CongestNN: An Bi-Directional Congestion Prediction Framework for Large-Scale Heterogeneous FPGAs

Chenyue Ma[1,*], Yifeng Xiao[1], Sifei Wang[2], Jun Yu[1,2] and Jianli Chen[1,*]
[1]State Key Laboratory of ASIC and System, Fudan University, Shanghai 200433, China
[2]Shanghai Fudan Microelectronics Group Co., Ltd, Shanghai 20000, China
[*]Email: cyma20@fudan.edu.cn, chenjianli@fudan.edu.cn

2021 IEEE 14th International Conference on ASIC (ASICON) | 978-1-6654-3867-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASICON52560.2021.9620520

***Abstract*—As FPGA technology continues scaling, design closure process requires more iterations during placement and routing (P&R) for large designs. Accurate advanced routing congestion prediction can effectively relieve this problem on a large scale, which is considered as one of the most significant and toughest tasks in the FPGA design flow. This paper proposes a novel deep-learning framework named CongestNN to predict routing congestion maps of both horizontal and vertical directions in the global placement (GP) stage with a unique combination of GP-based features. A-net, a newly designed type of Fully Convolutional Network (FCN), is deployed to accomplish the prediction part given the input features. The model is trained and tested on GPU using 22,682 cropped feature maps derived from 112 industrial benchmarks placed by the Procise placer and corresponding golden congestion images supplied by Xilinx Vivado. CongestNN is also incorporated into Procise straightforward to provide instant congestion predictions with the average PCC of 95.18%, while the runtime merely takes over one second.**

***Index Terms*—Routability; Congestion map; FPGA; Global placement; FCN**

I. Introduction

Placement and routing (P&R) are two key steps in the physical design flow of Field Programmable Gate Array (FPGA). These two steps map a circuit description into the physical layout and directly impact the ultimate FPGA performance. As the P&R process needs to be conducted for multiple iterations before the convergence to the final layout, this time-consuming process usually takes several hours, and in the worst case, routing may fail after hours of placement for the sake of high congestion.

Generally, routability information is not available until routing design completes, which is too late to modify layout results. Since making use of routability information has been proved effective in reducing congestion and shortening FPGA design closure time, routability needs to be predicted accurately as early as possible in the P&R process. Therefore, early estimation of routability is then considered as a crucial factor in the P&R process, especially for large-scale designs and receives close attention from both academia and industry.

In this paper, the problem of routability estimation is equated to the prediction of routing congestion maps as the distribution of routing congestion is highly related to routability. Originally, placement is conducted to optimize total wirelength and timing without taking routability into account, but placements with low performance may be produced, which takes longer implementation time and turn out to be unroutable. To solve this problem, most state-of-the-art P&R tools leverage the trial global router in the global placement (GP) stage to provide a global routing (GR) result and corresponding congestion distribution based on the current placement, with less time consumption than conventional routers. This method can achieve high accuracy but cost considerable long runtime.

Recently, advances in machine learning are widely used in many fields of electronic design automation (EDA). Various techniques are proposed to address the challenge of fast and accurate routability prediction. In [1], Routenet is built upon CNN to forecast the number of DRV of cell placement solutions before detail placement and the locations of DRC hotspots before detail routing. It is designed for ASIC instead of FPGA to improve the global router's prediction accuracy of DRC hotspots. For FPGA, [2] presents a CNN-based framework to predict routability during placement and integrate it into GPlace3.0 [3] to predict routability. However, the routability is only a binary value in which 1 represents routable and 0 represents unroutable. In the latest work [4], the problem is casted as an image-to-image translation task and a conditional generative adversarial network model is adopted to estimate congestion map for large-scale FPGA designs. The features are extracted after placement, which is not early enough to adjust the layout.

In this work, we present CongestNN, a GP-based routing congestion map predictor, for large-scale heterogeneous FPGA designs. The framework adopts an adapted Fully Convolutional Network (FCN) model called A-net, which makes fast and accurate routing congestion map predictions during GP. It utilizes well-chosen placement features related to routing congestion as model input and ground-truth congestion maps exported from Xilinx Vivado implementation tool after routing as labels to train and test the model.

