating CAD solutions to physical implementation problems has been extremely challenging due to the unavailability of modern benchmarks in the public domain. This work aims to address this challenge by proposing a process-portable machine learning (ML)-based methodology for synthesizing synthetic power delivery network (PDN) benchmarks that obfuscate intellectual property information. In particular, the proposed approach leverages generative adversarial networks (GAN) and transfer learning techniques to create realistic PDN benchmarks from a small set of available real circuit data. BeGAN generates thousands of PDN benchmarks with significant histogram correlation (p-value $\leq$ 0.05), demonstrating its realism and an average L1 Norm of more than 7.1%, highlighting its IP obfuscation capabilities. The original and thousands of ML-generated synthetic PDN benchmarks for four different open-source technologies are released in the public domain to advance research in this field.**

I. Introduction

The performance of new EDA algorithms and tools must be validated against prior art by testing their performance on a representative set of benchmark circuits. For this reason, circuit benchmarks have served as a linchpin in advancing EDA research for decades. Synthesis benchmarks [1]–[5] have driven design and test for many decades; placement and routing benchmarks [6]–[8] have prompted new approaches in physical design that have enabled new generations of open-source and commercial tools; power delivery network (PDN) benchmarks [9]–[11] have served as standard testcases for a large volume of work on the analysis and optimization of IR drop and electromigration (EM) reliability. However, benchmarks may become obsolete [12] and must constantly be updated to address technology scaling, new design classes, and new constraints. The process of releasing public-domain benchmarks is laborious, requires intense volunteer effort, and must overcome secrecy hurdles before release.

It is important to note that the paucity of benchmarks is not limited to public-domain benchmarks for academia. In principle, industry research and development groups have access to advanced circuits, implemented in cutting edge technologies. However, this apparent "problem of plenty" hides the fact that deriving meaningful benchmarks involves the labor-intensive process of curating information from design databases, overcoming internal barriers across groups within companies. Benchmarks must be extracted at an appropriate level of abstraction to exercise a specific tool within the design flow, while successfully encapsulating IP information in both design and technology. A design house or an EDA vendor who works with multiple companies may be contractually limited in sharing designs across projects. Moreover, the designers who create the design data are typically preoccupied with their current project and pressing deadlines, and have limited time to extract data from older designs.

As a result, in both academia and industry, EDA tool developers must work with a limited number of benchmarks that are relevant to a specific technology node or design style. A larger variety of testcases would serve to stress-test new algorithms and tools to make them more robust. The importance of generating realistic design data is compounded by a growing interest in machine learning (ML) based EDA, which has engendered a strong need for large benchmark suites for ML training, validation, and test sets.

* (equal contribution)

To overcome these problems, we propose BeGAN, an ML-based methodology for realistic, synthetic PDN benchmark generation. We employ generative adversarial networks (GANs) [13] to extract the characteristics of real current maps (CMs) in a training set and generate a large set of realistic synthetic PDN benchmarks (CMs and PDNs) that preserve these characteristics. GANs have been used in the ML world to synthesize images, based on training sets of many thousands of images, that are as realistic to the human eye as real images [14]–[16]. In the context of benchmark generation, we face a further constraint that our GAN-based approach must be trained on limited data: unlike image GANs that are trained on numerous widely available images, it is near-impossible to curate a database of thousands of real circuits even in industry. We use insights from recent work on training GANs using small datasets [17], [18].

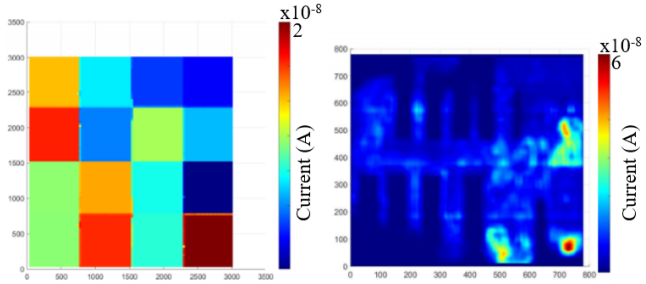

Figure 1: Current maps of (a) IBM power grid benchmark with uniform currents per region. (b) a RISC-V core.

We demonstrate our methodology by focusing on PDN benchmarks for static IR/EM analysis and PDN synthesis. While PDN benchmarks have been released in the past [10], [19], the methods used to generate them are ad hoc and do not focus on the generalizability of the benchmarking methodology across technologies and designs. In addition, existing PDN benchmarks [9], [10] are dated, few in number, and unsuitable for evaluating state-of-the-art CAD solutions for several reasons:

1) **Old technology:** These benchmarks use old technology with Al interconnect, 1.8V supply voltages, and most via resistances are zero. Today's interconnects now use Cu wires; supply voltages are much lower; and via resistances are significant.
2) **Unrealistic assumptions to protect IP:** The benchmarks provide region wise uniform currents, but realistic analysis requires finer granularity. Fig. 1 compares the CM of the IBM benchmarks with that of a modern RISC-V core.
3) **Limited number of benchmarks:** The available PDN benchmarks show little diversity and all use the same underlying current distribution. A wider diversity of benchmarks is particularly useful for trending research that applies supervised ML to PDN problems [20]–[23].
4) **Extremely high static IR drop:** Typically static IR drop values are within 5% of VDD, but these benchmarks display worst-case IR drops of up to 44% of VDD.

The main contributions of BeGAN are as follows:

1) We propose a generalizable GAN and transfer learning-based methodology for PDN benchmark generation.

2) We demonstrate the use of this methodology across four different technologies (including a commercial 12nm FinFET node) and show that the generated PDN benchmarks (CMs and power grids) protect IP while maintaining key benchmark traits.
3) We release [24] a suite of real circuit benchmarks and thousands of BeGAN PDN benchmarks for each (publicly-available) technology in the public domain.

Next, in Section II, we outline our overall approach to PDN benchmark generation. The details of the GAN-based ML approach are described in Section III, followed by a description and evaluation of the generated benchmarks in Section IV and V.

II. PDN Benchmark Generation Framework

A. A PDN benchmark

PDN benchmarks are useful to evaluate PDN synthesis [20] and PDN analysis methodologies [21]. The former requires benchmarks that provide current distributions while the latter requires current distributions, power pad locations, and synthesized PDNs. The common key aspect in the benchmarks is the current distribution, which models the place and route information for the design through block locations and power values per block.

In this work, we focus on the primary challenge of generating realistic CMs, i.e., 2-D spatial distributions of the current sources. For these CMs, PDN topologies and power pad locations for the PDN analysis problem can be generated by using open-source available software such as OpeNPDN [20].

Typical power grid benchmarks [11] model the PDN in the form of a circuit netlist consisting of resistors, voltage, and current sources as shown in Fig. 2. The resistor values are derived from the structure of the PDN, the voltage sources from the power pads, and the current sources from the location and current drawn by each instance.

