High-Definition Routing Congestion Prediction for Large-Scale FPGAs

Mohamed Baker Alawieh
ECE Department, UT Austin
mohdbaker@utexas.edu

Wuxi Li
ECE Department, UT Austin
wuxi.li@utexas.edu

Yibo Lin
ECE Department, UT Austin
yibolin@utexas.edu

Love Singhal
Intel Corporation
love.singhal@intel.com

Mahesh A. Iyer
Intel Corporation
mahesh.iyer@intel.com

David Z. Pan
ECE Department, UT Austin
dpan@ece.utexas.edu

ABSTRACT

To speed up the FPGA placement and routing closure, we propose a novel approach to predict the routing congestion map for large-scale FPGA designs at the placement stage. After reformulating the problem into an image translation task, our proposed approach leverages recent advancement in generative adversarial learning to address the task. Particularly, state-of-the-art generative adversarial networks for high-resolution image translation are used along with well-engineered features extracted from the placement stage. Unlike available approaches, our novel framework demonstrates a capability of handling large-scale FPGA designs. With its superior accuracy, our proposed approach can be incorporated into the placement engine to provide congestion prediction resulting in up to 7% reduction in routed wirelength for the most congested design in ISPD 2016 benchmark.

1 INTRODUCTION

The ceaseless down scaling of integrated circuit (IC) technologies continues to drive, as a byproduct, an up scale in the challenges and complexity associated with physical design. This in practice translates to extended design closure time, as multiple expensive design iterations are needed before converging at a final physical scheme, especially for large-scale designs.

To address this challenge, research has focused lately on developing predictive models that can make available, at early stages of the design flow, useful predictions about later stages. Particularly, in modern Field-Programmable Gate Array (FPGA) place and route flows, leveraging routing congestion information during the placement step has demonstrated significant performance improvement [1–4]. Thus, it is of vital importance to develop accurate routing congestion prediction models for large-scale FPGA designs.

Recently, advancements in machine learning have revolutionized almost every field of research by introducing a far-reaching data-driven perspective for problem solving, and electronic design automation (EDA) is no exception here. In EDA, machine learning applications span different tasks [5]. The main drive for this wide adoption is the exceptional speedup associated with machine learning techniques which in turn translates into faster design closure and better physical design quality.

Relevant to our work are the techniques proposed for routing congestion prediction [6, 7]. In [6], design rule checking (DRC) violations after detailed routing are predicted based on input placement solutions. The work presents two modes: 1) prediction of the total number of DRC violations with placement information only; 2) prediction of DRC hotspots with placement and global routing solutions. The major application of [6] is to guide detailed placement for the mitigation of local DRC violations after detailed routing, while it is not designed for the compliance of global routing congestion. On the other hand, the work in [7] proposes predicting the routing congestion maps for FPGA designs. It relies on features extracted from both placement and global routing schemes to perform the task. Despite the fact that such a setup can help accelerate the detailed routing process, it is of greater impact to predict the full routing congestion map with features exclusively obtained from the placement stage. Knowing that state-of-the-art FPGA placement engines are iterative, such a scenario can further speedup the design closure and improve routing quality by incorporating routing information into the placement engine.

This issue was addressed in the latest work on FPGA routing congesting prediction in [8] where predictions are made based on features extracted from the placement stage solely. In [8], a conditional generative adversarial network is used to model the routing congestion prediction as an image translation task. This model architecture, namely *pix2pix*, has been adopted recently to address different tasks in EDA [8–11]. While the scheme proposed in [8] is designed to provide routing predictions at placement stage, the proposed model and its associated features are ill-equipped to handle large-scale FPGA designs such as those in ISPD 2016 benchmark [12].

In fact, the approach presented in [8] is not capable of scaling to large designs due to three main limitations. The first is inherent in the *pix2pix* model architecture which has limited resolution and thus, poses a limit on the design size [13]. Additionally, the features extracted at the placement stage in [8] depreciate with the increase in design scale. In particular, the connectivity information is incorporated in the input as a connectivity map with flying lines. While such a map is expressive in small and uncongested designs with limited connectivity, it becomes obsolete with large and dense designs. Hence, with connectivity information distorted, the quality of the congestion prediction is expected to degrade. Moreover, the feature maps used in [8] are based on the VTR academic software [14] that cannot handle industrial-size designs.

To address these limitations, we propose a new placement-based routing congestion prediction approach for large-scale FPGA designs. Our proposed approach adopts a new conditional generative adversarial network (CGAN) model, namely *pix2pixHD*, which performs high-definition (*HD*) image translations for large images with high resolutions [15]. With *HD* image translation, our approach can achieve high prediction accuracy for large FPGA designs while relying on well-engineered features that can encode the placement and connectivity information for large-scale designs. Moreover, when substituting the congestion prediction used in state-of-the-art congestion aware placement engine, our prediction scheme results in better placement quality and achieves up to 7% reduction in routed wirelength for the most congested design in ISPD 2016 benchmark.

