Accurate Timing Path Delay Learning using Feature Enhancer with Effective Capacitance

He Liu[1,2*], Shengkun Wu[2], Simin Tao[2], Biwei Xie[3,2], Xingquan Li[4,2,✉] and Ge Li[1]
[1]School of Electronic and Computer Engineering, Peking University, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[4]School of Mathematics and Statistics, Minnan Normal University, Zhangzhou, China
Email: *liuh@stu.pku.edu.cn, ✉fzulxq@gmail.com

2023 International Symposium of Electronics Design Automation (ISEDA) | 979-8-3503-0451-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISEDA59274.2023.10218429

***Abstract*—Accurate timing path delay calculation is a critical technique during the process of chip design. For widely used path delay calculation methods, cell delay is captured from look-up table (LUT) and wire delay is calculated by classical Elmore model, which can be considered as first-order-moment matching approximation, efficient and easy to use but ignoring the resistive shielding. The wire delay calculated by commercial static timing analysis (STA) tools is still opaque and such lower-order moments delay have large errors compared with commercial STA tool. In this work, to address these limitations, we propose a learning-based timing path delay and wire delay prediction model with feature enhancer. Our model utilize the feature enhancer to calculate the wire delays with different classical wire delay metrics not only Elmore model, Delay with two Moments (D2M) model but also Effective Capacitance Metric (ECM) model and Modified D2M (MD2M) model, which captures resistive shielding by modeling the load capacitance with effective capacitance instead of total capacitance. Furthermore, an effective trick about label is setting the ratio of path delay from commercial tool to traditional method as label, which is critical for improving generalization and accelerating the convergence rate.**

Experimental results on multi-corner wire delay prediction demonstrate that our model using feature enhancer with effective capacitance achieves an average rRMSE values of 0.4% for trained designs and 3.3% for unseen test designs.

***Index Terms*—Static timing analysis, delay learning, enhancer, Transformer, ResNet**

I. Introduction

For high performance chip design, timing closure is the most critical requirement [1]. To obtain a timing satisfied chip design, an effective scheme is that design flow should continually call the timing evaluator to measure the timing slack during the chip design flow [2], [3]. Consequently, a good timing evaluator can fast return an accurate timing measure, which can further be used to guide the next design iteration [4]. The accuracy of path delay calculation is critical for STA. The path delay consists of the cell delay captured using the look-up table with the index of the input slew and the output capacitance and the wire delay calculated by classical algorithms such as Elmore and D2M models. However, when considering crosstalk between adjacent wires, the wire delay calculation becomes more complex [5].

To obtain the timing information more accurately and quickly, A large number of researchers attempted to use machine learning-based approaches. Previous studies in academic have proposed a variety of machine learning based methods to predict the timing. The work of [6] develop a machine-learning method to correlate the divergence of path slack and improve the correlation between different signoff tools. In [7], the authors develop machine learning-based models of interconnect delay and slew for internal iSTA to slow the deviation in the endpoint slack and the offset-based model realizes the prediction for endpoint slack more accurately. Cheng $et\ al.$ in [8] developed a wire delay estimation model using XGBoost, which extracts the net topological features to characterize RC networks and proposes a loop breaking algorithm. The authors of [9] introduce a random forest model that reduce the pessimism and improve the prediction accuracy of net delay and slew at the pre-routing stage. Yang $et\ al.$ [10] propose a pre-routing path delay estimation model by employing the transformer network and residual model, which improves the accuracy and speed of the pre-timing prediction compared with other machine machine learning-based methodology. An end-to-end GNN model is proposed in [11] to predict the pre-routing net delay and pin arrival times without invoking additional STA tools. The work of [12] use the look-ahead RC network and random forest based timing model to improve the correlation of pre-routing timing prediction.

To accurately calculate timing path delay and wire delay, several delay metrics based on higher-order moments have been proposed. Which are typically more accurate than Elmore delay and simpler than SPICE simulation. However, they are still computationally expensive and challenging to utilize in the physical design optimization process. It is crucial to employ machine learning models and select the proper lower-order moments delays which are efficient and easily computable to capture the more accurate results in a acceptable short time. As the prediction works described above, basic physics information and simple enhanced features such as Elmore delay and D2M delay were utilized as input features. However, the resistance shielding effect was ignored in the calculation process. By incorporating more accurate enhanced features, a more accurate prediction results can be achieved. In this paper, we compute more relevant and precise features such as ECM and MD2M delay that take into account the resistance shielding effect using a feature enhancer for wire

delay learning. To accelerate the convergence rate and enhance generalization, the ratio of the ground truth delay to the delay calculated by feature enhancer is employed as the label of our learning model. The learning accuracy can be further improved by this network structure.