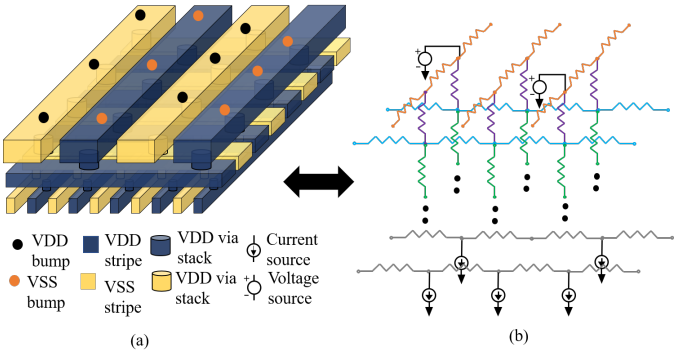

Figure 2: (a) A multilayer PDN stack and (b) its circuit netlist model.

A public domain release of several CMs present issues on two fronts:

- Since CMs encode information of the underlying design (cell locations) and technology libraries (current values), they may include sensitive IP that must be obfuscated.
- As mentioned in Section I, in both academia and industry, it is difficult to obtain a sufficiently large number of designs to exercise conventional or ML-based EDA tools.

BeGAN overcomes the above two issues by leveraging GANs to generate thousands of IP-obfuscated CMs in a process-portable manner. Using a small number (typically < 10) of CMs associated with real circuits, and our GAN-generated maps maintain the key traits of the original circuits while protecting IP. The rest of the benchmark, i.e., the PDN structure and bump assignments, is obtained from OpeNPDN [20] by using a combination of valid PDN templates and bump assignment strategies.

B. BeGAN framework

The BeGAN framework shown in Fig. 3 highlights our two-staged process-portable PDN benchmark generation method. The benchmarks are generated in the form of circuit netlists. The first stage is the GAN-based CM generation (Fig. 3(left)), and the second stage is the circuit generation stage which includes power grid synthesis and power bump assignment using OpeNPDN (Fig. 3(right)). The primary new technical challenge we address in this paper is in the *first stage*, which uses a GAN to generate realistic CMs representing patterns seen in real circuits.

1) GAN training for current maps: GANs are trained by a pair of neural networks, a generator and a discriminator, which operate adversarially, and can be understood as a two-player zero-sum game. During adversarial training, the discriminator distinguishes between the real data and synthetic data from the output of the generator and provides feedback to the generator to adjust its weights. The generator aims to produce a realistic synthetic output while the discriminator learns to avoid being fooled by the synthetic data from the generator by comparing the generated images to ground truth training images. Depending on whether the generator or the discriminator "wins," the weights of the loser are iteratively adjusted.

Typically, GANs have a large number of parameters: for example, the widely-used DCGAN [14] architecture has 36M trainable parameters. Traditional approaches for training GANs have relied on large datasets with thousands of points [25]. In the IC world, the number of realistically available datasets is minimal, even when we augment the training data with image variants generated by traditional techniques such as image shift, zoom, rotation, or flip. A conventional approach is prone to overfitting on limited data.

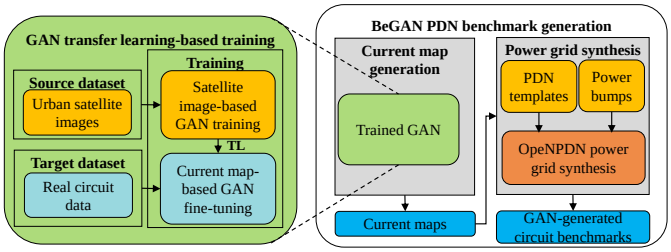

Figure 3: Overview of BeGAN benchmark generation: GANs are trained using TL to generate current maps. PDNs and power pad locations are generated using OpeNPDN [20].

We leverage recent advances [17], [18] on image generation from small datasets that are motivated by the high expense of data collection or privacy issues. In our approach, summarized in Fig. 3 (green box), we first pretrain a GAN using a large set of synthetic process-technology-independent images in a *source dataset* that have similar characteristics as on-chip CMs. Next, we tune this pretrained GAN model using transfer learning (TL) from a source dataset to a *target dataset* of CMs from a small set of real circuit designs at a specific technology node generated by the OpenROAD flow [26]. The pretraining of the GAN with the source dataset is process-technology-independent, i.e., the same model can be reused across technologies. A new process node only demands fine-tuning the pretrained weights for the new target dataset, ensuring our methodology is process-portable with a low computational cost. During inference, the GAN can rapidly generate thousands of CMs that maintain the key features of the real maps.

2) Benchmark circuit generation using OpeNPDN: Once the current pattern is generated, in the *second stage*, we supply interconnect and package parameters to OpeNPDN [20], an existing PDN topology generator to build a power grid for each CM. OpeNPDN uses the concept of PDN templates where each template is a DRC-correct,

valid building block of the PDN that varies in its metal layer utilization and hence equivalent resistance. For each CM in a specific technology, OpeNPDN uses various combinations of existing PDN templates from the existing set to create unique PDNs for each of the GAN-generated CMs. The VDD/VSS IO locations are based on a standard flip-chip package with different power assignment strategies, which aid in the diversity of the dataset.

For the circuit netlist, the intermediate metal layers of the synthesized PDN stripes are discretized into nodes at via locations, and the uppermost and lowermost layer stripes are discretized at a small, fixed interval to snap the current sources and voltage sources to the closest PDN node accurately. The resistance value for each element is based on the length, width, and per-unit resistances of the PDN segment in that specific technology.

With the above BeGAN framework, we have released thousands of circuit benchmarks on GitHub [24]. The benchmarks are released for three different open-source technologies, FreePDK45 with the Nangate Open Cell Library [26], SkyWater130nm (HD) [27], and ASAP 7nm [28]. However, due to the process-portable nature of our methodology, the same approach may be used with proprietary parameters as we have demonstrated with a commercial 12nm FinFET technology.

III. GAN and TL-based Current Map Generation

A. Synthetic image generation for GAN pretraining

A TL-based GAN training methodology helps overcome challenges pertaining to the limited number of available benchmarks. To employ the framework highlighted in Fig. 3 (green boxes), we must synthetically generate a set of source benchmarks that are sufficiently similar to the target datasets [25]. However, this is not a trivial extension of existing ML-world methods: a very domain-specific issue that must be overcome is the generation of the generic data images that are used in the source dataset. A vital aspect for the successful application of TL to training the GAN is to ensure similarity between the source and target datasets [25] as a pretrained source model is only fine-tuned during training with limited data. Moreover, the selection of the GAN architecture plays a vital role in ensuring that the target dataset performs incremental tuning on the pretrained model.