Our main contributions are summarized as follows:

- We cast the routing congestion problem as a *high-definition* image-to-image translation task where features from placement stage are used to estimate congestion map for large-scale FPGA designs.
- A new CGAN for high-definition image-translation, namely *pix2pixHD*, is adopted to predict FPGA congestion.
- We propose using well-engineered features extracted from the placement stage that are capable of encoding placement and connectivity information for large-scale designs.
- Our proposed framework achieves superior modeling performance compared to state-of-the-art FPGA congestion map prediction methods [4, 8] when using advanced image similarity evaluation metrics.
- Incorporating our approach into a placement engine results in up to 7% reduction in routed wirelength compared with the widely adopted congestion prediction method [4].

978-1-7281-4123-7/20/$31.00

The remainder of this paper is organized as follows. In Section 2, we present the problem formulation and the evaluation metrics used in our experiments. Then, the details of our proposed large-scale FPGA routing congestion prediction approach are shown in Section 3. Section 4 presents experimental results demonstrating the efficacy of our method, and conclusions are presented in Section 5.

2 BACKGROUND

2.1 Problem Description

In the FPGA physical design process, placement and routing are the two major stages that map a circuit description into the physical layout. Typically, such a process is iterative and requires multiple rounds of place and route (PnR) before converging to the final layout. Conventionally, the placement task was performed without taking routing requirements and behavior into consideration. However, state-of-the-art FPGA placement engines consider routability as one of the metrics governing the placement process [1–3]. With primitive routability estimation methods, fewer iterations are needed to get a clean design; thus a shorter design closure time is experienced.

FPGA routing congestion map prediction task aims to take the benefit from such inter-stage information passing into a next level by predicting the entire routing congestion map of the design based on information at the placement stage. This has many practical applications including placement strategy selection and placement adjustment to mitigate routability issues.

Precisely, the task is to train a model that is capable of estimating the routing congestion map for large-scale FPGA designs. As an input, the model takes a design netlist and a placement solution and it generates a congestion map prediction as an output. Technically, a good way to preserve spatial relations in the placement scheme is to encode features as an image. Such an approach was adopted in [6, 8] where input features are encoded into an image, and the desired output is generated in an image format as well. In one of the approaches proposed in [6], a set of features is first extracted from both the placement and global routing stages and then mapped to different layers of an input image. Next, this image is inputted to a trained model to predict an image showing the location of DRC hotspots in the design. While this approach targets detecting highly congested regions through DRC hotspot detection, it does not have the objective of predicting the complete routing congestion map. Besides, while no global routing information is needed to predict the total number of DRC hotspots in [6], predicting the DRC hotspot image, which reflects congestion locations, requires the global routing information. Thus, the work in [6] does not address our problem formulation in this work: estimating the complete routing congestion map based on placement stage information.

On the other hand, the work in [8] addresses precisely this objective by mapping the problem into an image translation task. In this task, placement features are mapped into an input image and a model is trained to predict the complete routing congestion map as an output image. However, this work is ill-equipped to handle industrial-size FPGA designs. Besides having a limited prediction resolution (256×256 pixels), the features used to represent placement and netlist information are not adequate for large-scale designs. For the connectivity information, it relies on *flying lines* to encode the connections in the FPGA design as shown in Figure 1(a). Clearly, this representation becomes obsolete for large-scale FPGA designs with over 700K nets. Figure 1(b) shows the connectivity representation with only 5K nets included, which, although complicated, is still comprehensible. However, including the entirety of 700K nets results in Figure 1(c) which is simply a blacked-out image with no useful information since all pixels have a value equal to zero (black color). Besides, keeping in mind the rectilinear nature of routing, *flying lines* representation fails to reflect an accurate estimate of routing demand at different locations in large designs. In addition, the image representation for the placement in [8] is based on the VTR academic software [14] that cannot handle industrial-size designs. Hence, such features limit the applicability of the work in [8] to small FPGA designs only.

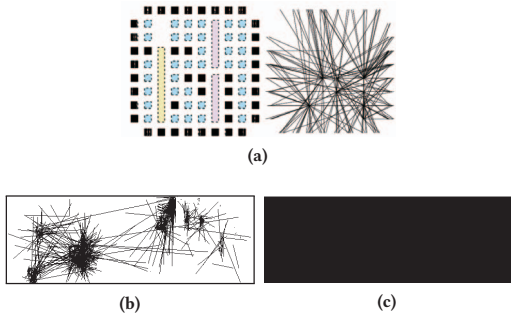


Figure 1: An example of connectivity representation used in [8] for a small design is shown in (a). (b) and (c) show the results of using the same representation for a large design with over 700K nets where (b) shows only 5000 nets and (c) demonstrates the failure in handling all 700K nets.

In this work, we propose a routing congestion prediction approach for large-scale FPGA designs which relies on a high-definition image translation framework paired with well-engineered feature encoding.

2.2 Evaluation Metrics

With the problem formulated as an image translation task, adequate image similarity metrics are needed to judge upon the quality of the results. Here, we present the different evaluation metrics used in our experiments to evaluate the proposed approach and compare it with state-of-the-art approaches.

Two pixel-level metrics, namely normalized root-mean-square-error and pixel accuracy, are used to evaluate the prediction. In addition, we propose using two image similarity metrics that are relevant to the FPGA routing congestion estimation task. The first is the structural similarity index that can capture local congestion clusters that possess spatial structures [16]. On the other hand, earth mover's distance is used as another metric to assess the difference in the pixel distribution between the golden and predicted images [17, 18] .