The major contributions of this work are summarized as follows:

- To build a high-accurate neural network model with strong generalization for path delay learning, basic electronic features (such as slew and capacitance, etc.) are combined with enhanced features, including cell delay from lookup table and net delay calculated by Elmore delay metric.
- For multi-corner wire delay learning, we utilize a feature enhancer to calculate wire delays with multiple traditional methods (Elmore, D2M, ECM, and MD2M) and their average value, taking into account resistive shielding by modeling load capacitance with an effective capacitance instead of total capacitance.
- The generalization of the trained model on unseen designs and convergence rate was greatly improved by using the ratio of PrimeTime path delay to traditional method instead of just PrimeTime path delay as the label.
- We improved the path delay and wire delay learning accuracy by using Transformer and ResNet with feature enhancer, and made the model more interpretable by explaining the principles of Transformer and ResNet.
- Experimental results of the path delay learning on open-source designs demonstrate that as for the slow corner, the performance of our model with feature enhancer in terms of average rRMSE values are 1.1% and 2.5% for trained designs and unseen designs, which are reduced by 1.55 $\times$ and 10.48 $\times$ compared with model trained without feature enhancer, respectively.
- For multi-corner wire delay prediction, our model achieves an average rRMSE values of 0.4% for trained designs and 3.3% for unseen designs.

The remainder of this paper is organized as follows. In Section II, we introduce the task of this work and select training features for learning. In Section III, our neural network model framework based on Transformer and ResNet are designed. Our experimental results and conclusions are presented in Sections IV and V, respectively.

II. Task and Features

In this section, we present the task of timing path delay and wire delay learning. A satisfying timing evaluator should achieve a accurate result in a short time. In this paper, we aim to train a neural network by a supervised learning procedure. For improving accuracy, we carefully select the basic features and enhance both the features and label in the learning model.

A. Basic Features

Selecting appropriate feature variables would not only reduce the training cost, but improve the effectiveness and performance of the neural network model. In this paper, we choose features listed in Table I as our basic input variables. Which consist of the timing and physical data extracted from circuit and library or calculated by some explicit formulations.

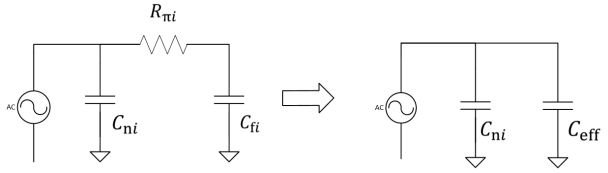

Fig. 1. Effective capacitance computatiaon based on a reduced π model.

TABLE I
Basic Features

Features Attributions	Features
Cell delay	Input transition load capacitance Signal polarity(rise/fall) Cell type Cell port name
Net delay	Output transition Resistance Capacitance

The transition time affects the drive strength of the fanout load cell on the same path. If the input transition time is large, that means the signal has more large delay through one stage. A stage consist of one cell and cell output pin connected net. The stage output transition affect next stage delay. For the signal polarity, since the asymmetry between the rise and fall polarity, which is affected by the different reaction sensitivity of device on rise and fall signal. Cell type and cell port name are related to the function and driven strength of the cell. Load capacitance affects the signal charge time, further affects the stage delay. And Resistance and capacitance play a significant role in interconnect delay.

For load capacitance, instead of modeling the load capacitance of each node by the sum of all capacitances C_L in the downstream parasitic RC tree [13] which ignoring the resistance shielding, we model the load capacitance by a single effective capacitance based on a effective π model, as depicted in Figure 1. The effective capacitance can be calculated in several steps. First, we apply the method in [4] to construte a π model. The input admittance $Y_i(s)$ of the circuit downstream of node i can be formulated in a Taylor series about $s = 0$ as follows:

$$Y_i(s) = y_{1,i}s + y_{2,i}s^2 + y_{3,i}s^3 + \cdots \tag{1}$$

where the $y_{1,i}$, $y_{2,i}$, $y_{3,i}$ are the moments of the admittance for node i, the π model of circuit downstream of node i can be calculated with the three moments:

$$R_{\pi i} = -\frac{y_{3,i}^2}{y_{2,i}^3}, \quad C_{\text{f}i} \quad = \frac{y_{2,i}^2}{y_{3,i}}, \quad C_{\text{n}i} = y_{1,i} - C_{\text{f}i} \tag{2}$$

Then the effective capacitance C_{eff} can be expressed as

$$C_{\text{eff}} = C_{\text{f}i}\left(1 - \text{e}^{-ED_i/\tau_i}\right) \tag{3}$$

where $\tau_i = R_{\pi i}C_{\text{f}i}$ and the ED_i means the Elmore delay from root node to node i.

