Pre-Routing Path Delay Estimation Based on Transformer and Residual Framework

Tai Yang, Guoqing He, Peng Cao
National ASIC System Engineering Technology Research Center, Southeast University, Nanjing, China
caopeng@seu.edu.cn

2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC) | 978-1-6654-2135-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASP-DAC52403.2022.9712484

Abstract - Timing estimation prior to routing is of vital importance for optimization at placement stage and timing closure. Existing wire- or net-oriented learning-based methods limits the accuracy and efficiency of prediction due to the neglect of the delay correlation along path and computational complexity for delay accumulation. In this paper, an efficient and accurate pre-routing path delay prediction framework is proposed by employing transformer network and residual model, where the timing and physical information at placement stage is extracted as sequence features while the residual of path delay is modeled to calibrate the mismatch between the pre- and post-routing path delay. Experimental results demonstrate that with the proposed framework, the prediction error of post-routing path delay is less than 1.68% and 3.12% for seen and unseen circuits in terms of rRMSE, which is reduced by 2.3~5.0 times compared with exiting learning-based method for pre-routing prediction. Moreover, this framework produces at least three orders of magnitude speedup compared with the traditional design flow, which is promising to guide circuit optimization with satisfying prediction accuracy prior to time-consuming routing and timing analysis.

KEYWORDS

Pre-routing delay estimation, Transformer network, Residual model

I. Introduction

Timing estimation and optimization are iteratively performed throughout the entire logic and physical design flow, which have become the bottleneck of design flow due to the frequent interaction with static timing analysis[1-2]. As the design flow gets closer to tape-out, the updated circuit timing faces nonnegligible mismatch between each stage of design flow, posing severe challenges for circuit optimization. For example, post-routing delays differ from placement delays mainly due to factors such as net topology, layer assignment and congestion. In order to expedite the pace of timing closure, a pre-routing timing estimation engine is required to predict the circuit timing effectively with satisfying accuracy.

The fast pre-routing estimation approaches have been studied for decades with traditional mathematical models[3-7]. However, most of them focus mainly on the wire length/delay instead of net delay or path delay. In practice, the impact of routing to the cell delay is much more significant than that of wire delay, which can be demonstrated in Fig. 1 for a practical industrial circuit implemented with near 40K nets in total under the process of TSMC 28nm technology. As shown in Fig. 1, the average increase rate of net delay after routing is listed for all corresponding types of cells in the circuit, which is contributed by the increase of cell delay and additional wire delay due to routing. With the net delay increase ranging from 1.03 to 2.75 times after routing, over 84%~99% of the increase is contributed by cell delay, which is because the parasitic capacitance of the wire degrades the cell delay much more than the additional wire delay itself.

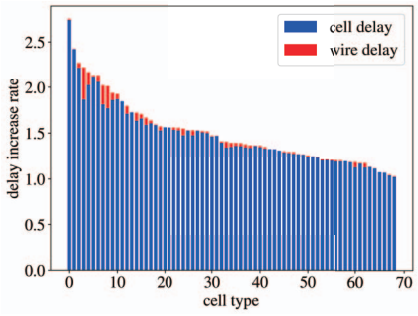

Fig. 1. Average increase ratio of net delays between routing and placement stages for all types of cells.

In order to achieve efficient and accurate timing estimation during design flow, plenty of researches have been devoted in machine learning-based methods. By exploiting the correlation of timing analysis results in different contexts, the learning algorithms including linear regression, random forest (RF), support vector machine (SVM) are adopted to predict the circuit timing for unobserved corners by that from observed corners[8], for PBA mode by that of GBA mode[9] and for SI mode based on non-SI mode timing reports[10].

In recent years, the learning-based methods have been extended in the application of pre-routing timing prediction [1-2,11-12]. In [11], a wire timing estimation is proposed and trained by XGBoost, where the topological features are extracted to capture the characteristics of RC networks and non-tree nets are converted to tree-structured nets with a loop breaking algorithm. In spite of the accurate wire delay prediction, it is not applicable for post-routing circuit delay prediction without the consideration of the impact of wire capacitance to cell delay. To reduce the pessimism of commercial pre-routing timing estimation tool, a net-based delay/slew model is introduced in [1] using learning algorithms including lasso regression, neural network and random forest, and then the overall circuit timing is obtained through PERT traversals, which may suffer from error accumulation for circuit path and additional computational complexity. A learning based parasitic estimation method is presented in [12] with the similar algorithms used in [1] for pre-layout analog circuit design. Considering the contribution of crosstalk to signal integrity and timing, a crosstalk prediction framework is proposed in [2] based on learning methods which can quickly identify the crosstalk prone nets at placement stage without any routing information by leveraging XGBoost and graph-based techniques.