The main contributions of our work are as follows:

- We propose a high-precision bi-directional routing congestion map prediction framework named CongestNN for heterogeneous FPGA architectures and incorporate it into the placement flow of Procise [5] to predict routability instantly.
- We design CongestNN to extract a brand new combination of features from the placement as prediction input and build a customized FCN architecture named A-net trained on GPU to cast routing congestion prediction.
- The experiment results show that the proposed approach has the merits of high PCC, low MANE and SDNE, and brief prediction time. It achieves the PCC value of 95.18% on average compared to the golden results of Vivado.

The rest of the paper is organized as follows. In Section II, we introduce the related concepts and formulate the problem. In Section III, our algorithm is described in detail. Experimental results are presented in Section IV, and the conclusion are drawn in Section V.

II. Preliminaries

A. Network Introduction CNN/FCN

Convolutional Neural Network (CNN) is a class of machine learning techniques that proved to be effective in prediction tasks. CNN is made up of several different functional layers to extract valuable information from input data and adjust the parameters of the CNN model for prediction. Diverse functional layers mainly include convolutional layer, pooling layer and fully-connected layer. Convolutional layer and pooling layer are responsible for features extraction and data down-sampling of the input data.

Passing through these layers, the input space will be compressed and high-level information is obtained. In CNN, the obtained data (denoted as matrices) will be flattened to 1 dimension and send to fully-connected layer for output.

Fully convolutional network (FCN) is a kind of CNN with only convolutional layers and pooling layers. FCN has an organized architecture called encoder-decoder, where encoder extracts feature representations from the raw data, and decoder processes the representations for required outputs by means like up-sampling. In recent studies, FCN has demonstrated its ability in image segmentation. We try to use FCN, for congestion estimation in FPGA designs.

B. Heterogeneous FPGA Architecture

The target FPGA device of our work is XC7K325T, part of the Xilinx Kintex-7 family [6], which has been widely used in industry due to its low cost and high system performance. Nevertheless, the proposed methods can also be applicable to other heterogeneous FPGA architectures by rectifying parameters concerned. Fig. 1(a) is a typical heterogeneous FPGA architecture, an array composed of heterogeneous programmable blocks including CLB, RAM and DSP, surrounded by programmable input/output (IO) blocks that interface the chip to outside. CLBs are the major resource to implement general-purpose combinatorial and sequential circuits, arranged in columns throughout the device. Switch boxes can provide both intra-slice as well as inter-slice connectivity.

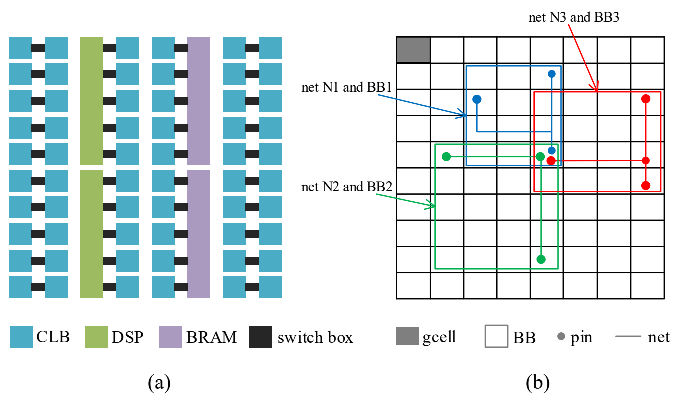

Fig. 1: Example of FPGA Architecture

The routing region is uniformly divided into unit global cells (gcells) encased by horizontal and vertical equidistant grid lines, as shown in Fig. 1(b). Each gcell has a number of horizontal and vertical routing resources for congestion estimation. To give a base for further description, it is necessary to add the definition of bounding box (BB) and half perimeter wirelength (HPWL). The BB of net n stands for the rectangle enclosing all the pins of the net. Fig.1(b) draws several nets and their corresponding BBs in pairing colors. And HPWL is the half perimeter of the BB, often used as the estimation of net wirelength. For net n, its HPWL is:

$$HPWL_n = w_n + h_n, \tag{1}$$

where w_n is the width of BB and h_n is the height of BB.