Finally, the effective load capacitance expression is given by

$$C_{leff} = C_{\text{n}i} + C_{\text{eff}} \tag{4}$$

B. Feature Enhancer

To solve the problem of generalization and achieve higher accuracy, we explore more effective features as the input variables of model. After careful analysis, it would be a wise decision to incorporate the timing path results calculated by traditional cell delay and several wire delay methods as supplementary enhanced features for training the model.

1) Cell Delay: A widely used scheme is based on 2D (two-dimensional) look-up tables (LUT), it is called as nonlinear delay model (NLDM). The 2D LUT composes of two variables, input slew (transition time) and output capacitance. In this work, a bilinear interpolation or extrapolation method is used to compute the cell arc delay [2].

2) Elmore Delay: Wire delay can be accurately captured by electrical simulation or solving circuit equation by some accurate numerical methods. However, in order to quickly obtain a satisfactory solution, the Elmore delay model is commonly used to calculate the wire delay . The Elmore delay metric can be formulated as

$$D_{Elmore} = \sum_{i \in N} \left(R_i * \sum C_D \right) \tag{5}$$

where N is the set of all nodes in the RC tree, and R_i denotes the resistance of node i. C_D means the sum of the capacitances downstream from node i.

3) D2M Delay: D2M is a delay metric that is calculated with the two moments of impulse response, means "delay with two moments", which is performed more accurate at the far end of the RC tree. The D2M formula is given by

$$D_{D2M} = \frac{m_1^2}{\sqrt{m_2}} \ln 2 \tag{6}$$

where m_1 and m_2 are the first two moments of the impulse response.

4) ECM Delay: Unlike the Elmore delay metric ignores the resistance shielding, effective capacitance metric (ECM) uses the methods mentioned in Section II-A to calculate the effective capacitance instead of the sum of the downstream capacitances. ECM delay metric has the same structure as the Elmore delay metric, but more accurate. ECM metric can be described as:

$$D_{ECM} = \sum_{i \in N} \left(R_i * \sum C_{leff} \right) \tag{7}$$

where R_i denotes the resistance of node i. C_{leff} means the effective capacitance of the downstream capacitances from node i.

5) MD2M Delay: MD2M is a modified method of D2M delay metric. Compared with D2M delay metric, MD2M delay metric substitute the computing method of $m1$ from Elmore delay metric to ECM delay metric considering resistance shielding. MD2M metric can be given by

$$D_{MD2M} = \frac{m_1^{'2}}{\sqrt{m_2^{'}}} \ln 2 \tag{8}$$

where $m_1^{'}$ and $m_2^{'}$ are the modified first two moments of the impulse response by applying effective capacitance instead of the sum of the downstream capacitance as load capacitance of each node during the computing process.

C. Features and Label Selection

1) Timing Path Delay Learning: Timing path delay consists of cell delay and wire delay. In this work, we selected several basic physics features and timing features such as input slew (S_i), load capacitance (C_l), signal polarity ($SP_{r/s}$), cell type (CT), Cell port name (CP), and output slew (S_o). Then, we alleviated the feature enhancer to calculate the timing path delay which includes the traditional interpolation cell delay (D_{cell}) and wire delay captured by traditional Elmore delay metric (D_{Elmore}), and call the timing path delay calculated by traditional methods with the above-mentioned feature enhancer as PD_{FE}.

$$PD_{\text{pred}} = f\left(S_i, SP_{r/s}, CT, CP, S_o, C_l, PD_{FE}\right) \tag{9}$$

where PD_{pred} is the predicted timing path delay.

In this study, we apply the PrimeTime to generate the path delay as the ground truth label and call the delay as PD_{PT}. Our experiments showed that using PD_{PT} as the label alone achieved a good training accuracy but poor generalization on test cases. Statistical analysis demonstrate that different timing path delays are dispersive with large variance, but the data of PD_{PT}/PD_{FE} are distributed in a small range. According to the frequency principle [14], to obtain a centralized label distribution, we utilized the ratio PD_{PT}/PD_{FE} as label instead of PD_{PT}. It is worth mentioning that PD_{FE} refers to the arrive times (AT) of pins on each timing path when used as an enhanced input feature. Howerve, when applyed in the lable PD_{PT}/PD_{FE}, it denotes the path delay of each path.