To guide the choice of synthetic images, we examine some key characteristics of chip-level CMs:

- Chips are placed and routed using Manhattan geometries and often contain rectangular macros or blockages.
- Due to well-known limitations on chip power and dark-silicon trends, the number of current hot spots is small.
- Modern chips are packaged using C4 technology with area IO, due to which chip hotspots have a relatively small footprint and may occur anywhere in the chip.

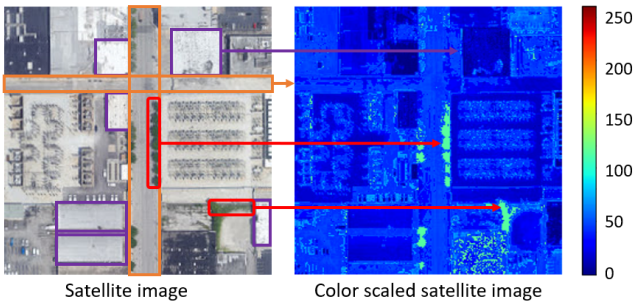

Figure 4: Urban satellite image and a derived CM.

Many of the features of chips with macros map on well to satellite images of urban areas in the United States where many features have straight edges, as illustrated by the example in Fig. 4. For example, the shapes and relative sizes of building rooftops (violet rectangles) are similar to macro blocks; the roads (red rectangles) form Manhattan geometries, and are similar to channels between macro blocks that may contain random logic; small, round boulevard trees (within the green rectangle) appear similar to current hotspots; and multiple smaller objects with different color intensities can be mapped to local current variations spread across the design. As shown in the figure, after appropriate color transformations, the spatial correlations of the satellite images (when converted to intensity distributions) are similar to CM heatmaps.

As we note from the transformation in Fig. 4, the image colors at left must be transformed to those at right to obtain a "chip-like" CM. In real designs, CMs are drawn on a jet scale colormap where a red color maps to high current density and a blue color maps to low current density. We also observe that in general, macros have lower current density, and the constrained channels have a higher number of hotspots. Our first transformation from the colormap of a satellite image to a jet colormap maps the color green (representing trees) to yellow/red, corresponding to higher current densities in hotspots.

Our next transformation matches the color density to that of real chip CMs, taken from the images of the small set of available circuits. Each RGB color pixel can have a value between 0–255. For a map to be realistic the average of these values across all pixels should not be too high, otherwise it will represent an unrealistic design with very large hotpots. We observed that the average value in CM images lies between 10% to 40% of the maximum of 255. In contrast, a transformed satellite image may have all pixels in a high range, which does not represent a valid chip CM. To avoid this, we scale the average color value of the transformed satellite images to a mean value, chosen randomly in the realistic range (10–40% of 255). As a result, the overall image, which has a preponderance of gray/brown colors, is mapped to a largely blue colormap with some hotspots.

We will show in Section IV that this transformation is successful in creating a viable source dataset, whose quality is superior to other families of images that were used as candidates for the source dataset.

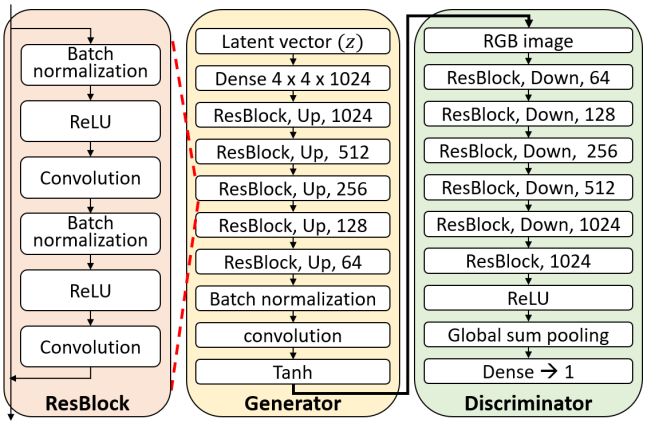

Figure 5: Architecture of SNGAN for image generation.

B. GAN architecture

Since their introduction six years ago [13], there have been rapid advances in the area of GANs. Some notable works include DCGAN [14], ProGAN [29], StyleGAN [15], BigGAN [30], and SNGAN [31]. We choose SNGAN as our source model because it has a well-written code base, making it easy to adapt to our application, and because its compute power requirements are modest as compared to alternatives such as BigGAN.

The SNGAN model has 90M trainable parameters and operates on images of size 128×128 pixels. The pitch of the M1 power rail and M4 power stripe above it is 2.4μm and 56μm respectively in a

45nm technology. At this granularity, a 128×128 current resolution is sufficient for a chip of 1mm×1mm where a pixel represents the current within a 7.8μm×7.8μm area, which is much lower than the M4 pitch. A notable feature of SNGAN is that it addresses instability issues in GAN training, by using spectral normalization of model weights to stabilize the training of the discriminator. Spectral normalization constrains the Lipschitz constant for regularization by controlling the spectral norm of each layer, thus providing better results and lower computational cost.

The SNGAN generator model is shown in Fig. 5, where the input is a latent vector $z \in \mathcal{R}^{128} \sim \mathcal{N}(0, I)$, that is randomly sampled from a truncated normal distribution, with the truncation threshold set to 0.4. Truncating the latent vector z improves the quality of generated samples at the cost of reduction in sample variety [30]. The first layer is a dense layer that maps the latent vector z to $4 \times 4 \times 1024$ dimensional intermediate features. Next, it uses five ResBlock layers that help in image denoising by directly transferring the coarse images across deconvolution networks. Each of these ResBlock layers, shown at left in the picture, is composed of two 3×3 convolution layers with nearest-neighbor upsampling, conditional batch normalization and ReLU layers, as shown by the inset at left in the figure. Finally, the generator has a batch normalization layer, and a $\tanh$ activation function that generates an output image of size 128×128.

The discriminator, shown at right in the figure, uses an input RGB image. Five stages of downsampling are performed on this image using ResBlock, followed by ReLU activation and global sum pooling, which, respectively, help to address the vanishing gradient and to stabilize the training. The final dense layer classifies the input image as a real or synthetic CM. For our applications, we train SNGAN on 4,167 urban satellite images so that it learns a set of weights for multiple filters that extract useful features in the training data.

C. Transfer learning model

During the transfer learning step, the training data is a small set of current density maps derived from real chips. We use the pretrained generator model of the SNGAN topology in Fig. 6. Instead of training new filters from scratch, a combination of the pretrained filters can be fine-tuned for the target domain, represented by our small dataset. In place of a discriminator, we use a loss function L to measure the distance of generated image to the target image.

