Multiscale Feature Attention and Transformer Based Congestion Prediction for Routability-Driven FPGA Macro Placement

1st Hao Gu
Dept. of National ASIC System Center
Southeast University
Nanjing, China
230218727@seu.edu.cn

2nd Xinglin Zheng
Dept. of National ASIC System Center
Southeast University
Nanjing, China
220236395@seu.edu.cn

3rd Youwen Wang
Dept. of National ASIC System Center
Southeast University
Nanjing, China
220236428@seu.edu.cn

4th Keyu Peng
Dept. of National ASIC System Center
Southeast University
Nanjing, China
kypeng@seu.edu.cn

5th Ziran Zhu
Dept. of National ASIC System Center
Southeast University
Nanjing, China
zrzhu@seu.edu.cn

6th Jun Yang
Dept. of National ASIC System Center
Southeast University
Nanjing, China
dragon@seu.edu.cn

***Abstract*—As routability has emerged as a critical task in modern field-programmable gate array (FPGA) physical design, it is desirable to develop an effective congestion prediction model during the placement stage. Given that the interconnection congestion level is a critical metric for measuring the routability of FPGA placement, we utilize that level as the model training label. In this paper, we propose a multiscale feature attention (MFA) and transformer based congestion prediction model to extract placement features and strengthen their association with congested areas for effective FPGA macro placement. A convolutional neural network (CNN) component is first designed to extract multiscale features from grid-based placement. Then, a well-designed MFA block is proposed that utilizes the dual attention mechanism on both spatial and channel dimensions to enhance the representation of each multiscale feature. By incorporating MFA blocks and CNN's output at each skip connection layer, our model substantially enhances its capability to learn features and recover more precise congestion level maps. Furthermore, multiple transformer layers that employ dynamic attention mechanisms are utilized to extract global information, which can significantly improve the difference between various congestion levels and enhance the ability to identify these levels. Based on the ten most congested and challenging benchmarks from the MLCAD 2023 FPGA macro placement contest, experimental results show that our model outperforms existing congestion prediction models. Furthermore, our model can achieve the best routability and score among the contest winners when integrated into the macro placer based on DREAMPlaceFPGA.**

***Index Terms*—machine learning, Xilinx UltraScale+ FPGA, congestion prediction, routability, macro placement**

I. Introduction

Field-programmable gate arrays (FPGAs) are celebrated for their high flexibility, rapid prototyping, and reconfigurability, which makes them suitable for a wide range of applications. A modern FPGA typically integrates millions of cells and nets, and thousands of macros, posing great challenges to placement and routing. Neglecting routability in the FPGA placement may lead to severe routing congestion, prolonging design cycles [1], [2]. In summary, a congestion prediction method is essential for optimizing routability. However, with the complexity of interconnections in advanced design architectures greatly, traditional congestion prediction methods [3], [4] may not obtain accurate enough results.

Due to the capability for accurate prediction and rapid decision-making, machine learning (ML)-based methods have been adopted for congestion prediction [5], [6], [7], [8]. Notably, [6] was the first to apply models based on convolutional neural network (CNN) and U-net architecture to predict congestion maps. Specifically, PGNN [7] combines graph neural network (GNN) with U-net to improve routing congestion. GNN is used to derive pin accessibility information from the pin proximity graph. Concurrently, U-net extracts routing congestion data from grid-based features, resulting in faster and more precise predictions of design rule check (DRC) hotspots. In addition, PROS 2.0 [8] integrates ResNet [9] with U-net for congestion prediction. This model uses ResNet to extract grid-based placement features and is trained with real global routing results. However, these methods are weak in global feature extraction and the ability to accurately recover congestion maps from hidden high-dimensional features.

Recently, to further motivate researchers to consider routability during FPGA macro placement, the MLCAD 2023 contest [10] provided novel benchmarks specifically designed for the 16nm Xilinx UltraScale+ architecture. This contest is designed to encourage the integration of ML methods with FPGA macro placement. However, the top three contestants primarily

This work was partially supported by the National Key Research and Development Program of China under Grant 2022YFB4400500, the National Natural Science Foundation of China under Grants 92373207 and 62104037, and the Zhishan Young Scholar Support Program of Southeast University under Grant 2242024RCB0052. (*Corresponding author:Ziran Zhu.*)

978-3-9826741-0-0/DATE25/© 2025 EDAA

achieved their advancements by refining RUDY-based analytical methods [11] rather than incorporating ML-based model. To improve the accuracy of congestion prediction and guide the FPGA placer to obtain a routing-friendly placement, this paper proposes a novel multiscale feature attention and transformer based congestion prediction model for FPGA macro placement. Our main contributions can be summarized as follows:

- A multiscale feature attention (MFA) and transformer based congestion prediction model is proposed to integrate CNN and transformer layers within U-net based framework. This model is designed to extract multiscale features from grid-based placement critical to routing and strengthen their association with congested areas.
- A well-designed MFA block is proposed that utilizes the dual attention mechanism on both spatial and channel dimensions to effectively extract the detailed multiscale features. Positioned on each skip connection layer, the MFA block preserves and enhances multiscale hidden features. This augmentation boosts the decoder's capability to recover more precise congestion level maps.
- Experimental results demonstrate that the proposed congestion prediction model outperforms previous models on the most challenging ten designs in the MLCAD 2023 public benchmarks. Furthermore, the FPGA macro placement method guided by the proposed model has improved the best routability score in the contest by 8%.

II. Preliminaries

A. MLCAD 2023 UltraScale FPGA Architecture

This subsection describes the FPGA architecture, 16nm Xilinx Ultrascale+ XCVU3P [10], employed in the 2023 MLCAD contest. The architecture features four heterogeneous site types: configurable logic blocks (CLBs), digital signal processors (DSPs), block RAMs (BRAMs) and ultra RAMs (URAMs). We regard DSP, BRAM and URAM as macros, while other instances are considered as cells. Besides, the cascade shape and region constraints imposed by real-world applications must be satisfied. On the one hand, each cascade shape constraint requires that specific macros must be placed on consecutive sites in a predetermined order. On the other hand, each region constraint requires that instances assigned to the constraint must be placed on sites within that region. Notably, instances not assigned to any region constraint can be placed on any sites.

B. Routability Metric

According to the MLCAD2023 contest [10], the routability metric comprises two primary components: S_{IR} and S_{DR}, which respectively represent initial and detailed routing scores. S_{IR} is calculated based on the routing congestion level identified by the Vivado initial router. Specifically, the congestion level is assessed across the interconnect tile grid, depicted in Fig. 1, where each tile grid corresponds to a distinct congestion level. As depicted on the left side of Fig.1, tiles are color-coded to indicate congestion intensity, with darker shades of yellow representing higher congestion level. Routing congestion occurs when routing resources in some tile grids are overused. According to [10], S_{IR} is determined by the interconnect congestion level in four directions (east, south, west, and north):

$$S_{IR} = 1 + \sum_{d=1}^{4} \left[\max\left(0, L_{\text{short},d} - 3\right)^2 + \max\left(0, L_{\text{global},d} - 3\right)^2 \right], \tag{1}$$

where $L_{\text{short},d}$ and $L_{\text{global},d}$ denote short and global congestion in directions d. Notably, penalties are imposed only for congestion levels of 4 and above.

Additionally, S_{DR} is derived from the number of iterations by the Vivado detailed router. An increased number of iterations indicates that congestion adversely affects the routability of the FPGA placement. The overall routability score S_R defined in the contest is :

$$S_R = S_{IR} \times S_{DR}. \tag{2}$$

The final evaluation score S_{score} defined in the contest is:

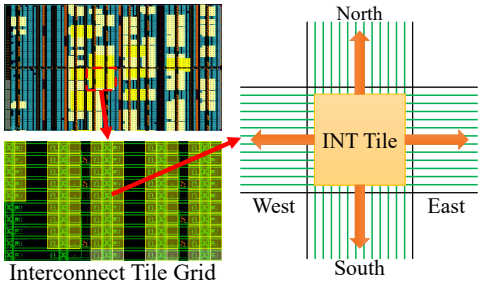

Fig. 1. An example of a target FPGA interconnect tile grid.

$$S_{score} = [1 + \max(0, T_{macro} - 10)] \times S_R \times T_{P\&R}, \tag{3}$$

where T_{macro} denotes the runtime of macro placement in minutes, S_R is the routability score defined in Eq. (2), and $T_{P\&R}$ is the runtime of the cell placement and routing performed by Vivado in hours.

III. Proposed Model Framework

A. Overview of Congestion Prediction Model

Fig. 2 shows the framework of our congestion prediction model, consisting of three components: grid-based input features, encoder with MFA blocks and transformer layers, and decoder. Specifically, the model inputs $X_{input} \in \mathbb{R}^{6\times H\times W}$ consist of grid-based placement features that capture various aspects of circuit placement, with W and H representing width and height of girds. The encoder first extracts multiscale features with CNN layers. After each CNN layer, an MFA block is incorporated that utilizes the dual attention mechanism on both spatial and channel dimensions to enhance multiscale feature representation. Furthermore, multiple vision transformer [12] layers that employ dynamic attention mechanisms are utilized to extract global information, which can significantly improve the difference between various congestion levels. In the decoder, the integrating of upsampling layer outputs with corresponding MFA blocks via skip connection layers enhances the capability of model to precisely recover congestion level maps. Finally, our model accurately predicts the congestion level map $Y_{out} \in \mathbb{R}_{+}^{1\times H\times W}$.