2) Wire Delay Learning: In this study on wire delay learning, physics variables such as port names, resistances and capacitances of each wire are extracted from the SPEF file. These features were then used by feature enhancer to calculate the four wire delays D_{Elmore}, D_{D2M}, D_{ECM} and D_{MD2M} by the delay metrics mentioned above, respectively. Basic features and enhanced features are demonstrated in Table II. We use the above four wire delay and the average D_{Avg} of the above four as the input features to predict the wire delay D_{pred}.

TABLE II
BASIC FEATURES AND ENHANCED FEATURES

Category	Features
Basic features	Port name, Resistance, Capacitance
Feature enhancer_1	Elmore, D2M
Feature enhancer_2	Elmore, D2M, ECM, MD2M, Average

The same principle as the above label selection, we choose the ratio D_{Golden}/D_{Avg} of golden wire delay D_{Golden} to average wire delay D_{Avg} of the four methods above as the label.

$$D_{\text{pred}} = f\left(D_{Elmore}, D_{D2M}, D_{ECM}, D_{MD2M}, D_{Avg}\right) \tag{10}$$

III. LEARNING MODEL

A. Network Framework

In this paper, we propose a neural network model for path delay and wire delay learning with Transformer and ResNet. The overall modeling flow is depicted in Fig. 2.

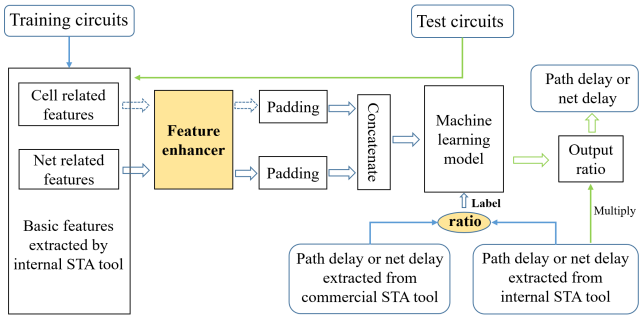

Fig. 2. Overall modeling flow.

Initially, the selected features of each path are extracted from timing reports. As for cell types and cell port names in the cell library should be encoded, which correspond to an unique id. Then, zero padding is employed on input features to guarantee consistent dimensions. Finally, the processed feature data was concatenated and fed into the encoder blocks of Transformer and then the ResNet.

B. Single-Head Self-Attention Transformer Encoder Block

In our work, the modified transformer encoder block consists of a single-head self-attention layer and two fully connected feed-forward networks. A residual connection is employed around each layer. The output of the encoder block is obtained after the addition and normalization process of the two feed forward networks.

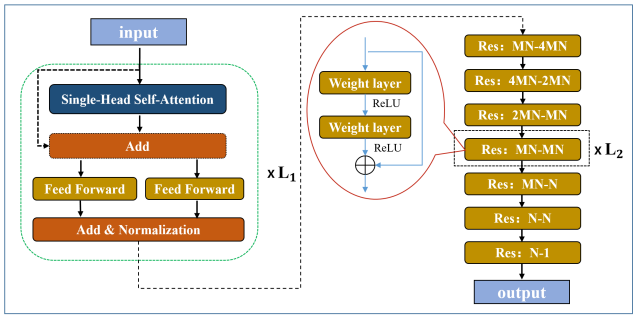

Fig. 3. Neural network architecture. Including the Encoder of Transformer and ResNet.

The attention function based on inner product is shown as below:

$$Attention = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \tag{11}$$

The three matrices Q_i, K_i, V_i, which are "queries", "keys", and "values" respectively. Attention can be considered as a function that maps the query and a set of key-value pairs to an output. d_k means the dimension of the key and value.

The single-head self-attention transformer model with q transformer encoder blocks is able to generate any desired polynomial of degree q of the input with no error with a finite number of free parameters [15]. Clearly, this result outperforms other classical deep learning models such as CNN with ReLU activation function which can only produce piecewise linear functions of the input and are unable to generate polynomials with no error. Based on this excellent approximate ability, single-head self-attention transformer encoder blocks are used to learn the comprehensive characteristic information and correlation between timing and basic features together with enhanced features. $L1$ denotes the layer number that can be modified.

C. Residual Network

As the depth of the network increases, the accuracy saturates and then drops off rapidly, such degradation problem can't be solved simply by stacking more layers. In ResNet, the sum of the output and input of each block is applyed as the input of the next layer. This specified operation enables neural network to be regarded as ensembles of relatively shallow networks [16].