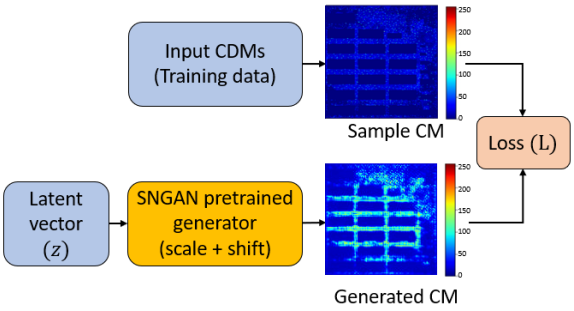

Figure 6: Transfer learning model using the pretrained SNGAN.

Our fine-tuning step uses a scale-and-shift transformation to each hidden activation of the generator as the transfer method [17]. Thus, the GAN does not require a full update of the entire model, but only learns scale-and-shift parameters to control the activation of filters, reducing the number of trainable parameters. For each channel of the hidden layer distribution, we update the filters as shown in Fig. 7:

$$W_l^{new} = \gamma_l \cdot W_l^{old} + \beta_l \tag{1}$$

Here, W_l^{old} represents the convolution filter in the l_{th} layer of the generator, W_l^{new} represents the set of updated weights after adaptation, γ_l represents the scale parameter for adaptation, and β_l represents the shift parameter. The initial values of these parameters are $\gamma_l = 1$ and $\beta_l = 0 \; \forall \; l$ and are updated by back-propagation during training. The scale-and-shift parameter is applied independently to each of the convolution layers of the five ResBlock layers (Fig. 5) to adapt the pretrained convolution filters in the generative model on the target dataset of CMs from real designs. We also update the γ and β parameters for class conditional batch normalization and the fully connected layers in the generator.

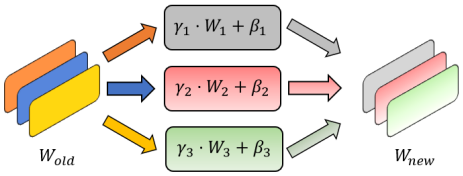

Figure 7: Scale-and-shift filters used for transfer learning.

The generator is trained using supervised learning as in Generative Latent Optimization (GLO) [32], by simultaneously estimating the latent vector z for all training data such that the generated data is close to the image in the training dataset. GLO leverages the inductive bias of deep ConvNets and learns to map learnable latent vectors to samples in a target dataset by minimizing reconstruction loss.

The loss function, L, used during training measures the distance of the generated image to the target image [17]:

$$L = ||x - G^{\text{new}}(z+\epsilon)||_1 + \textstyle\sum_l ||C^l(x) - C^l(G^{\text{new}}(z+\epsilon))||_1 + \text{L2 regularization}(z, \gamma, \beta) \tag{2}$$

The first component is the L1 loss, measured as the L1 Norm of pixel-wise distance between the generated images $G^{\text{new}}(z+\epsilon)$ and the corresponding target input images x, where, ϵ is a small random vector to avoid local minima. The second component is the perceptual loss between generated images and target input images, and is included based on recent works that have shown that high-quality images can be generated using perceptual loss [33]. This loss component is the sum, over all layers l, of the L1 Norm of differences between high-level image features, $C^l(x)$, that are extracted by passing both the generated and target images through pretrained VGG16 convolutional neural networks. The final component performs L2 regularization of training parameters Z, γ, and β to avoid overfitting.

IV. Evaluation of BeGAN-Generated Current Maps

For our experiments, we collect two datasets:

- A large dataset (24,000 datapoints) of urban satellite images from Google Maps, filtered down manually to 4,167 images with characteristics such as Manhattan edges, and a reasonable number, size, and distribution of trees (hotspots in the heat map).
- A small dataset of 10 circuit CMs using designs from [26]

The circuit designs correspond to open-source RTL designs available at [26], where the number of standard cells ranges from 10,000 to 300,000 instances, and the number of macros ranges from 0 to 30. We further subdivide the circuit-based CMs into two categories (Fig. 8) based on macro placement and current distribution: "Set 1" contains designs with no macros and with relatively uniform current distribution across the die, while "Set 2" contains designs with a large number of macros that result in nonuniform CMs. The layouts for "Set 1" are synthesized across four different technology nodes: ASAP 7nm, GF 12nm, FreePDK45 with Nangate Open Cell Library, and SkyWater 130nm(HD) using [26] with four circuit CMs in each set. These layouts are used to generate CMs: as an example, Fig. 9 illustrates CMs for the JPEG design across four technology nodes, showing significant differences across technology nodes. BeGAN is

trained on the features of each technology node separately. "Set 2" is synthesized only for the 45nm node due to the limited availability of memory compilers required to generate the macros present in the designs. For this node, "Set 2" contains six circuit CMs.

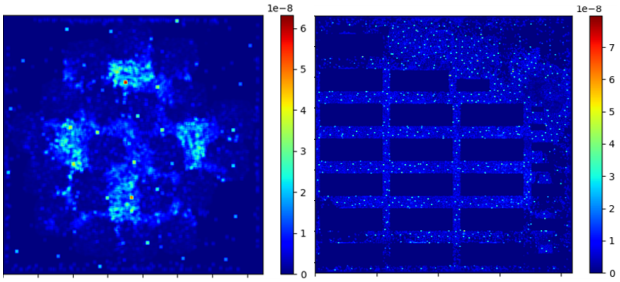

Figure 8: Example CMs (A/μm^2) in "Set 1" (left) and "Set 2"(right) on a 45nm technology node.

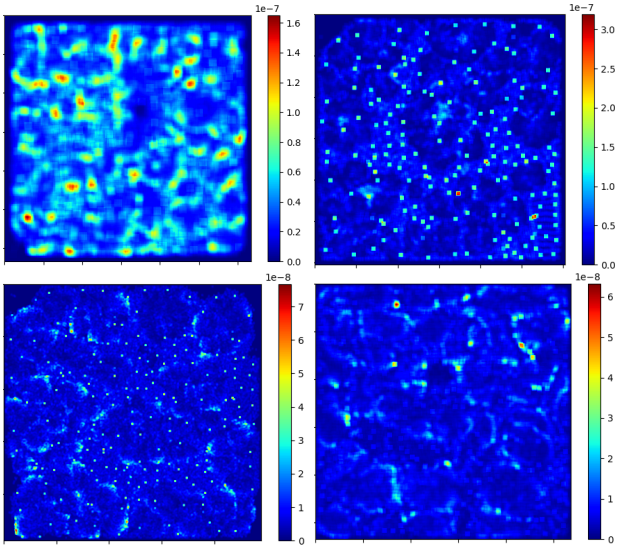

Figure 9: CMs of JPEG design from "Set 1" across four pdks ASAP 7nm (top-left), GF 12nm (top-right), FreePDK45 with Nangate Open Cell library (bottom-left), SkyWater 130nm(HD) (bottom-right)

We use the filtered urban satellite images for training the GAN once and then use TL to retrain the GANs on multiple sets of design CMs. During TL the scale-and-shift parameters are tuned based on the circuit designs. Each circuit CM in the set is of a different size representing the die area of the chip. However, for training the GAN we must use CMs of a fixed size to maintain consistency with the urban satellite images for the successful application of TL. Therefore, we convert the circuit CM representing the die area into a CM matrix with a fixed size of 128×128, where each value of the matrix is calculated by dividing the total current value within each region by the area of the region. This matrix is then converted to a 3-channel jet colormap image with pixel values between 0 to 255. During inference, the GAN generates a 3-channel RGB image of size 128×128 with intensity values from 0 to 255, representing a CM image.