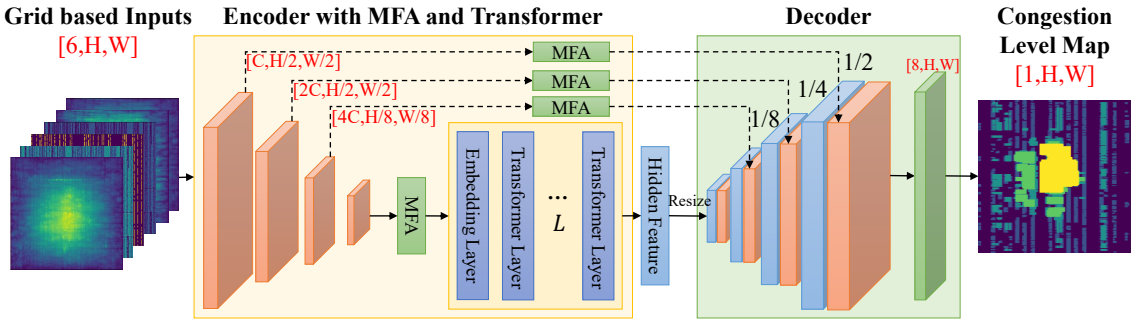

Fig. 2. The overall flow of our proposed congestion prediction model.

B. Grid-based Input Features

The input features extracted from the grid-based placement significantly impact the accuracy of congestion prediction. Based on the placement grids, we identify six key grid-based features that are strongly correlated with congestion. These features are derived from the distribution of routing resources, cells, and nets. Detailed descriptions of each feature are provided below. **Macro Map** indicates the occupancy status of each grid by a macro, which can help identify congestion areas with macro locations. **Horizontal/Vertical Net Density Map** estimates the density of nets that are expected to route each grid horizontally or vertically. **RUDY Map** is a popular routing demand estimation technique, which is the superposition of horizontal net density and vertical net density. **Pin RUDY Map** refers to a pin density map inspired by RUDY, representing the horizontal and vertical pin density of all nets within a grid. **Cell Density Map** is defined as the number of cells in each grid.

C. Encoder with MFA block and Transformer Layer

1) Multiscale Feature Extraction: As depicted in Fig. 2, our congestion prediction model employs four CNN layers to extract multiscale features from the grid-based placement input features. Specifically, the grid-based input features $X_{input} \in \mathbb{R}^{6\times H\times W}$ are processed through four CNN layers within ResNet architecture [9] to perform the downsampling operation. Each CNN layer effectively reduces the dimensions of input features (height and width) by half and doubles the number of channels. Fig. 5 shows the detailed architecture of our model. The number in each block indicates the output size of each layer. Consequently, the output size of four CNN layers (Down) are $[H/2, W/2]$, $[H/4, W/4]$, $[H/8, W/8]$ and $[H/16, W/16]$ with channel counts of C, $2C$, $4C$, and $8C$, respectively. With the help of extracting multiscale features, our model can effectively capture information across various scales.

To address the gap inherent in traditional U-Net [13] architecture, which lacks the ability to capture global information, we incorporate MFA blocks within the skip connection layers. Each CNN layer is followed by an MFA block that utilizes the dual attention mechanism on both spatial and channel dimensions to enhance the representation of each feature. The input to each MFA block corresponds to the scales of $[C, H/2, W/2]$, $[2C, H/4, W/4]$, $[4C, H/8, W/8]$ and $[8C, H/16, W/16]$, respectively. Additionally, after all CNN layers extract features, an additional MFA block is adopted before the vision transformer layer. This MFA block performs specialized feature processing on the input of the transformer, which can enhance the transformer layer's capability to extract hidden features. The detailed description of the MFA block will be given in Section III-C2.