To alleviate the degradation problem, the residual network model employs a shortcut-connection to train extremely deep architectures. The advantage of residual network is that it can avoid the vanishing gradient problem. The residual block structure and the residual network we adopted is shown in Fig. 3. $Res : MN - 4MN$ refers a residual block with MN dimensional input and $4MN$ dimensional output (the same below), where M represents the number of columns of the input matrix and N represents the number of rows. $L2$ indicates the layer number that can be adjusted.

IV. Experimental Results

In this work, we conduct two experiments to validate the useful of the feature enhancer. We implement our models using PyTorch and scikit-learn toolkit. All train and evaluate experiments are performed on a Linux machine with two NVIDIA A100 GPUs, 4 Intel Xeon Platinum 8380 CPUs at 2.3 GHz, and 1T RAM.

Experiment1. Timing path delay learning of single slow and fast corners with 130nm technology. For slow corner, the layer L_1 of Transformer is 5 and the layer L_2 of ResNet is 10. As for fast corner, L_1 and L_2 are modified as 4 and 10, respectively.

Experiment2. Wire delay learning of multi-corners with 28nm technology. The layer L_1 of the encoder blocks of Transformer is 12 and the layer L_2 of ResNet is 5.

A. Design of Experiment1 and Results

1) Experiment1 Setup: In this work, We utilized the internal STA tool with traditional delay calculation methods and commercial STA tool PrimeTime to produce timing reports on the open-source designs which vary in scale on SkyWater 130nm technology. The circuit are divided into training sets and testing sets, and the descriptive statistics of these benchmark circuits are summarized in Table III. The upper four benchmarks statistics were split into 2-parts, 80% of the circuits statistics used for training , the remaining 20% along with the lower two benchmarks statistics utilized for testing. We tested the open-source benchmarks in both the fast and slow corners, which correspond to different PVT (Process, Voltage, Temperature) conditions.

2) Comparison on Features and Label: To confirm the effectiveness of our feature enhancer and label scheme on path delay learning, we compare the accuracy of the same neural network with different features and labels. As is shown in

TABLE III
BENCHMARK STATISTICS.

Benchmarks	#Cell	#Net	Slow Corner		Fast Corner	
			Train #Path	Test #Path	Train #Path	Test #Path
picorv32	14033	14057	123344	30837	86804	21701
jpeg_encoder	49915	62281	31332	7834	31861	7966
aes_cipher	18672	18926	64027	16007	68577	17145
ibex_core	14199	15023	7086	1772	64419	16105
riscv	11595	11655	0	29742	0	59385
gcd	359	388	0	15799	0	12626
total	108773	122300	225789	101991	251661	134928

Table IV, the accuracy of the results were evaluated with the value of relative root means square error (rRMSE). Furthermore, the average absolute error (AAE) of path delay and wire delay are provided. The data in column "Ratio$_3$" are the ratio of data in "w.o.FE" to data in "w.FE".

From column "w.o.FE" in Table IV, it is evident that the model without feature enhancer and use PD_{PT} as the label achieves accurate results on the testing set of trained designs but fails to perform well on unseen ones. It suggests that the trained model lacks generalization on unseen cases. Furthermore, the results in column "w.FE" of Table IV are overall satisfactory. Specially, trained model with feature enhancer and PD_{PT}/PD_{FE} as label obtain satisfactory average rRMSE/AAE values of 0.025/84.72 and 0.016/35.08 on unseen cases in slow and fast corners, respectively. The column "Ratio$_3$" highlights the advantage of the models trained with enhanced features and enhanced label PD_{PT}/PD_{FE}. This model reduces the rRMSE/AAE values by 10.48/5.49 times and 4.38/4.41 times compared with the model trained without feature enhancer on slow and fast corners, respectively.

3) Comparison on Models: Furthermore, to evaluate the effectiveness of the proposed neural network, we design another experimental comparison. The values in column "Ratio$_1$" and "Ratio$_2$" are the ratio between data in "RF" and "LGBM" to data in "w.FE", respectively. As for slow corner, our model reduce the rRMSE and AAE values by 19.71 $\sim$ 1.07 times and 27.82 $\sim$ 1.04 times compared with the random forest-based model meanwhile 18.29 $\sim$ 0.7 times and 25.80 $\sim$ 0.67 times compared with the LightGBM model, respectively.

For clearity, the relerative error distribution of path delay predicted by different models are compared for unseen circuit "gcd" of fast corner, which is plotted in Fig. 4. The red line clearly shows that our model with feature enhancer and PD_{PT}/PD_{FE} as label outperforms the other models, as a higher proportion of path errors are concentrated near zero point.