We consider two problems to understand the strengths of this model: a *quality* problem to ensure that it can generate a large variety of high-quality CMs, and a *reconstruction* problem to verify that the model does not leak out training data.

A. Quality of GAN-generated CM Images

To quantify the quality of generated CM images, we propose to use three metrics to select and/or evaluate CM images based on their features and their correlation with the original CMs.

The **chi-square (χ^2) test** compares the histograms of the BeGAN-generated and real CMs. Histograms are generated by dividing the 0–255 pixel values for each color channel into eight bins and counting the number of pixels in a particular interval of the bins, which are then min-max normalized. We use the chi-square metric to select CM images that show a high correlation between generated and original CM histograms. For the i^{th} bin, if O_i and E_i are, respectively, the frequency of the BeGAN-generated and the circuit CMs,

$$\chi^2 = \sum_{i=1}^{k} (O_i - E_i)^2 / E_i \tag{3}$$

The χ^2 test determines whether there is an association between two histograms. A lower χ^2 value is better and a higher value indicates that two histograms are independent. For an identical histogram of current densities (note that identical histograms do not mean that the CM images are identical, because the histogram condenses information from the CM images), the χ^2 metric is zero, and it increases with the disparity between histograms. For an eight-bin histogram comparison, a χ^2 value more than 20 will result in a significance value below 0.05, meaning that the histograms are independent. We select the images with a statistically significant correlation between min-max normalized histograms (p-value ≤ 0.05).

Histogram correlation ensures that the total amount of area with high current density remains correlated but these areas may have different spatial distributions: two CM images having the same histograms can look different based on the position of hotspots. We ensure structural and spatial correlation between the hotspots using FID and EMD metrics from the machine learning world.

The **Fréchet Inception Distance (FID)** score is a standard ML-world metric that measures the quality of GAN-generated images. The FID score uses the Inception V3 network to capture computer-vision-specific features of a given image, and is calculated using the mean and covariance of these features. According to [34], there is a linear correlation between the FID score and the quality of images, where a lower FID corresponds to better quality images.

$$FID = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2 * \sqrt{(C_1 * C_2)}) \tag{4}$$

Here, μ_1 and μ_2 are the feature-wise means of the real and generated image features, C_1 and C_2 are the covariance matrices of the real and generated image features, and $Tr(.)$ finds the trace of a matrix.

The **Sliced Wasserstein Distance (SWD)** factorizes 2-D distributions of the CM images into 1-D distributions and then calculates the Wasserstein distance, also commonly known as earth mover's distance (EMD), between these 1-D distributions. EMD is intuitively explainable as the minimum cost of turning one 1-D topology ("pile of dirt"), representing a hotspot, into another, where the cost is the amount of dirt moved times the distance by which it is moved.

$$EMD(P, Q) = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} d_{i,j} \right) \Big/ \left(\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} \right) \tag{5}$$

Here, we assume the input CM distribution P has n clusters (groups of hotspot pixels projected on the 1-D plane) and the generated CM Q has m clusters. We denote the distance between the mean of clusters i and j by $d_{i,j}$, and the movement from cluster i of P to cluster j of q that minimizes the overall cost by $f_{i,j}$. A low EMD corresponds to a high correlation in the spatial variance.

Using the three metrics: We use the χ^2 test as a filter to discard insufficiently correlated CM images, and then use FID and SWD to check the similarity metrics of the remaining CM images. Table I shows the FID and SWD metrics for BeGAN, starting with either urban satellite images or ImageNet for training the GAN. We use the L1 Norm to show sufficient difference of the generated CM images from the original CM images, as discussed in the next section. The FID score is calculated between the selected set of CM images

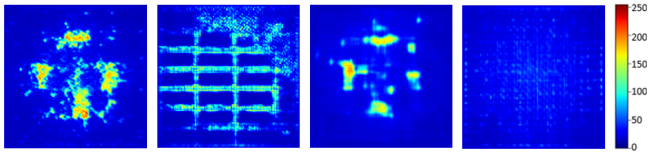

Figure 10: Comparison of generated CM images from satellite images use the satellite images (first and second) and ImageNet trained model (third and fourth) for a 45nm technology.

Table I: Comparative scores for different models (lower values imply better correlation [34], [29]).

Generated maps	ImageNet CMs			Satellite CMs		
	FID	SWD	L1	FID	SWD	L1
Set1 (7nm)	238.26	781.43	0.256	151.20	687.58	0.193
Set1 (12nm)	232.45	800.46	0.243	150.25	549.44	0.149
Set1 (45nm)	322.52	200.54	0.221	148.73	141.74	0.071
Set1 (130nm)	207.41	628.43	0.043	192.18	517.56	0.15
Set2 (45nm)	248.31	190.34	0.154	153.90	160.34	0.089

(selected from generated CM images after χ^2 test) and the set of input CM images associated with real CMs. The SWD and L1 scores are calculated by comparing the generated CM images to the highest correlated input CM image, and the mean value across all selected CM images is reported in the table.

Our first set of results trains the scale-and-shift parameters of the generator, pretrained on two datasets: ImageNet and the transformed urban satellite image dataset described in Section III. In Table I we can see that the GAN that is pretrained on satellite images consistently has a lower FID and SWD score, thus producing better quality CM images, than a GAN pretrained on ImageNet. This demonstrates that the filters learned by the GAN model on these Manhattan features in satellite images are essential to BeGAN, and the use of an arbitrary dataset yields results with fewer details. The results on four technology nodes – 7nm, 12nm, 45nm, and 130nm for "Set 1" – demonstrate that our methodology can easily be extended to different technology nodes. "Set 2" consists of designs with memory blocks, and shows the applicability of our method to memory-based designs.

Fig. 10 shows a comparative example of generated CM images with the closest correlation to the original CM images in Fig. 8. The first two CM images are generated using satellite images while the third and fourth CM images are generated using the ImageNet pretrained model. We observe that BeGAN-generated CM images have better characteristic features. This difference gets highlighted specifically between the second and fourth CMs of "Set 2" where the designs are macro dominated.