2) Multiscale Feature Attention (MFA) block: The MFA block is primarily composed of two key sub-modules: the position attention module (PAM) and the channel attention module (CAM) [14]. PAM emphasizes the representation of specific features by capturing spatial dependencies across any two positions in features. Given an input feature $M \in \mathbb{R}^{N\times H\times W}$, we feed it into a convolution layer to produce three features B, C, and D. B and C are reshaped to $\mathbb{R}^{N\times L}$, where $L = H \times W$ is the number of pixels. The spatial attention map $P \in \mathbb{R}^{L\times L}$ is then computed as follows:

$$P_{ji} = \frac{\exp(B_i^T \cdot C_j)}{\sum_{i=1}^{L} \exp(B_i^T \cdot C_j)}, j \in \{1, 2, ..., L\}, \tag{4}$$

where P_{ji} represents the influence of position i on position j, and $(\cdot)^T$ means the transpose operation. Subsequently, we reshape D into $\mathbb{R}^{N\times L}$ and matrix multiply it with the transpose of P_j:

$$M_j^p = \alpha \sum_{i=1}^{L} (P_{ji} \cdot D_i) + M_j = \alpha(D \cdot P_j^T) + M_j, \tag{5}$$

where α is a learnable parameter. The final output $M^p = [M_1^p, M_2^p, ..., M_j^p] \in R^{N\times L}$ is reshaped back into $\mathbb{R}^{N\times H\times W}$.

CAM reshapes the input feature $M \in \mathbb{R}^{N\times H\times W}$ to $\mathbb{R}^{N\times L}$. To capture the dependencies between channels, channel attention map $C \in \mathbb{R}^{L\times L}$ is computed as follows:

$$C_{ji} = \frac{\exp(M_i^T \cdot M_j)}{\sum_{i=1}^{L} \exp(M_i^T \cdot M_j)}, j \in \{1, 2, ..., L\}, \tag{6}$$

where C_{ji} represents the influence of channel i on channel j. Subsequently, a matrix multiplication is conducted between M and the transpose of C_j:

$$M_j^c = \beta \sum_{i=1}^{N} (C_{ji} \cdot M_i) + M_j = \beta(M \cdot C_j^T) + M_j, \tag{7}$$

β is also a learnable parameter. Similar to PAM, final output $M^c = [M_1^c, M_2^c, ..., M_j^c]$ is also reshaped into $\mathbb{R}^{N\times H\times W}$.

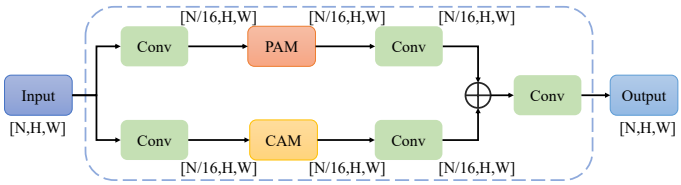

Fig. 3. Detailed structure of the proposed MFA Block.

As shown in Fig. 3, the MFA architecture integrates PAM and CAM to obtain outstanding multiscale feature extraction capabilities. For the PAM/CAM branch, the input features undergo a convolution layer to diminish the channel by a factor of $\frac{1}{16}$, which facilitates the subsequent extraction of position features. Then, the outputs from both branches are summed and passed through a convolution layer to restore the original feature dimensions. According to the input size of MFA obtained in Section III-C1, the output of each MFA block corresponds to the scales of $[C, H/2, W/2]$, $[2C, H/4, W/4]$, $[4C, H/8, W/8]$ and $[8C, H/16, W/16]$, respectively. The MFA block utilizes the complementary strengths of both PAM and CAM, effectively enhancing the multiscale features and improving the model's performance in congestion level prediction.

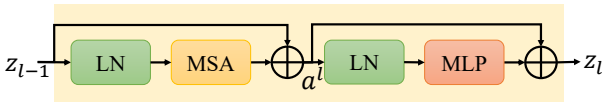

Fig. 4. Detailed architecture of a vision transformer layer.

3) Vision Transformer Layer: After multiscale feature extraction by CNN layers, an additional MFA block before the vision transformer layer to enhance its hidden feature extraction capabilities. The output scale of this MFA block is $[8C, H/16, W/16]$. The vision transformation layer is used to augment global feature extraction capabilities. It begins with an embedding layer that processes the input feature and reshapes it into $[C_t, L]$ ($L = W/16 \times H/16$), where C_t is the number of channels designed for better adaptation to transformer layers. After processing, the features, defined as sequences z_0, are fed into the L multimodal vision transformer layers arranged in series. For each layer l, where $l \in \{1, 2, ..., L\}$, the input z_{l-1} undergoes transformation to produce the output z_l.