Given an FPGA architecture of size $H \times W$, images $\hat{Y}$ and Y with size $H \times W$ and range $[0, 255]$ representing a predicted routing congestion map and its corresponding golden map, the details of the used metrics are presented below.

Definition 1 (Normalized Root-Mean-Square-Error - NRMS)**.** NRMS is defined as the normalized root mean square pixel difference between $\hat{Y}$ and Y. Mathematically, it can be expressed as:

$$\text{NRMS} = \frac{\sqrt{\sum_{i=1}^{H}\sum_{j=1}^{W}(y_{i,j} - \hat{y}_{i,j})^2}}{(y_{max} - y_{min}) \cdot (H \times W)}. \tag{1}$$

Definition 2 (Pixel Accuracy - PIX)**.** Pixel accuracy is defined as the normalized pixel-level difference between $\hat{Z}$ and Z, the predicted congestion map and the golden one after exposure normalization. Mathematically, this can be expressed as:

$$\text{PIX} = \frac{\sum_{i=1}^{H}\sum_{j=1}^{W}|z_{i,j} - \hat{z}_{i,j}|}{255 \cdot (H \times W)}. \tag{2}$$

Definition 3 (Structural Similarity Index - SSIM)**.** Structural Similarity Index is a perception-based metric that captures changes in the structural information between images. It can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality [16]. Mathematically, over a $k \times k$ window of the images $\boldsymbol{Y_k}$ and $\hat{\boldsymbol{Y}}_{\boldsymbol{k}}$, SSIM can be expressed as:

$$\text{SSIM}_k = \frac{(2\mu_{\boldsymbol{Y_k}}\mu_{\hat{\boldsymbol{Y}}_{\boldsymbol{k}}} + c_1) \cdot (2\sigma_{\boldsymbol{Y_k},\hat{\boldsymbol{Y}}_{\boldsymbol{k}}} + c_2)}{(\mu^2_{\boldsymbol{Y_k}} + \mu^2_{\hat{\boldsymbol{Y}}_{\boldsymbol{k}}} + c_1) \cdot (\sigma^2_{\boldsymbol{Y_k}} + \sigma^2_{\hat{\boldsymbol{Y}}_{\boldsymbol{k}}} + c_2)}, \tag{3}$$

where $\mu_{\boldsymbol{Y_k}}$ is the pixel averages over window $\boldsymbol{Y_k}$ and $\sigma^2_{\boldsymbol{Y_k}}$ is the corresponding variance, $\sigma_{\boldsymbol{Y_k},\hat{\boldsymbol{Y}}_{\boldsymbol{k}}}$ is the covariance of $\boldsymbol{Y_k}$ and $\hat{\boldsymbol{Y}}_{\boldsymbol{k}}$, and c_1 and c_2 are two terms used to ensure division stability. Intuitively, the value

of $SSIM_k$ is high when the pixel averages in the two windows are close and the covariance between them is high.

Definition 4 (Earth Mover's Distance - EMD)**.** EMD is the minimum amount of work to match the pixel distributions of Y and $\hat{Y}$, normalized by the total weight of the lighter distribution. It is widely used in content-based image retrieval to compute distances between the color histograms of two images. EMD's details are omitted due to page limit. Reader is referred to [17, 18] for details.

3 CONGESTION MAP PREDICTION

3.1 Feature Extraction

To leverage the impressive success of conditional generative adversarial networks for FPGA routing congestion prediction, this task should be cast as image translation where the objective is to map an image from the feature domain to the output domain. In the output domain, the desired congestion map can be directly viewed as a Red-Green-Blue (RGB) image with 2 non-zero channels representing the vertical and horizontal routing congestion as shown in Figure 2 where the vertical and horizontal congestion maps are mapped to the green and red channels respectively, with the blue channel set to zero. Besides, to ensure the generalization of the learned model, the dataset is normalized such that the highest congestion level in the data is mapped to pixel level 255.

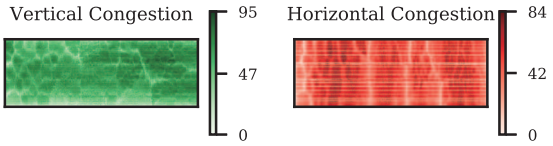


Figure 2: Sample encoded routing congestion map is shown. Vertical congestion is mapped to the green channel, while the horizontal one is mapped to the red.

On the other hand, feature representation in the input space is not trivial. In practice, two types of information need to be encoded in the input image: (i) the placement scheme and (ii) the connectivity information. While both types are present in the placement results and the netlist, our task is to map them into an image format with minimal loss in information relevant to the routing procedure. For the placement information, pin density reflects an accurate representation of the placement scheme. Besides, it provides an insight into the routing congestion since regions with higher pin density are more likely to suffer from congestion issues when routing. Therefore, we use pin density as the first feature which mainly captures the necessary placement information and encode it on the blue channel of the input image as shown in Figure 3(a).