B. Obfuscation of input data

Next, we check for obfuscation provided by GAN-generated CM images using Manhattan distance metric. commonly used in Trojan detection [35] to validate substantial difference between the BeGAN-generated CM images and original CM images:

An important criterion for benchmark generation is that the circuit training dataset should be irreproducible from the generated images, i.e., the BeGAN-generated images should not allow the training data to be reconstructed exactly. Though inverting a generator is difficult, noting that a single output CM image may map to multiple latent vectors z, there are some approaches to recover latent vectors [36] used during generation. These methods work by training z to minimize $\|CM(z) - CM(z')\|$ but for the generator to be fully invertible, errors should be near-zero for all target images. Our choice of GLO as the loss function forces the generator to reconstruct the entire dataset, which adds an averaging loss and avoids overfitting to any particular training data [32].

We test for overfitting by observing the distribution of L1 Norm between generated CM image and input CM images. Fig. 11 shows the L1 Norm between the generated CM images of Set 1 in the ASAP 7nm technology node. We can see that we have more than 10% loss with a mean (μ) of 0.193 and standard deviation (σ) of 0.04, which shows considerable obfuscation of data. Based on this, we can see that the model produces diverse set of CM images and does not produce images identical to any input image. A similar distribution has been observed for other sets of generated data, and the mean values are listed in Table I.

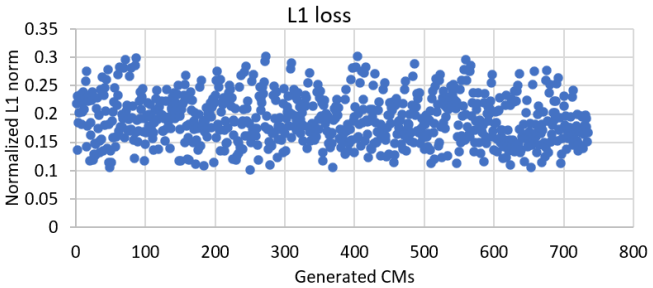

Figure 11: Distribution of L1 Norm between generated CM image and input CM image on ASAP 7nm data (μ: 0.19, σ: 0.05).

C. Significant distribution of CM Images

The input latent space for BeGAN is a 128-dimensional hyperspace with each point drawn from a Gaussian distribution and for each of these points new CM images are generated. Based on this large number of possible input distributions, BeGAN can generate a large number of CMs. If we consider a finite level of precision ϵ, the number of generated images is finite, even if it is large. For a random distribution, the level of precision is the maximum value the true mean can deviate from the sample mean estimation (p_i). Since we do not know the distribution of the generated CM images, we assume that the sample mean (p_{i-1}) represents the true mean once we move beyond a minimum threshold of data-points (we set this threshold to 500). Our stopping criterion is:

$$\Delta p_i = p_i - p_{i-1} \leq \epsilon, \quad i \geq 500 \tag{6}$$

We choose an ϵ value of 0.01 (average change pixel values by four, across 128 $\times$ 128 $\times$ 3 images) to generate thousands of CMs with acceptable coverage. This ensures sufficient variety in the benchmarks: Fig 12 shows some of the randomly generated CM images, visually showing that the BeGAN-generated CM images cover a diverse set of characteristics.

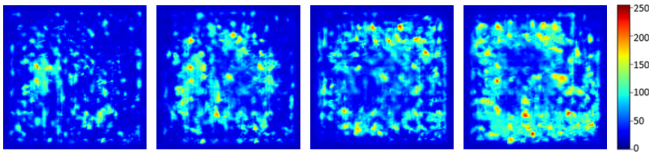

Figure 12: Randomly sampled CM images from BeGAN-generated CMs of "Set 1" GF 12nm technology.

The pretraining of the GAN and TL tuning is performed on a NVIDIA Tesla V100 GPU. The pretraining of the SNGAN model with the satellite images has been done for 450,000 iterations, and takes nine days. During transfer learning, all models are trained for 10,000 training iterations with 25 input CM images generated by flipping and rotating the original CM images. We use the Adam optimizer with initial learning rate of 0.1. During transfer learning, we update the scale and shift parameters for each batch normalization filter and the fully connected layer in the generator, accounting for a total of 16,506 parameters. The transfer learning takes two hours for training and can generate a new CM image in less than a millisecond.

V. Evaluation of BeGAN-generated PDN Benchmarks

The GAN-generated CM images are 3-channel 128×128-sized RGB images with intensity values from 0 to 255. For each GAN-generated CM, we use scaling and coarsening techniques to move to a "chip-like" current map where each pixel represents the current in a fixed size region of the generated benchmark. The scaling method translates the intensity value distributions of the GAN-generated CMs to chip current magnitudes, while the coarsening technique deals with mapping the 128×128-sized CM to a die area.

The scaling technique first converts the 3-channel input to a single channel distribution by converting the RGB CM to gray scale. Next, the CM is multiplied by a chosen current value (constrained by a minimum and maximum value specified in the technology library file) to generate current distribution of the right order for that technology. The freedom of choice in this scaling value allows this parameter to act as a tunable knob to generate diverse current intensity values from the GAN-generated image. The coarsening technique assumes an input die size and averages the current across all the pixels in that region. For example, if the die size is 512×512, each pixel in the 128×128 current map is averaged across a 4×4 area to generate CMs for a specific die-size. Note that for sufficient diversity in die sizes for each generated benchmark, we choose a random die size that lies in the range of the real CM sizes in that technology.

For each CM, in both the real circuit set and the scaled and GAN-generated set, we generate two varieties of the PDN: (i) using OpeNPDN [20], which generates an irregular power grid using predefined PDN templates, and (ii) using the OpenROAD flow [26], which builds a uniform predefined power grid. We use the thousands of GAN-generated CMs and perform PDN synthesis to generate thousands of circuits for across all the four technologies including commercial GF 12nm PDK termed as BeGAN benchmarks. We release both the real and BeGAN benchmarks publicly [24] for the open-source available ASAP 7nm, SkyWater 130nm, and FreePDK45 technologies. Each released benchmark is in SPICE format and consists of a current map, its corresponding power grid, and power bump locations for a standard flip-chip package.

A. Realism in BeGAN benchmarks

In addition to the machine learning world metrics discussed in Section IV-A, we also evaluate the BeGAN benchmarks from a VLSI world standpoint using worst-case IR drop, average IR drop, and hotspot intensities across the full-chip as metrics.

Table II summarizes the average and worst-case IR drop of a few representative samples of BeGAN benchmarks and the real benchmarks for GF12 nm and Nangate 45nm technologies.[1] The worst-case and average IR drop of the representative BeGAN benchmarks are within a small range ($\approx$ 1.0mV) of real circuit benchmarks. The numbers also are in the range of industry-standard static IR drop. (0.5%–5% of VDD). We also compare the number of *%Hotspot regions*, where a hotspot region is defined as a 5μm×5μm area on the chip where the average IR drop is greater than 4mV. On each line of the table, we show a representative BeGAN benchmark with similar IR drop and *%Hotspot* characteristics. The very small differences in the table indicate that the BeGAN-generated current maps have hotspots of similar intensities when compared to the original circuit and maintain the key features of the original data.