Transformer's ability to model long-range dependencies based on the self-attention mechanism allows it to efficiently capture global features. As illustrated in Fig. 4, each vision transformer layer mainly consists of three primary components: multihead self-attention (MSA), multi-layer perceptron (MLP), and layer normalization (LN), with residual connections enhancing the flow within the block. The output after the MSA block a_l is computed as follows:

$$a_l = MSA(LN(z_{l-1})) + z_{l-1}, \tag{8}$$

where $LN(\cdot)$ denotes the layer normalization operation. MSA block uses Q_{l-1} (query), K_{l-1} (key), and V_{l-1} (value) derived from z_{l-1} through linear transformations. The scaled dot-product attention is computed as:

$$MSA(Q_{l-1}, K_{l-1}, V_{l-1}) = Softmax(\frac{Q_{l-1}K_{l-1}^T}{\sqrt{d_k}}V_{l-1}), \tag{9}$$

where $\sqrt{d_k}$ is a normalization factor to avoid abnormal gradients during training. Therefore, the output z^l can be expressed as follows:

$$z_l = MSA(LN(a_l)) + a_l. \tag{10}$$

After passing through all the vision transformer layers, z_L in the last transformer layer serves as the output. In summary, the vision transformer layers can significantly refine the difference between various congestion levels, thereby enhancing the model's capacity to identify them.

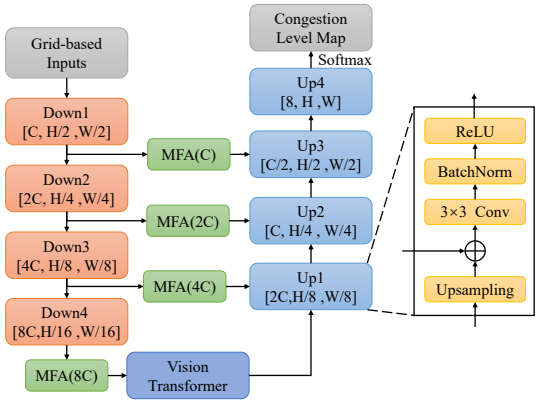

Fig. 5. Detailed architecture of our proposed model.

D. Decoder

As depicted in Fig. 5, the architecture of the decoder comprises four upsampling blocks. Each upsampling block consists of an upsampling layer, a skip connection layer and a CNN layer. By merging the upsampling layer with MFA blocks at each skip connection layer, our model substantially enhances its capability to learn features and recover more precise congestion level maps. Specifically, this process begins with the final output from the encoder, which serves as the initial input to the decoder. At each upsampling block, the input features are upsampled by $2\times$ and concatenated with the corresponding skip connection feature maps from the encoder. The output of each block is passed through a 3×3 convolutional layer, a batch normalization layer and a $ReLU$ layer before being used as the input for the subsequent block. And the output dimension of first three blocks are $[2C, H/8, W/8]$, $[C, H/4, W/4]$ and $[C/2, H/2, W/2]$. The last block needs to change the size of the feature into $8 \times H \times W$, then be subjected to a softmax layer to normalize the predictions. This sequence ensures that the decoded features are accurately translated into classification labels, culminating in the production of the congestion level map with size $1 \times H \times W$.

IV. Proposed FPGA Macro Placement Method

In this section, we summarize our FPGA macro placement method and introduce how the proposed congestion prediction

TABLE I
PREDICTION COMPARISON OF DIFFERENT ML-BASED METHODS ON THE MLCAD 2023 BENCHMARKS.

Design name	#LUT	# FF	#DSP	#BRAM	U-net [6]			PGNN [7]			PROS2.0 [8]			Ours		
					ACC↑	R^2↑	NRMS↓	ACC↑	R^2↑	NRMS↓	ACC↑	R^2↑	NRMS↓	ACC↑	R^2↑	NRMS↓
Design_116	370K	315K	2052	648	0.804	0.827	0.160	0.847	0.857	0.167	0.849	0.856	0.167	0.885	0.890	0.144
Design_120	383K	315K	2052	648	0.742	0.763	0.241	0.777	0.790	0.224	0.803	0.815	0.208	0.855	0.852	0.183
Design_136	315K	268K	1870	590	0.784	0.777	0.221	0.826	0.812	0.200	0.844	0.826	0.189	0.882	0.864	0.164
Design_156	338K	291K	1961	619	0.791	0.804	0.208	0.819	0.829	0.199	0.846	0.835	0.189	0.886	0.860	0.173
Design_176	370K	315K	2052	648	0.811	0.863	0.105	0.838	0.845	0.128	0.879	0.859	0.110	0.892	0.893	0.104
Design_180	383K	315K	2052	648	0.867	0.915	0.132	0.878	0.916	0.131	0.904	0.934	0.116	0.923	0.946	0.104
Design_190	312K	256K	1824	576	0.813	0.821	0.157	0.827	0.832	0.152	0.883	0.882	0.124	0.903	0.901	0.112
Design_197	323K	268K	1870	590	0.764	0.749	0.175	0.799	0.782	0.162	0.793	0.771	0.166	0.858	0.832	0.137
Design_227	363K	303K	2006	634	0.752	0.754	0.215	0.828	0.820	0.178	0.863	0.851	0.160	0.893	0.881	0.140
Design_237	379K	315K	2052	648	0.789	0.802	0.166	0.841	0.845	0.143	0.859	0.861	0.135	0.875	0.867	0.126
Average					0.792	0.808	0.178	0.828	0.833	0.168	0.852	0.849	0.156	**0.885**	**0.878**	**0.139**
Ratio					0.894	0.919	1.282	0.935	0.948	1.214	0.963	0.966	1.128	**1.000**	**1.000**	**1.000**