These comparisons affirm the superiority of our neural network based on Transformer and ResNet and trained with basic features together with feature enhancer listed in II-C as input and PD_{PT}/PD_{FE} as label, Which demonstrates a remarkable efficiency and accuracy.

B. Design of Experiment2 and Results

1) Experiment2 Setup.: To validate the effectiveness of our model which utilizes the feature enhancer, we conduct experiments on wire delay learning for multi-corners. The accuracy is measured by the rRMSE/AAE values on designs

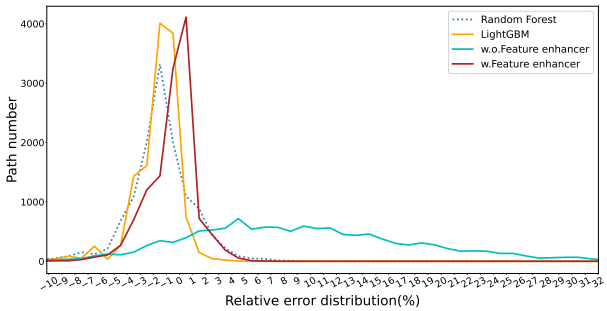

Fig. 4. Error distribution of different models on unseen circuit "gcd".

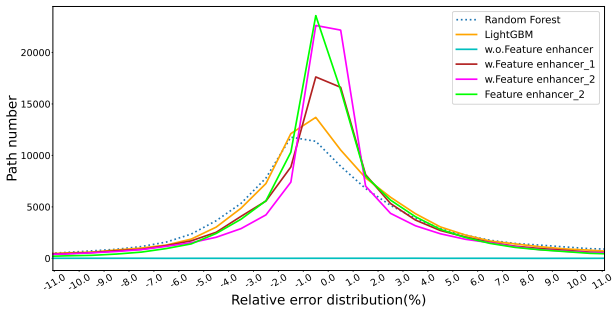

Fig. 5. Relative error distribution of different models and features on unseen benchmark "Group4".

from 4^{th} integrated circuit EDA elite challenge, which is based on 28nm technology.

2) Comparison on Features and Label: The data in column "Ratio$_3$", "Ratio$_4$" and "Ratio$_5$" are the ratio of data in "w.o.FE", "w.FE$_1$" and "w.FE$_2$" to data in "FE$_2$", respectively.

Table V demonstrates that trained model with only feature enhancer_2 listed in Table II and D_{Golden}/D_{Avg} as label achieved the best average rRMSE/AAE values of 0.004/0.23 and 0.033/1.5 for trained and unseen corners, respectively. Which achieves a reduction of average rRMSE/AAE values by 1.25/1.04 times and 1.27/1.17 times compared with the model trained with basic features together with feature enhancer_2, and up to 5.75/2.43 times and 2.64/1.35 times compared with model trained with basic features and feature enhancer_1, and up to 51.75/40.04 times and 6.30/6.45 times compared with model trained only with basic features and basic label D_{Golden}.

3) Comparison on Models: The data in column "Ratio$_1$" and "Ratio$_2$" show the ratio of the "RF" and "LGBM" data to the "FE$_2$" data, respectively. Our model reduce the rRMSE and AAE values by 9.25 $\sim$ 2.73 times and 5.19 $\sim$ 2.27 times compared to the random forest-based model meanwhile 5.5 $\sim$ 2.24 times 4.48 $\sim$ 1.76 times compared with LightGBM model, respectively.

For clearity, the relerative error distribution of wire delay predicted by different models are compared for unseen circuit "Group4". In Fig. 5, the green line clearly shows the advantage of our model with feature enhancer_2 only and D_{Golden}/D_{Avg} as label over the model with basic features and the random forest, LightGBM-based model, as a higher proportion of path errors are concentrated near the zero point.

In addition, the feature enhancer costs 61.05s on average for the designs to calculate all of the aforementioned enhanced

TABLE IV
THE COMPARISON OF LEARNING ACCURACY OF SINGLE CORNER BETWEEN DIFFERENT LEARNING MODELS(RRMSE / AAE).