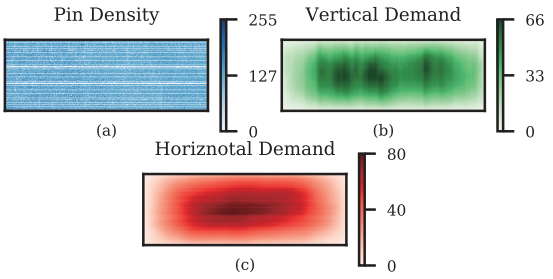


Figure 3: Sample encoded feature map is shown. Pin density is mapped to the blue channel, vertical demand is mapped to green, and the horizontal is mapped to red.

As for the connectivity information, we propose a systematic approach to encode FPGA connectivity that, while reflecting the routing demand, is capable of handling large-scale FPGA designs. Towards this end, we adopt a routing demand estimation framework analogous to those proposed in [4, 19]. The key idea is to compute, for each net, the probability of it being routed on each of the vertical and horizontal grids within its bounding box.

To elaborate on this, we consider the simple example shown in Figure 4. For a given net, the bounding box of its pins (red circles) is first obtained where x_{net} and y_{net} represent the dimensions of the bounding box. Horizontally, the net has a probability of $1/y_{net}$ to be routed in a particular grid inside the bounding box along the $x-$axis. Similarly, it has a probability of $1/x_{net}$ to be routed on each vertical grid along the $y-$axis. Thus, the horizontal demand map is incremented by $1/y_{net}$ at each location in the blue shaded region. Similarly, the vertical demand map is incremented by $1/x_{net}$ at each location in that region. This routine is applied for all nets in the design to get the vertical and horizontal demand maps. The vertical and horizontal demand maps are encoded on the green and red channels of the input image as shown in Figure 3(b) and (c) respectively after applying a normalization step similar to that used for the output.

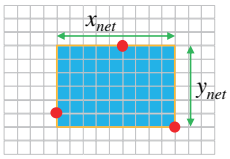


Figure 4: A routing demand computation example is shown.

At the end of this process, we can obtain the 3-channel input image carrying pin density information and the vertical and horizontal routing demand maps. On the other hand, the output image is formed by encoding the vertical and horizontal congestion maps on two channels of the image. An example normalized input/output RGB image pair is shown in Figure 5 where the input and output images are presented in Figure 5(a) and Figure 5(b) respectively.

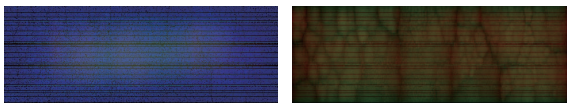
3.2 HD Image Translation Model

With an adequate training dataset encoded as demonstrated in Section 3.1, a CGAN model designed for high-definition image translation can be trained to predict congestion maps for large-scale FPGA designs. Recently, CGAN model *pix2pix* [13] has been adopted in EDA to address several challenges [8, 9]. A CGAN takes an input image in one domain and tries to generate a mapped output in another domain. Typical examples include image colorization and aerial to map and edge to photo translations.

Practically, a CGAN is composed of two main components: the *generator* and the *discriminator*. The generator G is trained to produce images in the output domain, based on an input image in another domain, 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".

Despite its success in the aforementioned tasks, *pix2pix* falls short of addressing the routing congestion prediction task for large-scale FPGA designs due to its limited resolution (256×256). Such constraint limits the size of FPGA designs the method can handle. To address this limitation, we propose using a new *pix2pixHD* model developed for high-definition image translation tasks [15]. With an output resolution reaching 4096×2048, *pix2pixHD* fits well for large-scale FPGA routing congestion task. To enable such a high-definition image generation, *pix2pixHD* introduces three main features: (i) a coarse-to-grain generator, (ii) a multi-scale discriminator, and (iii) a robust adversarial learning objective function [15].

3.2.1 Generator. The generator is decomposed into two sub networks: a global generator (G_1) and a local enhancer (G_2). As shown in Figure 6, the global generator consists of three components: a convolutional



(a) (b)

Figure 5: An example input/output image pair is shown.

front-end $G_{1,C}$, a set of residual blocks $G_{1,R}$ [20], and a deconvolutional back-end $G_{1,D}$ [15, 21]. The local enhancer has a similar structure, however, it is designed to generate images with double the resolution of the global generator. Moreover, as shown in Figure 6, the input to the residual block $G_{2,R}$ is the sum of two feature maps: the output feature map of $G_{2,C}$ and the last feature map from the back-end of the global generator $G_{1,D}$ [15]. This is intended to help integrate the global information from G_1 to G_2.

During training, the global generator is trained first and then the local enhancer. Finally, we jointly fine-tune the entire generator together. The architectural details for $G_{1,C}$ and $G_{1,D}$ in the global generator are shown in Tables 1 and 2 respectively. As for $G_{1,R}$, it is formed of nine residual blocks each having two 3×3 convolutional layers with 1024 filters [20]. Similarly, $G_{2,R}$ in the local enhancer is composed of three such residual blocks with 64 filters instead of 1024, and the details of $G_{2,C}$ and $G_{2,D}$ are summarized in Table 3.