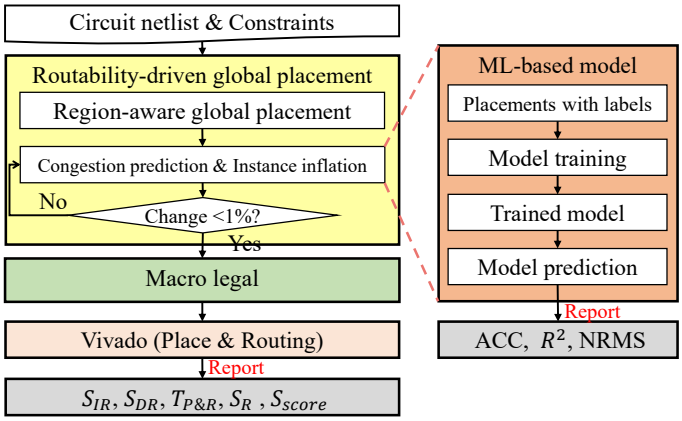

Fig. 6. The overall flow of our proposed method.

model is incorporated into it. As shown in Fig. 6, our macro placer is based on DREAMPlaceFPGA [2] with modifications to satisfy the cascade shape constraints and region constraints outlined in Section II-A. The target FPGA macro placement method is divided into two parts: routability-driven global placement and macro legalization. Before global placement, to address cascade shape constraint, we employ the cascade handling technique introduced in [11], merging macros subject to the same cascade shape constraint into a single large cluster before global placement. The routability-driven global placement consists of two stages: (1) region-aware global placement; and (2) congestion prediction and instance inflation. To effectively handle region constraints, a region tension function is added to the global placement model in the first stage. When the instance distribution meets the conditions $Overflow_t < 0.25$ for $t \in \{DSP, BRAM, URAM\}$ and $Overflow_t < 0.15$ for $t \in \{LUT, FF\}$, congestion prediction and instance inflation will be performed.

Most importantly, to enhance routability, we utilize our trained congestion prediction model, detailed in Section III, to predict congestion map $Y_{out} \in \mathbb{R}_+^{1\times H\times W}$ instead of the original RUDY method. Given that penalties to S_{IR}, defined in Eq. (1), are only incurred when the congestion level is greater than 3, we only inflate the instances in the grid where Y_{out}^i exceeds 3. Hence, the routability-optimized area for each instance/cluster b_i is calculated as follows:

$$A_i^{est} = A_i \times \min\{[\max(1, Y_{out}^i - 2)]^{2.5}, \epsilon\}, \tag{11}$$

where A_i is the current area of b_i, A_i^{est} is the estimated area of b_i, and ϵ is an empirical constant to avoid instances overinflating. Then, the target area increase $\triangle A_i$ of b_i is defined as $\triangle A_i = A_i^{est} - A_i$. To prevent the total inflation area from exceeding the total area of each resource type $t \in T$, the inflation area needs to be scaled by the following factor according to its resource type t:

$$\tau_t = \min\left(\frac{A_t^p - \sum_{i\in t} A_i}{\sum_{i\in t} \triangle A_i}, 1\right), \tag{12}$$

where A_t^p is the total available placement area of t. Then, the final area A_i^{update} of b_i after inflation is calculated as follows:

$$A_i^{update} = A_i + \tau_t \times \triangle A_i. \tag{13}$$

V. EXPERIMENTAL RESULTS

A. Experimental Setup and Datasets

We implemented our congestion prediction model in $Python$ with $Pytorch$, and the proposed FPGA macro placement method in a combination of $C++$ and $Python$. All experiments are conducted on a $Ubuntu$ 20.04 machine with the hardware platform equipped with $AMD\ EPYC$ 7763 $3.5GHz$ CPUs and $NVIDIA\ Geforce$ 4090 GPUs. Besides, we used Xilinx Vivado 2021.1 [15] to perform placement and routing, generate ground truth congestion labels and report critical routability metrics. We used the Adam optimizer with a learning rate of 0.001 when training the model. In Section III, the number of transformer layers L was set to 12. All grid-based features and ground truth labels are resized to the same size 256×256 (i.e., $W, H = 256$).