	Design	Accuracy of slow corner (rRMSE / AAE (ps))							Accuracy of fast corner (rRMSE / AAE (ps))						
		RF	LGBM	w.o.FE	w.FE	$Ratio_1$	$Ratio_2$	$Ratio_3$	RF	LGBM	w.o.FE	w.FE	$Ratio_1$	$Ratio_2$	$Ratio_3$
Train	picorv32	0.008/ 8.10	0.007/5.06	0.016/23.02	0.007/5.61	1.14/1.44	1.0/0.90	2.29/4.10	0.007/2.60	0.008/4.88	0.021/18.83	0.012/11.14	0.58/0.23	0.67/0.44	1.75/1.69
	jpeg_encoder	0.075/170.66	0.069/147.85	0.015/29.83	0.014/22.86	5.36/7.47	4.93/6.47	1.07/1.30	0.039/25.38	0.039/24.28	0.033/23.76	0.032/22.10	1.22/1.15	1.22/1.10	1.03/1.08
	aes_cipher	0.046/83.10	0.046/84.09	0.022/30.48	0.018/20.37	2.56/4.08	2.56/4.13	1.22/1.50	0.023/25.46	0.023/23.46	0.018/22.70	0.017/21.68	1.35/1.17	1.35/1.08	1.06/1.05
	ibex_core	0.138/1053.81	0.128/977.36	0.007/37.81	0.007/37.88	19.71/27.82	18.29/25.80	1.0/0.998	0.022/25.0	0.021/19.19	0.023/32.36	0.021/25.83	1.05/0.97	1.0/0.74	1.10/1.25
Average		0.032/84.75	0.030/77.80	0.017/26.54	0.011/13.20	2.91/6.42	2.73/5.89	1.55/2.01	0.019/17.45	0.019/16.06	0.022/23.97	0.018/19.16	1.06/0.91	1.06/0.84	1.22/1.25
Test	riscv	0.029/125.04	0.019/79.94	0.133/559.52	0.027/119.70	1.07/1.04	0.70/0.67	4.93/4.67	0.021/59.01	0.020/54.10	0.058/164.08	0.015/39.57	1.40/1.49	1.33/1.37	3.87/4.15
	gcd	0.028/23.33	0.025/20.75	0.361/286.60	0.022/18.88	1.27/1.24	1.14/1.10	16.41/15.18	0.025/20.39	0.025/19.43	0.128/109.95	0.019/13.98	1.32/1.46	1.32/1.39	6.74/7.86
Average		0.0286/89.75	0.021/59.41	0.212/464.84	0.025/84.72	1.14/1.06	0.84/0.70	10.48/5.49	0.022/52.24	0.021/48.02	0.070/154.59	0.016/35.08	1.38/1.49	1.31/1.37	4.38/4.41

TABLE V
THE COMPARISON OF LEARNING ACCURACY OF MULTI CORNERS BETWEEN DIFFERENT LEARNING MODELS (RRMSE / AAE).

	Design	Net	Wire path	Accuracy of multi-corners (rRMSE / AAE (ps))											FE runtime (s)
				RF	LGBM	w.o.FE	$w.FE_1$	$w.FE_2$	FE_2	$Ratio_1$	$Ratio_2$	$Ratio_3$	$Ratio_4$	$Ratio_5$	
Train	Group0	53599	99465	0.033/0.87	0.021/0.81	0.218/8.70	0.023/0.48	0.005/0.21	0.005/0.19	6.6/4.58	4.2/4.26	43.6/45.79	4.6/2.53	1.0/1.11	69.7
	Group1	53599	100227	0.035/1.04	0.021/0.89	0.195/9.28	0.026/0.56	0.005/0.23	0.005/0.23	7.0/4.52	4.2/3.87	39.0/40.35	5.2/2.43	1.0/1.0	57.9
	Group2	53599	98871	0.033/1.01	0.020/0.91	0.199/8.79	0.023/0.56	0.005/0.24	0.004/0.23	8.25/4.39	5.0/3.96	49.75/38.22	5.75/2.43	1.25/1.04	65.4
	Group5	53599	98460	0.037/1.39	0.022/1.11	0.191/10.20	0.023/0.70	0.004/0.30	0.004/0.28	9.25/4.96	5.5/3.96	47.75/36.43	5.75/2.50	1.0/1.07	47.9
	Group6	53599	99539	0.034/1.25	0.021/1.09	0.195/10.25	0.023/0.68	0.004/0.29	0.004/0.28	8.5/4.46	5.25/3.89	48.75/36.61	5.75/2.43	1.0/1.04	47.8
	Group7	53599	98429	0.033/1.01	0.021/0.91	0.202/8.64	0.022/0.54	0.005/0.24	0.004/0.22	8.25/4.60	5.25/4.14	50.5/39.27	5.5/2.46	1.25/1.09	65.0
	Group8	53599	99353	0.032/0.75	0.022/0.71	0.248/9.01	0.023/0.40	0.005/0.16	0.005/0.16	6.4/4.69	4.4/4.44	49.6/56.31	4.6/2.5	1.0/1.0	67.9
	Group9	53599	98413	0.036/1.09	0.022/0.94	0.207/8.78	0.023/0.54	0.005/0.23	0.004/0.21	9.0/5.19	5.5/4.48	51.75/41.81	5.75/2.57	1.25/1.10	67.1
Average				0.034/1.05	0.021/0.92	0.207/9.21	0.023/0.56	0.005/0.24	0.004/0.23	8.5/4.57	5.25/4.0	51.75/40.04	5.75/2.43	1.25/1.04	61.1
Test	Group3	53599	100413	0.094/3.76	0.076/2.92	0.205/9.84	0.083/2.20	0.042/1.91	0.032/1.61	2.94/2.34	2.38/1.81	6.41/6.11	2.59/1.37	1.31/1.19	59.6
	Group4	53599	97956	0.090/3.15	0.074/2.45	0.211/9.50	0.090/1.86	0.042/1.58	0.033/1.39	2.73/2.27	2.24/1.76	6.39/6.83	2.73/1.34	1.27/1.14	62.2
Average				0.092/3.46	0.075/2.69	0.208/9.67	0.087/2.03	0.042/1.75	0.033/1.5	2.79/2.31	2.27/1.79	6.30/6.45	2.64/1.35	1.27/1.17	60.9