In spite of the prior works for pre-routing prediction, the wire- or net-oriented timing prediction limits the accuracy and efficiency of the learning-based approaches due the following reasons. First, the wire delay and net delay are predicted

individually through circuit path, which neglects the delay correlation along the path due to slew propagation. Second, the post-routing path delay is obtained by accumulating the estimated net delays, leading to prediction error accumulation as well as computational complexity increase.

To tackle the above issue, an efficient and accurate pre-routing path delay prediction framework is proposed in this work by employing transformer network and residual model, where the timing and physical information at placement stage is extracted as sequence features for transformer network while the residual model is utilized to calibrate the mismatch between the pre- and post-routing path delay.

The main contributions are summarized as follows.

- This is the first study to predict pre-routing path delay based on transformer network, to the best of our knowledge, which exploits the correlations of the timing and physical information through circuit path by its multi-head self-attention mechanism.
- Owing to the physical-guided residual model, the path delay mismatch between the placement and routing stages are learned by extracting the complex characteristics from input features, leading to significant precision improvement.
- The application of path-based prediction model outperforms the net-based models remarkably in terms of accuracy, especially for unseen circuits, and speeds up by at least three orders of magnitude compared with commercial tool.

The rest of the paper is organized as follows: Section II presents the related preliminaries. Section III and IV present the feature selection and the proposed prediction framework in detail. Experimental results and discussion are given in Section V, followed by conclusion in Section VI.

II. Preliminaries

A. Transformer Network

The significant limitation of traditional sequential calculations, such as convolution neural networks (CNN) or recurrent neural networks (RNN), is time-consuming. In order to reduce sequential computation, transformer network is proposed by [13] and it is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNN or convolution. In spilt of how far they are in the input or output sequence, self-attention enables the network to learn dependencies between distant positions in the sequence and calculate its representation.

It is more accurate and more parallelizable than complex recurrent or convolutional neural networks. Hence, in recent years, transformer has been applied to different fields, such as music genre classification[14], machine translation[15], session-based recommendation[16] and risk prediction on electronic health records[17].

B. Residual Model

Residual modeling is a very common approach for incorporating physical knowledge in machine learning models, where a machine learning model is used to predict the errors made by a physics-based model [18]. In formula (1) and (2), the input D are physically related to a target variable of interest Y. The physics-based numerical model, $f_{PHY}: D \rightarrow Y_{PHY}$ is used to simulate the value of the target variable. The outputs of the physics-based model Y_{PHY} is combined with the input features D as all inputs of the machine learning model f_{HPD} to calibrate the physics-based model and predict the target variable.

$$f_{PHY} : D \rightarrow Y_{PHY} \tag{1}$$

$$f_{HPD} : X = [D, Y_{PHY}] \rightarrow Y \tag{2}$$

The basic goal of *HPD* model is to combine physics-based model and machine learning model so as to overcome their complementary deficiencies and leverage information in both physics and data. The *HPD* modeling achieve an even lower value of test RMSE than that of machine learning [18-20].

III. Feature Selection

Feature selection is of vital importance for the effectiveness of machine learning based models. In this work, the features are extracted from the net reports and timing reports by PrimeTime at placement stage. Besides the pre-routing path delay, the related timing and physical information for each net of a data path are extracted as feature sequences. By taking the N-stage data path depicted in Fig. 2 as an example, the features consist of the timing and physical information for the N-stage net sequence, where the timing information includes the transition time, cell delay and signal polarity of each net while the physical information includes the cell type, the location of cell pin and the load capacitance of the net. All the extracted features are listed in Table I.

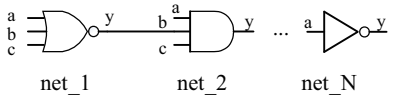

Fig. 2. An example of net sequence in data path.

IV. Pre-routing Path Delay Prediction Framework

A. Overview

In order to predict the post-routing path delay at placement stage, a prediction framework is proposed based on transformer network and residual model, as illustrated in Fig. 3.

Initially, the timing and physical features extracted from the net sequence of data path are pre-processed to be compatible with the following transformer network except the path delays at placement stage. During this step, each continuous feature is discretized by binning operation while each discrete feature is performed with a tokenizer to establish the mapping between the corresponding categorical variable and numbers. After that, zero padding is carried out to keep the dimensions of the input features as a constant, which is determined by the longest path in the data set.