3.2.2 Discriminator. For high-resolution image generation, the discriminator needs to have a large receptive field [15]. For that, a multi-scale discriminator, with three discriminators (D_1, D_2 and D_3) that operate at different image scales, is used [15]. Both the real and synthesized images are downsampled by a factor of 2 and 4 to create an image pyramid of 3 scales. With this structure, the three discriminators are trained to differentiate real and synthesized images at the 3 different scales. The discriminator operating at the coarsest scale has the largest receptive field with a global view. On the other hand, the one at the finest scale encourages finer details in generated images. The 3 discriminators have the same structure where a 70×70 PatchGAN network is used [13] with 4 convolutional layers each having a filter size of 4×4 and with 64, 128, 256 and 512 filters respectively.

3.2.3 Training. With the three discriminator networks, the conventional GAN objective function can be expressed as [13, 15]:

$$\min_{G} \max_{D_1, D_2, D_3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k). \tag{4}$$

where

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_k) = \mathbb{E}_{x,y}[\log D_k(x, y)] \\ + \mathbb{E}_x[\log(1 - D_k(x, G(x)))]. \end{aligned} \tag{5}$$

To improve this loss, a feature mapping loss based on the discriminator is added to push the generator towards producing natural statistics at multiple scale [15]. Specifically, we extract features from multiple layers of the discriminator and learn to match these intermediate representations from the real and the synthesized image [15]. Mathematically, the feature matching loss can be expressed as:

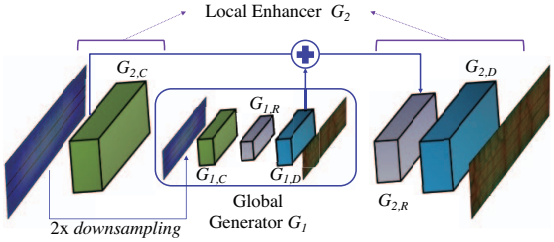


Figure 6: The the generator in the *pix2pixHD* model is shown. It comprises two components, a global generator G_1 and a local enhancer G_2.

Table 1: The details of $G_{1,C}$ are shown.

Lyr	Filter	# Filters	Str
1	7×7	64	1
2	3×3	128	2
3	3×3	256	2
4	3×3	512	2
5	3×3	1024	2

Table 2: The details of $G_{1,D}$ are shown.

Lyr	Filter	# Filters	Str
1	3×3	512	1/2
2	3×3	256	1/2
3	3×3	128	1/2
4	3×3	64	1/2
5	7×7	3	1

Table 3: The details of G_2 are shown.

Component	Layer	Filter	# Filters	Str
$G_{2,R}$	1	7×7	32	1
	2	3×3	64	2
$G_{2,D}$	1	3×3	64	1/2
	2	7×7	3	1

$$\mathcal{L}_{FM}(G, D_k) = \mathbb{E}_{x,y} \sum_{i=1}^{T} ||D_k^{(i)}(x, y) - D_k^{(i)}(x, G(x))||_1 \tag{6}$$

where T is the total number of layer and $D_k^{(i)}$ denotes the output feature map of the discriminator D_k at the $i-$th layer. This feature matching loss is related to the perceptual loss [21], which has been shown to be useful for style transfer application [13]. The final loss function is the sum of Equations (5) and (6) in addition to a VGG-loss term added to enhance accuracy [15, 21].

4 EXPERIMENTAL RESULTS

In our experiments, we use ISPD 2016 contest benchmark [12] with 480×168 device size. Compared to VTR benchmark [14] used in previous work [8] with less than 12K cells, 2016 contest benchmark [12] contains large designs with up to 1.1M cells as shown in Table 4.

4.1 Data Preparation and Training Setups

For each benchmark, 200 different random net weighting schemes are obtained, then the corresponding placement results are generated using elfPlace [1]. Then, routing is performed using NCTU-GR [22] to obtain the golden routing congestion maps. As a first step, the feature extraction process described in Section 3.1 is performed to prepare both the input and output images for the learning task. Then, two evaluation setups are performed. In Setup 1, 11 out of the 12 designs are used for training, while the 12^{th} design is used for testing. For each design d, testing is performed using a model that has seen all designs except for d. This setup is intended to demonstrate the generalization capability of the model outside the training dataset. While in Setup 2, the data corresponding to each particular design is split into 80% used for training and 20% used for testing. With this setup, the models see only the design of interest during training; hence, they are design specific and can be used to provide routing congestion information within the PnR iterations for the given design.

4.2 Performance Evaluation

We use the evaluation metrics introduced in Section 2.2 to compare our approach to the following existing ones:

RUDY [4]: a model-based congestion estimation. It was introduced in Section 3.1 as one of the features used in our approach.

pix2pix [8, 13]: a machine learning approach based on the model proposed in [8]. However, as shown in Section 2.1, the connectivity feature used in [8] fails for large-scale designs, besides, the model can only handle designs with size up to 256×256. Thus, to enable this comparison, two adjustments were done. The first is that images are scaled down to 256×256 and the CGAN model proposed in [8] is used to predict congestion maps. In addition, we use the features proposed in our work instead of those used in [8] which are not applicable for large-scale designs. Hence, this approach can be summarized as using features proposed in this work with the learning model proposed in [8] along with proper image scaling.