features. These results demonstrates that feature enhancer_2 with ECM metric and MD2M metric that take resistance shielding into account contribute to the improvement in accuracy within a reasonable runtime.

V. CONCLUSIONS

In this work, we improved the accuracy of timing path delay and wire delay learning by capturing cell delay with LUT and wire delay calculated by not only Elmore and D2M delay metrics but ECM and MD2M delay metrics, which consider the resistance shielding that using the effective capacitance instead of the sum of the capacitances to ground with feature enhancer. To further improve the learning performance and accurate convergence rate, we employed the ratio of the ground truth delay to the delay calculated with feature enhancer as the label for timing path delay and wire delay learning. Experiments results on real world designs show that the enhanced features and label result in a remarkable generalization and prediction accuracy for path delay and wire delay compared with the ground truth value from commercial tool.

ACKNOWLEDGMENT

This work is supported in part by the Major Key Project of PCL (PCL2021A08), and Shenzhen Fundamental Research Program (GXWD20201231165807007-20200806163656003).

REFERENCES

[1] B. Jayaram and C. Rakesh, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer Science & Business Media, 2009.

[2] Z. Guo, T.-W. Huang, and Y. Lin, "Gpu-accelerated static timing analysis," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[3] C. Guth, V. Livramento, R. Netto, R. Fonseca, J. L. Güntzel, and L. Santos, "Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, pp. 141–148, 2015.

[4] Kashyap, C. V, C. J. Alpert, and A. Devgan, "An" effective" capacitance based delay metric for rc interconnect," in *IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000. IEEE/ACM Digest of Technical Papers (Cat. No. 00CH37140)*, pp. 229–234, IEEE, 2000.

[5] T.-C. Lee and Y.-L. Li, "Incremental timing-driven placement with approximated signoff wire delay and regression-based cell delay," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2434–2446, 2019.

[6] S. S. Han, A. B. Kahng, S. Nath, and A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2014.

[7] A. B. Kahng, S. Kang, H. Lee, S. Nath, and J. Wadhwani, "Learning-based approximation of interconnect delay and slew in signoff timing tools," in *2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1–8, IEEE, 2013.

[8] H.-H. Cheng, I. H.-R. Jiang, and O. Ou, "Fast and accurate wire timing estimation on tree and non-tree net structures," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[9] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine Learning-Based Pre-Routing Timing Prediction with Reduced Pessimism," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2019.

[10] T. Yang, G. He, and P. Cao, "Pre-routing path delay estimation based on transformer and residual framework," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 184–189, IEEE, 2022.

[11] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proceedings of the 59th Annual Design Automation Conference 2022*, ACM, 2022.

[12] X. He, Z. Fu, Y. Wang, C. Liu, and Y. Guo, "Accurate timing prediction at placement stage with look-ahead rc network," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, p. 1213–1218, 2022.

[13] M.-C. Kim, J. Hu, and N. Viswanathan, "Iccad-2014 cad contest in incremental timing-driven placement and benchmark suite: Special session paper: Cad contest," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 361–366, 2014.

[14] Z.-Q. J. Xu and H. Zhou, "Deep frequency principle towards understanding why deeper learning is faster," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10541–10550, 2021.

[15] Z. Fang, Y. Ouyang, D.-X. Zhou, and G. Cheng, "Attention enables zero approximation error," *arXiv preprint arXiv:2202.12166*, 2022.

[16] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in NIPS*, vol. 29, 2016.