To better evaluate the performance of our model, we select the ten most congested and challenging benchmarks from the MLCAD 2023 FPGA macro placement contest [10]. Table I lists the benchmark statistics. Each benchmark contains thousands of BRAMs, hundreds of DSPs, and hundreds of thousands of LUTs and FFs. For each benchmark, our proposed FPGA macro placement method presented in Section IV is performed with varying parameters to generate 30 different placement results. To enhance the robustness of our proposed

TABLE II
ROUTABILITY-DRIVEN PLACEMENT COMPARISON ON THE MLCAD 2023 BENCHMARKS.

Design name	UTDA [11]					SEU					MPKU-Improve [16]					Ours				
	S_{score}	S_R	$T_{P\&R}$	S_{IR}	S_{DR}	S_{score}	S_R	$T_{P\&R}$	S_{IR}	S_{DR}	S_{score}	S_R	$T_{P\&R}$	S_{IR}	S_{DR}	S_{score}	S_R	$T_{P\&R}$	S_{IR}	S_{DR}
Design_116	55.33	99	0.56	9	11	32.69	54	0.61	6	9	17.72	40	0.44	5	8	28.60	45	0.64	5	9
Design_120	7.30	16	0.46	2	8	4.26	9	0.47	1	9	6.28	14	0.45	2	7	6.07	14	0.43	2	7
Design_136	23.08	60	0.38	6	10	6.08	16	0.38	2	8	3.45	10	0.35	1	10	2.93	9	0.33	1	9
Design_156	6.69	16	0.42	2	8	8.80	21	0.42	3	7	64.09	88	0.73	8	11	8.51	21	0.41	3	7
Design_176	98.54	110	0.90	11	10	36.62	60	0.61	10	6	52.33	90	0.58	9	10	31.20	54	0.58	6	9
Design_180	100.14	64	1.56	8	8	91.87	80	1.15	8	10	23.21	44	0.53	4	11	61.69	63	0.98	7	9
Design_190	36.20	90	0.40	6	15	35.79	54	0.66	6	9	24.78	80	0.31	10	8	20.62	45	0.46	5	9
Design_197	7.39	24	0.31	3	8	5.65	18	0.31	2	9	6.01	21	0.29	3	7	5.56	18	0.31	2	9
Design_227	21.31	56	0.38	7	8	9.62	24	0.40	3	8	8.41	21	0.40	3	7	7.98	21	0.38	3	7
Design_230	9.71	28	0.35	4	7	25.03	66	0.38	6	11	4.54	12	0.38	2	6	20.99	54	0.39	6	9
Average	36.57	56.30	0.57	5.80	9.30	25.64	40.20	0.54	4.70	8.60	21.08	42.00	**0.44**	4.70	8.50	**19.41**	**34.40**	0.49	**4.00**	**8.40**
Ratio	1.88	1.64	1.17	1.45	1.11	1.32	1.17	1.10	1.18	1.02	1.08	1.22	**0.91**	1.18	1.01	**1.00**	**1.00**	1.00	**1.00**	**1.00**

model further, we also implement dataset augmentation techniques, which included rotating the features and labels by $90°$, $180°$, and $270°$. Consequently, each benchmark contributes 120 sets of input features and labels, resulting in a total of 1200 sets across all benchmarks.

B. Performance of Congestion Prediction

In this subsection, we compare the performance of our congestion prediction model with several state-of-the-art ML-based models, including U-net [6], PGNN [7], PROS2.0 [8]. We meticulously replicated the functionalities of U-net, PGNN and PROS2.0 by adhering to the model described in their papers. To effectively measure the performance of our congestion prediction model, we adopt the following commonly used metrics [6]: accuracy (ACC), which quantifies the overall correctness of classifying each grid into the correct congestion level; coefficient of determination (R^2), which measures how well the variations in congestion levels are captured by the model; and normalized root mean square error (NRMS), which measures the quality of the predicted image.

As shown in Table I, our model outperforms other models in terms of average ACC, R^2 and NRMS. Notably, Our model is the only one that employs a hybrid CNN-transformer architecture, while the other three are based on pure CNN layers. Specifically, our model enhances the average ACC by 10.6%, 6.5% and 3.7%, R^2 by 8.1%, 5.2% and 3.4%, and improves NRMS by 28.2%, 21.4% and 12.8%, over U-net, PGNN, and PROS2.0, respectively. This superior performance is attributed to the incorporation of MFA block and transformer layers within our model, which adeptly extract global information from the grid-based features, thereby yielding a higher R^2 compared to the other models.