C. Problem Formulation

In this work, we aim to propose a high-accuracy bi-directional congestion map prediction framework for large-scale heterogeneous FPGA designs to estimate routability ahead of routing, making use of the inter-placement layout information. Besides, the derived framework is embedded into the global placement flow to predict routability instantly. The early estimation of routability refers to predicting routing congestion map images, which is highly related with routability, after layout-driven placement in the global placement stage.

Problem 1. *Given the GP result of a benchmark, the objective is to find a fast and accurate congestion map predictor:*

$$f_{congestion} : X_i \in \mathbb{R}^{w\times h\times F} \to Y_i \in \mathbb{R}^{w\times h\times 2}$$

For the i_{th} benchmark, the model input is F features highly correlated with routing congestion extracted from GP files, represented as $X_i \in \mathbb{R}^{w\times h\times F}$. The output label is ground-truth congestion maps from Vivado, and the target output is a dual-channel matrix $Y_i \in \mathbb{R}^{w\times h\times 2}$, corresponding to $w\times h$ gcells horizontally and vertically.

III. Algorithm

A. Feature Extraction

To get the useful information correlated with routing congestion from the layout after global placement, five features are extracted as follows. Fig. 2 shows the sample feature maps generated from a benchmark in RGB mode.

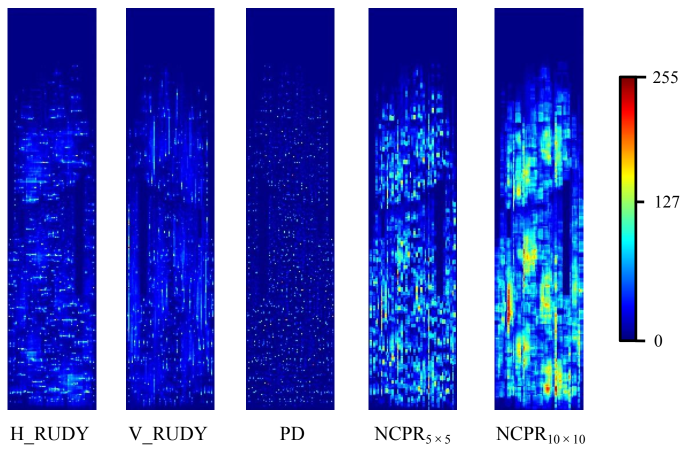

Fig. 2: Sample feature maps of a benchmark

1) Horizontal and Vertical RUDY: Rectangular Uniform wire Density (RUDY) [7] is a widely used pre-routing congestion estimator which can be quickly calculated. Since all routers will manage to route nets within the bounding boxes and the routing demand of one net is inconsequential in comparison to that of thousands and even millions of nets in total, RUDY can precisely model the net distribution over the chip area and exactly estimate the routing demand of each net. Here we divide RUDY into horizontal and vertical RUDY to better estimate routing demand in these two directions.

$$RUDY_n = \frac{HPWL_n}{S_n}, \tag{2}$$

$$H\_RUDY(x,y) = \sum_{n=1}^{N} c_n \omega_n RUDY_n \frac{w_n}{w_n + h_n}, \tag{3}$$

$$V\_RUDY(x,y) = \sum_{n=1}^{N} c_n \omega_n RUDY_n \frac{h_n}{w_n + h_n}, \tag{4}$$

where (x, y) is the coordinate of the gcell to be computed, N is the set of nets, $HPWL_n$ is the half perimeter wirelength of net n, S_n is the area occupied by the bounding box of net n (i.e. BB_n), c_n is a Boolean variable that denotes if (x, y) locates in BB_n, w_n and h_n is the width and height of BB_n, and ω_n is a correction factor positively related with the pin count of net n to increase the accuracy of $HPWL$ on high-fanout nets.

2) Pin Density (PD): Considering that a net may not cross the border of a gcell, pin density, i.e. the number of pins inside a gcell, is utilized to improve the estimation of routing

congestion.

$$PD(x,y) = \sum_{n=1}^{N} pins_n(x,y), \tag{5}$$

where $pins_n(x,y)$ is the number of pins located in (x,y) for net n.