4.2.1 Setup 1. With Setup 1, Figures 7 and 8 show the resulting congestion maps corresponding to two samples from designs FPGA-2 and FPGA-8, respectively. Both vertical and horizontal golden routing congestion maps are shown in addition to the prediction results from the proposed approach (Proposed), *pix2pix*, and RUDY. Evidently, one can visually notice that the proposed approach can produce the best prediction results. To quantify this superior performance, Tables 5 and 6 show a comparison of the evaluation metrics across the different approaches for both vertical and horizontal congestion respectively. In these tables, the visual observation is validated numerically with the proposed approach out-performing other approaches in all evaluation metrics, on average.

4.2.2 Setup 2. Compared to Setup 1, models in Setup 2 are design specific, hence, they are expected to generate better results. This is validated in Table 7 where a comparison between Setup 1 and Setup 2

Table 4: The ISPD 2016 benchmark details are shown.

Benchmark	#LUT	#FF	#RAM	#DSP	#Ctrl Set
FPGA-1	50K	55K	0	0	12
FPGA-2	100K	66K	100	100	121
FPGA-3	250K	170K	600	500	1281
FPGA-4	250K	172K	600	500	1281
FPGA-5	250K	174K	600	500	1281
FPGA-6	350K	352K	1000	600	2541
FPGA-7	350K	355K	1000	600	2541
FPGA-8	500K	216K	600	500	1281
FPGA-9	500K	366K	1000	600	2541
FPGA-10	350K	600K	1000	600	2541
FPGA-11	480K	363K	1000	400	2091
FPGA-12	500K	602K	600	500	1281
Resources	538K	1075K	1728	768	N/A

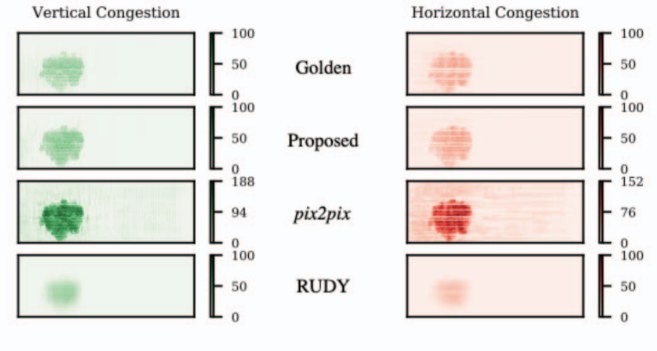


Figure 7: A sample congestion prediction result is shown for FPGA-2.

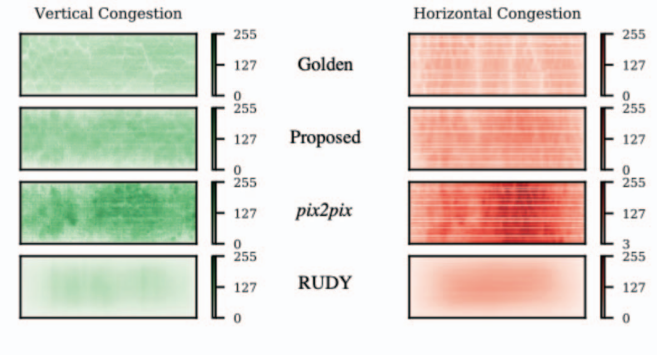


Figure 8: A sample congestion prediction result is shown for FPGA-8.

using the proposed approach is shown. In the table, only the average evaluation over the 12 benchmarks is reported due to space limit.

4.3 Applications

Our proposed routing congestion prediction approach has several applications; here, we present two important ones.

Routability-driven placement: our prediction framework is used within the placement engine so that congestion can be accounted for during the placement process. In the original implementation of elfPlace [1], which is considered among the state-of-the-art routability-aware placement engines, RUDY [4] is used to predict congestion during the placement process. Here, we use our proposed approach instead and compare the results with the original case with RUDY under two scenarios. In the first scenario, the original routing capacity of the FPGA board is assumed. Table 8 summarizes the results showing routed wirelength (Rtd WL) when RUDY and our proposed approach are used for congestion prediction during placement under full routing capacity. Knowing that designs FPGA-5, 7 and 11 are the congested ones, it is clear that our proposed approach can improve the placement quality to achieve better routed wirelength. Besides, design FPGA-5 is by far the most congested and our proposed routing congestion prediction can achieve 7% reduction in routed wirelength for this design.

On the other hand, and to further demonstrate the efficacy of our proposed approach, the routing capacity is reduced to 88% its original value in an attempt to push more designs towards congestion. Since FPGA-5 is highly congested, it is excluded from this scenario because any small reduction in routing capacity can make the design unroutable. The new routing capacity (88% of original capacity) is chosen such that all placements in the training dataset, excluding FPGA-5, are routable. Also shown in Table 8 are the routed wirelength results for this scenario where FPGA-11 is now the most congested design. As shown in the table, our proposed approach can achieve better results for the congested designs with more than 1% reduction in wirelength on FPGA-11. It is important to note that FPGA-4 and FPGA-12 experienced minor wirelength increases in scenarios 1 and 2 respectively. However, it is clear that our proposed approach does not have inherent limitation in handling specific design since FPGA-4 and FPGA-12 did not experience performance degradation in scenarios 2 and 1 respectively; instead, FPGA-4 experienced an improvement under scenario 2. These results show that our predictions have better correlation with the golden solution when compared to RUDY which enables improved placement quality resulting in better wirelength during routing. Moreover, since the prediction time for RUDY and and our proposed approach is negligible compared to the placement time, as will be shown in Section 4.4, the overall placement time is not affected by the prediction method used.