TABLE I
Summary of Features

Features	Data attribute	Description
Pre-routing path delay	Cont.	Pre-routing path delay represents the path delay without parasitic parameter effect. It also provides a baseline for post-routing path delay prediction.
Timing feature sequences		
Input/output transition	Cont.	Input transition is generally related to the cell delay and the output transition, which is further involved with the delay of the subsequent cell.
Pre-routing Cell delay	Cont.	The delay on input pin is wire delay and the delay on output pin is cell delay, which are the most direct timing characteristic.
Signal polarity (r/f)	Disc.	For cell delays, rising cell delay and falling cell delay are different even if the input transition time and output load are the same.
Physical feature sequences		
Cell type	Disc.	Cell type is related to the function and driven strength of the cell.
Capacitance	Cont.	Cell delay is generally proportional to the load capacitance, which includes only pin cap. at placement stage.
Location x	Cont.	The coordinates (x, y) of each pin can measure the cell distance in a data path, which affect routing, buffering, total load capacitance and thus delay/transition time.
Location y	Cont.	

Cont.: Continuous data; Disc.: Discrete data

Then 3 transformer networks (in *transformer*$_i$, *i=1 or 2*) are established to perform the pre-processed feature sequences. One is for the concatenation of all timing and physical features to capture the comprehensive information of data path. The other 2 networks are respectively responsible for transition and pre-routing cell delay features since they are considered as the dominant relevance with the post-layout path delay. A pooling layer is used after each transformer network to convert the output from three dimensions to two dimensions while the subsequent dense and dropout layers are used to reduce the number of parameters and possibility of overfitting. The output of the dense and dropout layers for feature sequences are concatenated with the corresponding output for the pre-routing path delay before reduce dimension through another dense layer.

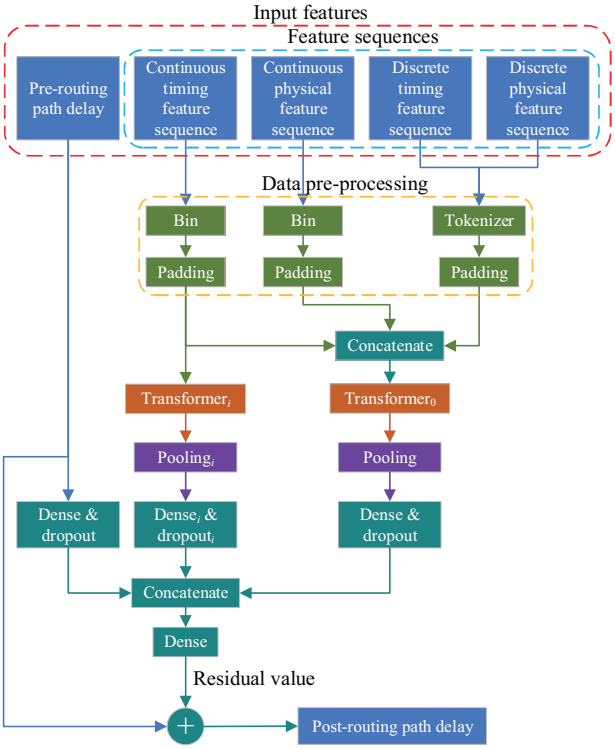

Fig. 3. Overview of the prediction framework.

Finally, by considering the output of last dense layer as the residual value of the pre- and post-routing path delay, the residual model is constructed to predict the post-layout path delays, where the pre-routing path delay can be considered as the baseline of physical model for the post-layout path delay.

B. *Transformer Network*

In this work, the encoder part of transformer is used to learn the characteristic of a data path and the correlation of the timing and physical information through circuit path by its multi-head self-attention mechanism. As Fig. 4 shows, the processed features go through the input embedding and position embedding layers and the results is the input of transformer encoder network. The encoder layer has two sub-layers, which are the multi-head self-attention mechanism and a position-wise fully connected feed-forward network. A residual connection is employed around each layer, followed by layer normalization. All of them are described in detail as below.