3) Nets Cut Per Region (NCPR) [8]: NCPR is the total number of nets cut in a specified region, and a net cut means there is a net that has at least one pin both inside and outside the region. Two window sizes are taken into consideration: 5×5 and 10×10.

$$NC_n(x,y) = \begin{cases} 1 & \text{if } pins\_w_n(x,y) < pins_n(x,y), \\ 0 & \text{else}, \end{cases} \tag{6}$$

$$NCPR_{w\times w}(x,y) = \sum_{n=1}^{N} NC_n(x,y), \tag{7}$$

where $pins\_w_n(x,y)$ is the number of pins for net n inside the window of (x,y), NC_n(x,y) is a Boolean variable depending on if net n cuts the window, that is to say, if net n has at least one pin both inside and outside the window region.

All of these features can be easily and efficiently computed in several milliseconds. After extracting the data of these five features respectively from each benchmark, the data is converted into grayscale feature maps, which looks more intuitive.

Algorithm 1: Proposed Feature Extraction Algorithm

```
Input: Global Placed Netlist of the $i_{th}$ benchmark
Output: Stack of feature maps $X_i \in \mathbb{R}^{w\times h\times F}$
 1: for $n \in N$ do
 2:     for $p \in Pins(n)$ do
 3:         $(x,y) \leftarrow$ the x and y coordinate of p
 4:         $pins_n(x,y)$++
 5:         $pins\_w_n(x,y)$++ if p in Window
 6:         update $BB_n$: (xmin, xmax, ymin, ymax)
 7:     end for
 8:     add $pins_n(x,y)$ to $PD(x,y)$
 9:     for $(x,y) \in BB_n$ do
10:         compute and add to H_RUDY and V_RUDY(x,y)
11:         if 0 <$pins\_w_n(x,y)$ <$pins_n(x,y)$ then
12:             $NCPR(x,y)$++
13:         end if
14:     end for
15: end for
16: remove the columns of DSPs and RAMs and normalize
17: visualize and stack the features sequentially
18: return $X_i \in \mathbb{R}^{w\times h\times F}$
```

For a single benchmark, the complete flow of feature extraction is described in Algorithm 1. Besides calculating five features (Lines 1-17), Line 18 removes the columns of DSPs and RAMs, because Vivado only gives the vertical and horizontal routing congestion of CLBs. Then in line 19, the features $X_{ij} \in \mathbb{R}^{w\times h}$ are visualized and stacked in sequence into a three-dimensional input tensor $X_i \in \mathbb{R}^{w\times h\times F}$, where the width of feature map w = 77, the height h = 350 and the number of features F = 5.

B. A-Net Model

With the extracted features from different FPGA designs, the A-Net model is designed for predicting congestion maps, as shown in Fig. 3. The left part of the A-Net can be regarded as a generator to extract high-level features of the raw input, while the right part is a discriminator for image expansion. The number shown below each block represents the channel number. Given that each cell corresponds to a congestion value, the model is responsible for contraction and recovery. The yellow arrow denotes a down-sampling process with 2×2 max-pooling operation, followed by a convolutional layer to double the channel number from the blue block to the white block. On the other hand, the red arrow represents a general up-sampling method called bilinear interpolation, followed by a convolutional layer to halve the channel number. Also, a concatenated skip connection [9] is used to combine more low-level features to the generated maps. An FPGA design has fixed architecture, which demonstrates the importance of the low-level features.

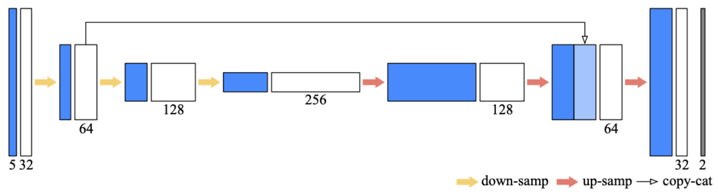

Fig. 3: Structure of A-Net Model.

The size of the raw input is 64×64 with 5 channels, obtaining from extracted features, and the output congestion maps are 64×64 with 2 channels, representing horizontal congestion and vertical congestion respectively.

C. CongestNN Prediction Framework

The prediction framework of the proposed CongestNN is illustrated in Fig. 4, composed of feature extraction and A-net model explained above.