Placement quality ranking: On the other hand, routing congestion prediction can be used to rank placement quality when different placements are available. Instead of routing all placements, those with high congestion can be eliminated immediately. Table 9 compares the number of matches among the 10% most congested placements between the golden results and different prediction methods. As shown in the figure, our predictions can achieve the best average matching.

4.4 Runtime Comparison

Based on the aforementioned performance evaluation, it is clear that our proposed approach is the most suited to handle the routing congestion prediction task for large designs. It can be used for placement enhancement and ranking which can significantly improve routing quality. Instead of running the complete routing scheme to get the congestion information, the prediction model can provide this information at the placement stage. Table 10 compares the runtime for the prediction models to that of running the routing flow (denoted NCTU). The proposed approach can achieve up to **5000**× and **51**× speedup when using GPU and CPU-only respectively. More importantly, the prediction time is constant and is independent of the design size.

5 CONCLUSION

In this work, we propose a novel routing congestion map prediction framework for large-scale FPGA designs at the placement stage. It starts by casting the problem as a high-definition image translation task, then uses state-of-the-art *HD* image translation models. The proposed framework uses as an input well-engineered features representing the placement scheme and design connectivity for large-scale designs which are encoded on different channels of the input image. Our proposed approach demonstrates superior performance in terms of evaluation metrics used when compared to existing approaches. Moreover, its incorporation in the placement step results in up to 7% reduction in wirelength.

6 ACKNOWLEDGMENT

This work is supported in part by Intel Strategic Research Alliance.

REFERENCES

[1] W. Li, Y. Lin, and D. Z. Pan, "elfPlace: Electrostatics-based placement for large-scale heterogeneous fpgas," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[2] R. Pattison, Z. Abuowaimer, S. Areibi, G. Grewal, and A. Vannelli, "GPlace: A congestion-aware placement tool for ultrascale fpgas," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–7.

[3] G. Chen, C.-W. Pui, W.-K. Chow, K.-C. Lam, J. Kuang, E. F. Young, and B. Yu, "RippleFPGA: routability-driven simultaneous packing and placement for modern fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 10, pp. 2022–2035, 2017.

[4] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2007, pp. 1226–1231.

[5] Y. Lin, M. B. Alawieh, W. Ye, and D. Pan, "Machine learning for yield learning and optimization," in *IEEE International Test Conference (ITC)*, 2018.

Table 5: A comparison between different estimation methods is shown for the vertical congestion prediction under Setup 1.

	NRMS (lower better)			SSIM (higher better)			PIX (lower better)			EMD ($\times 10^{-2}$) (lower better)		
Design	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed
FPGA-1	0.079	0.844	0.077	0.931	0.195	0.952	0.047	0.318	0.014	0.043	0.58	0.02
FPGA-2	0.107	1.049	0.089	0.885	0.672	0.912	0.037	0.101	0.048	0.052	0.156	0.05
FPGA-3	0.178	1.487	0.467	0.679	0.418	0.714	0.093	0.087	0.066	0.129	0.307	0.174
FPGA-4	0.27	0.664	0.17	0.578	0.447	0.777	0.096	0.089	0.056	0.164	0.224	0.107
FPGA-5	0.388	0.124	0.34	0.514	0.629	0.644	0.086	0.069	0.054	0.196	0.103	0.181
FPGA-6	0.205	1.01	0.276	0.566	0.371	0.681	0.132	0.102	0.093	0.141	0.322	0.191
FPGA-7	0.271	0.369	0.186	0.477	0.521	0.705	0.112	0.094	0.083	0.189	0.111	0.184
FPGA-8	0.321	0.773	0.211	0.55	0.446	0.689	0.209	0.151	0.131	0.158	0.339	0.168
FPGA-9	0.329	0.562	0.279	0.521	0.473	0.075	0.207	0.112	0.232	0.175	0.088	0.15
FPGA-10	0.226	0.838	0.181	0.572	0.327	0.577	0.222	0.162	0.163	0.083	0.274	0.076
FPGA-11	0.301	0.613	0.111	0.491	0.463	0.644	0.179	0.133	0.102	0.178	0.100	0.088
FPGA-12	0.191	0.999	0.32	0.62	0.300	0.494	0.184	0.17	0.144	0.128	0.189	0.131
Average	0.239	0.778	**0.226**	0.616	0.439	**0.656**	0.134	0.133	**0.099**	0.137	0.233	**0.127**

Table 6: A comparison between different estimation methods is shown for the horizontal congestion prediction under Setup 1.