B. Diversity in BeGAN PDN benchmarks

Besides maintaining realism, as described in Section IV-A and V-A, in this section we highlight (i) the diversity of the PDN in BeGAN benchmarks by evaluating instance-level and PDN-node level IR drops, and (ii) the flexibility of our methodology in producing a diverse set of testcases that can challenge tool developers.

[1]IR drop statistics for the other technologies across thousands of designs generated by BeGAN are available on GitHub [24].

Table II: IR drop statistics for the original circuits with OpenROAD power grids and four representative BeGAN benchmarks generated in GF 12nm and eight representative benchmarks in FreePDK45.

Tech.	Original circuits				Randomly sampled BeGAN benchmarks			
	Design	Worst IR drop (mV)	Avg. IR drop (mV)	*% Hotspot* regions	BeGAN benchmark	Worst IR drop (mV)	Avg. IR drop (mV)	*% Hotspot* regions
12nm	AES	1.88	1.16	0	BeGAN1	3.51	2.11	0
	Dynamic_node	0.94	0.68	0	BeGAN2	2.00	1.60	0
	Ibex	4.84	3.61	20.41	BeGAN3	4.73	2.86	18.00
	JPEG	5.53	3.34	23.98	BeGAN4	5.55	2.63	18.48
45nm	AES	5.91	2.56	8.61	BeGAN5	5.62	2.74	7.32
	Swerv	2.85	1.64	0	BeGAN6	2.17	1.33	0
	Ibex	10.73	3.50	23.05	BeGAN7	9.57	4.71	29.12
	JPEG	7.90	5.18	88.45	BeGAN8	8.48	4.35	79.96
	BP_BE	4.91	2.00	2.53	BeGAN9	4.51	2.23	1.8
	BP_FE	5.14	1.81	3.46	BeGAN10	5.13	2.36	4.8
	BP	5.22	2.26	2.12	BeGAN11	5.36	2.57	1.36
	BP_multi	4.94	2.22	0.68	BeGAN12	4.52	2.28	0.69

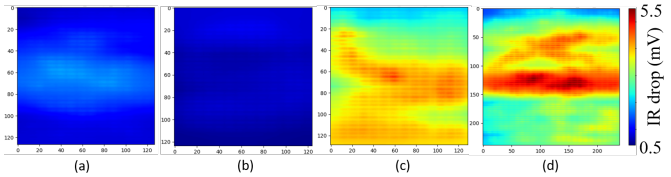

Figure 13: Instance-level IR drop heatmaps for (a) AES, (b) Dynamic_node, (c) Ibex, and (d) JPEG in GF 12nm technology.

A challenge faced by tool developers in academia and industry is the unavailability of an adequately *diverse* test suite to evaluate their solutions. In particular, this diversity has more value for supervised ML-based solutions to PDN related problems [21], [22], [37] where the model is only as good as the data it is trained on.

While global full-chip metrics such as worst-case IR drop and average IR drop across the BeGAN benchmarks are comparable to the original benchmarks (Section V-A), the detailed instance-level IR drop contours across the entire BeGAN benchmark set and the original benchmarks show significant differences such as changes in IR drop hotspot locations. Fig. 13 and Fig. 14 show the instance-level IR drop distributions of the original benchmarks and BeGAN benchmarks in 12nm technology, respectively. The differences in the contours are attributed to a unique combination of the GAN-generated CM, a synthesized PDN (regular or irregular), and a C4 bump assignment pattern in each BeGAN benchmark. We further highlight the differences in the PDN topologies across the BeGAN benchmarks by plotting per-PDN node histograms of IR drop as shown in Fig. 15. The histograms show a different distribution across the benchmarks due to a different PDN synthesized across the GAN-generated CM with different number of PDN nodes. The histograms are compared against the histograms of the original benchmarks. The figures show that the average PDN node IR drop is similar (less than 1mV difference) across benchmarks, i.e., inheriting the traits of the original benchmarks and the differences lie in the histogram distributions across the range of static IR drops, demonstrating diversity in the PDN topologies while maintaining realism.

BeGAN, as an automatic benchmark generation framework, has two knobs to increase IR drop distribution diversity of the benchmarks: (i) changing the current scaling knob based on the technology-specific minimum and maximum current constraints and (ii) changing the PDN templates which amounts to a change in PDN density.

Fig. 16 shows the difference in the per PDN node IR drop histograms of the BeGAN4 benchmark after changing (a) the current by 2× and then re-synthesizing the PDN using OpeNPDN and (b) shuffling the OpeNPDN templates. These two knobs can aid to variety in the generated benchmarks. The first knob which scales current forces OpeNPDN to select denser templates, the second knob allows a user-specified set of templates. In this case we choose sparser templates decreasing the number of PDN nodes and increasing the

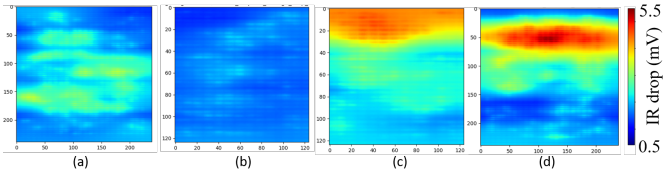

Figure 14: Instance-level IR drop heatmaps for (a) BeGAN1, (b) BeGAN2, (c) BeGAN3, and (d) BeGAN4 in GF 12nm technology.

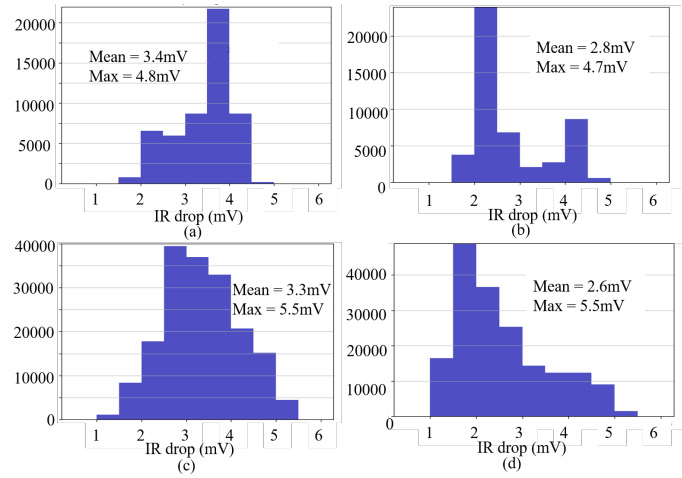

Figure 15: PDN node IR drop histograms for (a) Ibex, (b) BeGAN3, (c) JPEG, and (d) BeGAN4 benchmarks with similar mean and worst-case IR drop but different per-PDN node IR drop distributions.

worst-case IR drop from 5.5mV (Fig. 15(d)) to 8.8mV (Fig. 16).