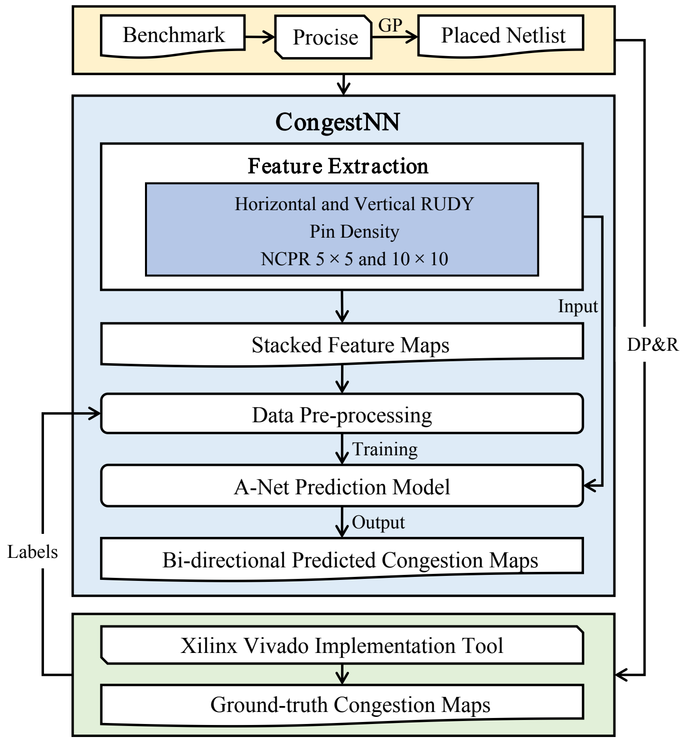

Fig. 4: Our Algorithm Flow

At first, benchmarks are global placed (GP) by Procise (a commercial placer) and the placed netlist of each benchmark is generated. Then features are extracted from the netlists and converted to stacked maps.

With regard to labels, as shown in Fig. 4, Xilinx Vivado implementation tool is used to generate output labels for datasets. This is done by pipelining Procise to Vivado. The placement results are added in Vivado to create a new project and conduct the following implementation. After detail placement and routing (DP&R) complete, Vivado reports the data of horizontal and vertical routing congestion per CLB. The data is then exported and visualized as pictures. To better distinguish the congestion over 100%, the congestion percentage ranging from 0 to 200 is mapped to image grayscale value ranging from 0 to 255. These ground-truth maps from Vivado are regarded as labels.

Afterwards, data pre-processing is conducted to augment the dataset for the training and testing process. The feature maps and ground-truth congestion maps of all the testcases are further cropped into an array of smaller square images which is set to be 64×64 and the crop stride is 3. Besides, there may be completely black images which means no congestion and these repetitive pictures are deleted as they have no use for training.

CongestNN is further deployed into the placement flow of Procise. Given a new benchmark, first global place and get the placed netlist, and in the CongestNN module, five features presented in Sec. III-A are extracted. Finally, these features are inputted to the trained A-Net without cropping, and bi-directional predicted congestion maps are outputted.

IV. Experimental Results

Our proposed algorithm is embedded into the Procise placer, implemented using the C++ programming language, and compiled using gcc 5.4.0. The A-Net model is implemented using PyTorch 1.6.0 and trained on a single Nvidia Tesla v100 GPU. Then the obtained model is converted to a TorchScript model as the C++ interface.

Using 112 industrial benchmarks, we obtained 22682 five-dimensional cropped 64×64 feature images, labelled by ground truth from Vivado. 70% of them are randomly selected as the training set and the remaining 30% as the testing set in model generation.

TABLE I: Performance on each benchmark

Benchmark	PCC	MANE	SDNE	Prediction Time(s)
case1	0.9420	0.0015	0.0059	1.6235
case2	0.9443	0.0028	0.0103	1.6330
case3	0.9704	0.0150	0.0310	1.6556
case4	0.9452	0.0006	0.0034	1.6086
case5	0.9594	0.0035	0.0101	1.6271
case6	0.9576	0.0017	0.0082	1.6438
case7	0.9759	0.0021	0.0082	1.6261
case8	0.9168	0.0010	0.0061	1.6243
case9	0.9608	0.0018	0.0080	1.6675
case10	0.9321	0.0181	0.0401	1.6187
case11	0.9426	0.0175	0.0345	1.6311
case12	0.9745	0.0052	0.0117	1.6251