C. Performance of Routability-Driven Placement

Routability is a crucial metric for assessing the quality of FPGA placement. Table II presents the routability results of our proposed FPGA macro placement method with the winner teams (i.e., UTDA [11], SEU, and MPKU-imporve [16]) from the MLCAD 2023 FPGA macro placement contest [10]. Notably, UTDA and SEU are tied for first place in the contest, and MPKU-Improve is the improved version of MPKU after the contest. With the codes provided by the authors of these three teams, we were able to run all the methods on the same machine for a fair comparison.

As detailed in Table I, our method achieves superior results across four metrics: the initial routing score S_{IR}, detailed routing score S_{DR}, overall routability score S_R, and the final score S_{score}. Since T_{macro} of all methods is less than the specified 10 minutes, it does not affect the final score S_{score}, and we omit it from our comparison. MPKU-Improve achieves the shortest $T_{P\&R}$ because of its excellent performance in Design_116 and Design_180, which makes their average $T_{P\&R}$ lower compared to other methods. However, despite the average $T_{P\&R}$ of our methods is longer than MPKU-Improve, we surpass MPKU-Improve in both the average of S_R and S_{score}. Specifically, our method outperforms UTDA, SEU, and MPKU-Improve in the average routability score S_R by 64%, 17%, 22%, respectively. In terms of the final score S_{score}, our method surpasses UTDA, SEU, and MPKU-Improve by 88%, 32%, 8%, respectively. The experimental results confirm that our method achieves the best routability performance of FPGA macro placement and overall score among the contest winners.

VI. CONCLUSIONS

In this paper, we have proposed an MFA and transformer based congestion prediction model for routability-driven FPGA macro placement. We first designed a CNN component to extract multiscale features from grid-based placement, followed by MFA blocks that utilize the dual attention mechanism on both spatial and channel dimensions to enhance the representation of each multiscale feature. By incorporating MFA blocks and CNN's output at each skip connection layer, our model substantially enhances its capability to learn features and recover more precise congestion level maps. Besides, we adopted multiple transformer layers that employ dynamic attention mechanisms to extract global information, which can significantly improve the difference between various congestion levels and enhance the ability to identify these levels. Experimental results have shown that our model outperforms the existing congestion prediction models. Furthermore, our model can achieve the best routability and score among the contest winners when integrated into the macro placer based on DREAMPlaceFPGA.

REFERENCES

[1] Y. Meng, W. Li, Y. Lin, and D. Z. Pan, "elfplace: Electrostatics-based placement for large-scale heterogeneous fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 1, pp. 155–168, 2021.

[2] R. S. Rajarathnam, M. B. Alawieh, Z. Jiang, M. Iyer, and D. Z. Pan, "DREAMPlaceFPGA: An open-source analytical placer for large scale heterogeneous fpgas using deep-learning toolkit," in *Proceedings of IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 300–306, 2022.

[3] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proceedings of IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2007.

[4] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, "Ntuplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 12, pp. 1914–1927, 2014.

[5] 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 *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2018.

[6] H. Szentimrey, A. Al-Hyari, J. Foxcroft, T. Martin, D. Noel, G. Grewal, and S. Areibi, "Machine learning for congestion management and routability prediction within fpga placement," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 25, no. 5, pp. 1–25, 2020.

[7] K. Baek, H. Park, S. Kim, K. Choi, and T. Kim, "Pin accessibility and routing congestion aware drc hotspot prediction using graph neural network and u-net," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2022.

[8] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Young, "PROS 2.0: A plug-in for routability optimization and routed wirelength estimation using deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 1, pp. 164–177, 2022.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[10] I. Bustany, G. Gasparyan, A. Gupta, A. B. Kahng, M. Kalase, W. Li, and B. Pramanik, "The 2023 mlcad fpga macro placement benchmark design suite and contest results," in *Proceedings of ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, 2023.

[11] Z. Xiong, R. S. Rajarathnam, Z. Jiang, H. Zhu, and D. Z. Pan, "DREAMPlaceFPGA-MP: An open-source gpu-accelerated macro placer for modern fpgas with cascade shapes and region constraints," *arXiv preprint arXiv:2311.08582*, 2023.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, 2015.

[14] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3146–3154, 2019.

[15] Xilinx Corporation, "Zynq ultrascale+ device technical reference manual." Online, 2022.

[16] J. Mai, J. Wang, Y. Chen, Z. Guo, X. Jiang, Y. Liang, and Y. Lin, "OpenPARF 3.0: Robust multi-electrostatics based fpga macro placement considering cascaded macros groups and fence regions," in *Proceedings of IEEE International Symposium of Electronics Design Automation (ISEDA)*, pp. 374–379, 2024.