VI. Conclusion

We propose BeGAN, a benchmark generation methodology, and demonstrate it for static PDN-related applications. The methodology overcomes the challenges involved with the unavailability of data, and IP obfuscation by using GANs and transfer learning to generate realistic current maps of any specified chip size. The automatically-generated benchmarks can aid data augmentation techniques for ML-based CAD solutions in a world with limited data, or can be used as a standalone test set to evaluate solutions related to PDN synthesis and analysis problems. The benchmarks are released on GitHub [24].

Acknowledgments: This work was supported in part by the DARPA IDEA program, the Louise Dosdall Fellowship, and by a University of Minnesota Doctoral Dissertation Fellowship. We would like to thank the Cadence Voltus team in Austin TX, especially Albert Zeng and Chao Jiao, for pointing out and clarifying the problem.

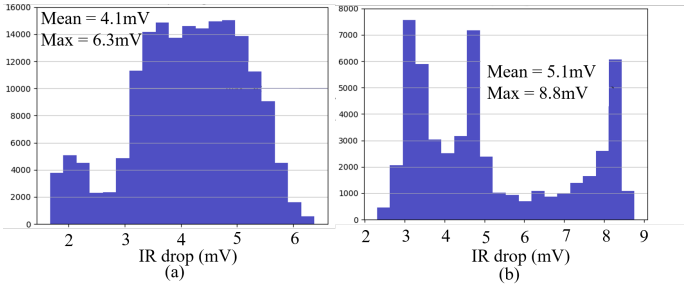

Figure 16: PDN node IR drop histograms for BeGAN with (a) scaled current and resynthesized PDN, and (b) changed OpeNPDN templates.

References

[1] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a targeted translator in FORTRAN," in *ISCAS*, 1985. (Distributed by magnetic tape to attendees of the special session, "Recent algorithms for gate-level ATPG for fault simulation and their performance assessment.").

[2] F. Brglez, *et al.*, "Combinational profiles of sequential benchmark circuits," in *Proc. ISCAS*, pp. 1929–1934, 1989.

[3] F. Corno, *et al.*, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test*, vol. 17, pp. 44–53, Jul/Sep 2000.

[4] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0," tech. rep., MCNC Technical Report, 1991.

[5] C. Albrecht, "IWLS 2005 benchmarks," 2005. iwls.org/iwls2005/benchmark_presentation.pdf.

[6] G.-J. Nam, *et al.*, "The ISPD2005 placement contest and benchmark suite," in *Proc. ISPD*, pp. 216–220, 2005.

[7] G.-J. Nam, *et al.*, "The ISPD global routing benchmark suite," in *Proc. ISPD*, pp. 156–159, 2008.

[8] W.-H. Liu, *et al.*, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proc. ISPD*, 2019.

[9] S. Nassif, "Power grid analysis benchmarks," in *Proc. ASP-DAC*, pp. 376–381, 2008.

[10] "IBM power grid benchmarks." web.ece.ucsb.edu/~lip/PGBenchmarks/ibmpgbench.html.

[11] Z. Li, *et al.*, "2012 TAU power grid simulation contest: Benchmark suite and results," in *Proc. ICCAD*, pp. 478–481, 2012.

[12] R. Aitken, "Time to retire our benchmarks," *IEEE Des. Test*, p. 88, 2010.

[13] I. J. Goodfellow, *et al.*, "Generative adversarial networks," in *Proc. NeurIPS*, pp. 2672–2680, 2014.

[14] A. Radford, *et al.*, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arxiv:1511.06434*, 2016.

[15] T. Karras, *et al.*, "A style-based generator architecture for generative adversarial networks," in *Proc. CVPR*, pp. 4396–4405, 2019.

[16] T. Karras, *et al.*, "Analyzing and improving the image quality of stylegan," in *Proc. CVPR*, pp. 8107–8116, 2020.

[17] A. Noguchi and T. Harada, "Image generation from small datasets via batch statistics adaptation," in *Proc. ICCV*, pp. 2750–2758, 2019.

[18] M. Zhao, *et al.*, "On leveraging pretrained GANs for limited-data generation," in *Proc. ICML*, pp. 11340–11351, 2020.

[19] P.-W. Luo, *et al.*, "Benchmarking for research in power delivery networks of three-dimensional integrated circuits," in *Proc. ISPD*, pp. 17–24, 2013.

[20] V. A. Chhabria, *et al.*, "Template-based PDN synthesis in floorplan and placement using classifier and CNN techniques," in *Proc. ASP-DAC*, pp. 44–49, 2020.

[21] V. A. Chhabria, *et al.*, "Thermal and IR drop analysis using convolutional encoder-decoder networks," in *Proc. ASP-DAC*, pp. 690–696, 2021.

[22] Z. Xie, *et al.*, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. ASP-DAC*, pp. 13–18, 2020.

[23] Y.-C. Fang, *et al.*, "Machine-learning-based dynamic IR drop prediction for ECO," in *Proc. ICCAD*, pp. 1–7, 2018.

[24] "BeGAN-benchmarks." github.com/UMN-EDA/BeGAN-benchmarks.

[25] Y. Wang, *et al.*, "Transferring GANs: generating images from limited data," in *Proc. ECCV*, pp. 220–236, 2018.

[26] "The OpenROAD project." github.com/The-OpenROAD-Project.

[27] "SkyWater 130nm PDK." github.com/google/skywater-pdk.

[28] L. T. Clark, *et al.*, "ASAP7: A 7-nm FinFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, July 2016.

[29] T. Karras, *et al.*, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. ICLR*, 2018.

[30] A. Brock, *et al.*, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. ICLR*, 2019.

[31] T. Miyato, *et al.*, "Spectral normalization for generative adversarial networks," in *Proc. ICLR*, 2018.

[32] P. Bojanowski, *et al.*, "Optimizing the latent space of generative networks," in *Proc. ICML*, pp. 600–609, 2018.

[33] J. Johnson, *et al.*, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. ECCV*, pp. 694–711, 2016.

[34] M. Heusel, *et al.*, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. NeurIPS*, pp. 6629–6640, 2017.

[35] K. Hu, *et al.*, "High-sensitivity hardware trojan detection using multimodal characterization," in *Proc. DATE*, pp. 1271–1276, 2013.

[36] Z. C. Lipton and S. Tripathi, "Precise recovery of latent vectors from generative adversarial networks," in *ICLR Workshop*, 2017.

[37] V. A. Chhabria, *et al.*, "MAVIREC: ML-aided vectored IR-drop estimation and classification," in *Proc. DATE*, pp. 1825–1826, 2021.