	NRMS (lower better)			SSIM (higher better)			PIX (lower better)			EMD ($\times 10^{-2}$) (lower better)		
Design	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed	RUDY	*pix2pix*	Proposed
FPGA-1	0.107	0.786	0.067	0.89	0.231	0.955	0.068	0.306	0.026	0.063	0.508	0.033
FPGA-2	0.101	0.906	0.065	0.844	0.599	0.928	0.048	0.134	0.033	0.043	0.187	0.037
FPGA-3	0.199	1.058	0.307	0.477	0.424	0.78	0.118	0.098	0.073	0.134	0.343	0.177
FPGA-4	0.275	0.671	0.152	0.376	0.51	0.795	0.128	0.095	0.072	0.177	0.267	0.092
FPGA-5	0.341	0.171	0.285	0.372	0.705	0.642	0.13	0.077	0.063	0.246	0.118	0.191
FPGA-6	0.218	0.647	0.173	0.354	0.465	0.797	0.167	0.108	0.116	0.149	0.341	0.167
FPGA-7	0.252	0.439	0.159	0.306	0.611	0.778	0.143	0.127	0.105	0.127	0.198	0.151
FPGA-8	0.296	0.878	0.239	0.266	0.462	0.782	0.261	0.199	0.182	0.206	0.348	0.148
FPGA-9	0.297	0.517	0.268	0.228	0.617	0.442	0.24	0.169	0.217	0.165	0.205	0.23
FPGA-10	0.209	0.505	0.13	0.283	0.491	0.705	0.249	0.146	0.154	0.186	0.316	0.095
FPGA-11	0.351	0.332	0.227	0.19	0.651	0.691	0.262	0.19	0.143	0.257	0.058	0.22
FPGA-12	0.243	0.541	0.185	0.295	0.505	0.726	0.224	0.204	0.189	0.18	0.161	0.094
Average	0.241	0.621	**0.189**	0.407	0.523	**0.752**	0.17	0.155	**0.115**	0.162	0.255	**0.137**

Table 7: A comparison between Setup 1 and Setup 2 is shown.

		Setup 1	Setup 2
NRMS	Horizontal	0.189	0.067
	Vertical	0.226	0.057
SSIM	Horizontal	0.752	0.879
	Vertical	0.656	0.815
PIX	Horizontal	0.115	0.058
	Vertical	0.099	0.053
EMD ($\times 10^{-2}$)	Horizontal	0.137	0.074
	Vertical	0.127	0.045

Table 8: The wirelength comparison under different prediction methods is shown.

Design	Full Routing Capacity			Reduced Routing Capacity		
	Rtd WL RUDY	Rtd WL Ours	Imp	Rtd WL RUDY	Rtd WL Ours	Imp
FPGA-1	336117	336117	0.00%	336117	336117	0.00%
FPGA-2	691618	691618	0.00%	691618	691618	0.00%
FPGA-3	3062734	3062734	0.00%	3062734	3062734	0.00%
FPGA-4	5550659	5551473	-0.01%	5557608	5551473	**0.11%**
FPGA-5	10538770	9797007	**7.04%**	N/A	N/A	N/A
FPGA-6	5773333	5773333	0.00%	5777149	5773333	0.07%
FPGA-7	9182199	9163640	**0.20%**	9199730	9163640	**0.39%**
FPGA-8	9053192	9053192	0.00%	9055093	9055093	0.00%
FPGA-9	11641853	11635870	**0.05%**	11652436	11635870	**0.14%**
FPGA-10	5515319	5515319	0.00%	5515319	5515319	0.00%
FPGA-11	11777500	11757650	**0.16%**	11877778	11757650	**1.01%**
FPGA-12	6235694	6235694	0.00%	6224962	6235694	-0.17%

Table 9: The number of matches among the top 10% most congested placements are compared across different methods.

Design	1	2	3	4	5	6	7	8	9	10	11	12	Average
RUDY	4	4	2	2	0	0	4	2	2	5	3	10	3.16
pix2pix	1	3	2	1	0	1	0	1	2	2	0	1	1.17
Proposed	13	12	4	5	6	1	2	2	3	2	3	3	4.67

Table 10: The runtime comparison is presented (sec).

Design	1	2	3	4	5	6	7	8	9	10	11	12
NCTU	3	4	15	19	29	29	34	30	41	44	45	53
pix2pix	GPU: 0.117 CPU: 0.284											
Proposed	GPU: 0.099 CPU:1.050											

[6] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: routability prediction for mixed-size designs using convolutional neural network," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[7] C.-W. Pui, G. Chen, Y. Ma, E. F. Young, and B. Yu, "Clock-aware ultrascale fpga placement with machine learning routability prediction," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE Press, 2017, pp. 929–936.

[8] C. Yu and Z. Zhang, "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[9] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[10] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[11] M. B. Alawieh, Y. Lin, Z. Zhang, M. Li, Q. Huang, and D. Z. Pan, "GAN-SRAF: Sub-resolution assist feature generation using conditional generative adversarial networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[12] "ISPD 2016 routability-driven fpga placement contest," http://www.ispd.cc/contests/16/ispd2016_contest.html, accessed: 2019-05-20.

[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[14] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed *et al.*, "VTR 7.0: Next generation architecture and cad system for fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, p. 6, 2014.

[15] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[17] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[18] E. Levina and P. Bickel, "The earth mover's distance is the mallows distance: Some insights from statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 251–256.

[19] C.-L. E. Cheng, "RISA: Accurate and efficient placement routability modeling," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1994, pp. 690–695.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] 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.

[22] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," vol. 32, no. 5, pp. 709–722, 2013.