TABLE II: Overall performance of CongestNN

PCC	MANE	SDNE	Prediction Time(s)	Epochs	Train Time(s)
0.9518	0.0059	0.0148	1.6320	300	3266

The metrics used for evaluation are given below:

1) Pearson Correlation Coefficient (PCC): measures the correlation between two variables, ranging from - 1 to 1, where 1 represents perfect positive correlation. (Higher better)
2) Mean Absolute Normalized Error (MANE): measures the normalized error between the ground-truth and predicted congestion maps. (Lower better)
3) Standard Deviation of the Normalized Error (SDNE): measures the dispersion level of MANE. Lower SDNE means the values tend to be close to the mean value. (Lower better)
4) Prediction Time: measures the time used for prediction.

We implemented CongestNN on 12 relatively complex industrial benchmarks to evaluate the model performance. The experimental results on each benchmark are listed in Table I. It can be seen that our proposed algorithm achieves high scores across all the benchmarks and the metrics of each benchmark vary in a small range. The overall performance is shown in Table II. The average PCC achieves 95.18%, close to perfect prediction, while the average MANE and SDNE are both low and nearly zero. Besides, the prediction takes just over one second. Fig. 5 shows the comparison of golden and predicted congestion maps for case10. To make the results easier to observe, the original grayscale images are transformed into RGB images. For both horizontal and vertical directions, the results of CongestNN and Vivado are evidently similar to each other.

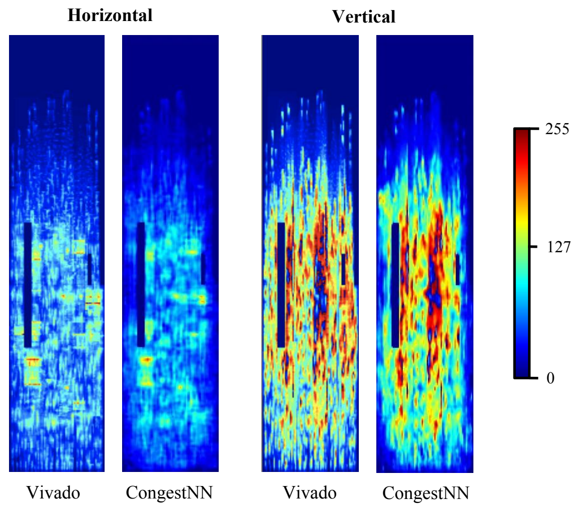

Fig. 5: A sample congestion prediction result of case10

V. Conclusion

In this paper, an effective bi-directional congestion prediction framework named CongestNN has been presented to solve the problem of early routability estimation at the global placement stage. It starts with the extraction of features to estimate the congested areas and congestion extent from the placed layout. And then the A-net prediction model is trained and tested using the pre-processed set of features together with the labels generated by Vivado. Experimental results on 12 benchmarks demonstrate that CongestNN can effectively predict congestion given a global placed netlist with a fairly short run time.

Acknowledgement

This work was supported by the State Key Laboratory of ASIC & System Grant 2021MS008, and by the National Natural Science Foundation of China under Grants 61977017.

References

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

[2] A. Alhyari, A. Shamli, Z. Abuwaimer, S. Areibi, and G. Grewal, "A deep learning framework to predict routability for fpga circuit placement," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 334–341.

[3] Z. Abuowaimer, D. Maarouf, T. Martin, J. Foxcroft, and A. Vannelli, "An efficient routability-driven analytic placer for ultrascale fpga architectures," *ACM Transactions on Design Automation of Electronic Systems*, vol. 23, no. 5, pp. 1–33, 2018.

[4] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, and D. Z. Pan, "High-definition routing congestion prediction for large-scale fpgas," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 26–31.

[5] Shanghai Fudan Microelectronics Group, "Procise," https://fmsh.com.

[6] Xilinx, "Kintex-7 FPGAs," https://www.xilinx.com/products/silicon-devices/fpga/kintex-7.html.

[7] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Design, Automation Test in Europe Conference Exhibition*, 2007, pp. 1–6.

[8] D. Maarouf, A. Alhyari, Z. Abuowaimer, T. Martin, and A. Vannelli, "Machine-learning based congestion estimation for modern fpgas," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 427–4277.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.