(1) Input embedding and position embedding: A learned embedding layer is applied to convert the processed input sequence to vectors of dimension d_{model}. Then, a position embedding (PE) layer based on the below trigonometric functions is used to help the network to understand the relative and absolute position information of the input sequence, where *pos* is the position and *m* represents dimension. Each feature has its own embedding layer. The matrix *X* in Fig. 4 is the concatenation of embedding output of all features.

$$PE_{(pos,2m)} = \sin\left(\frac{pos}{10000^{\frac{2m}{d_{model}}}}\right) \tag{3}$$

$$PE_{(pos,2m+1)} = \cos\left(\frac{pos}{10000^{\frac{2m}{d_{model}}}}\right) \quad (4)$$

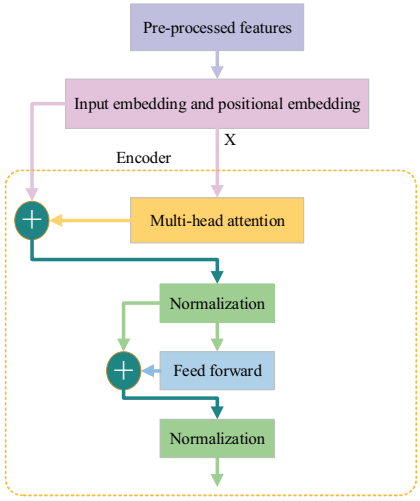

Fig. 4. Encoder of transformer network.

(2) Multi-head self-attention mechanism: Fig. 5 is the structure of multi-head self-attention mechanism. The input is the concatenation of embedding layers' output X. The $2j$-1 th and $2j$ th row vector in X is the representation of the j th cell input and output pin, respectively, in a data path as Fig. 5 (a) depicts. First of all, linearly project the input X h times with different, learned linear projections to obtain Q_i, K_i and V_i, with d_k, d_k and d_k, dimensions, respectively, which is called h-head self-attention. $i \in [1, h]$ and i is the integer.

Then, as shown in Fig. 5 (b), the attention function is performed in parallel on each of these projected versions of queries(Q_i), keys(K_i) and values(V_i), yielding d_k-dimensional Z_i. The attention function based on scaled dot-product is shown as below. It computes the dot products of Q_i with all K_i, divides each by $\sqrt{d_k}$, and apply a *softmax* function to get the weights on V_i.

$$Attention = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i \quad (5)$$

Consistent with X, the $2j$ th row vector in Q_i, K_i and V_i is the representation of the j th cell output pin in a data path, too. Therefore, in Fig. 5 (b), after the *softmax* function, $Correlation\_Matrix_{(a,b)}$, which is the values of row a and column b in *Correlation_Matrix*, represents the correlation between pin_a and pin_b . Owning to this, the proposed model can capture the correlation among all pins in a data path in timing and physical aspects with global perspective.

$$\begin{aligned} MultiHead &= Concat(head_1,...,head_h)W^O \\ where\ head_i &= Attention(XW_i^Q, XW_i^K, XW_i^V) \end{aligned} \quad (6)$$

At last, concatenate Z_1 to Z_h and then projected the results by W^O, resulting in the final values, as Fig. 5 (c) illustrates.

To sum up, the overall computing process of multi-head attention is described in Fig. 5, where X is the output of embedding layer, i=1,2,…,h, $d_{model} = hd_k$.

(3) Position-wise fully connected feed-forward network: A feed-forward layer gets the results from previous self-attention layer. The values of different heads can be performed in parallel in feed-forward layer. It consists of two linear transformations with an activation function of ReLU in between.

$$FFN(t) = \max(0, tW_1 + b_1)W_2 + b_2 \quad (7)$$

where t is the input of the feed-forward network.

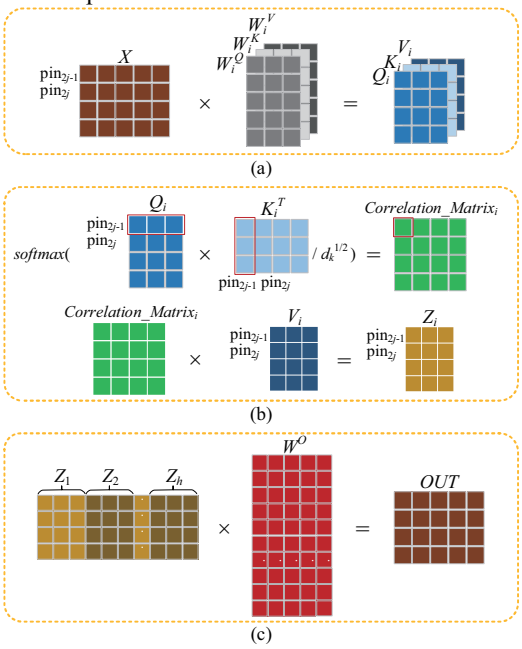

Fig. 5. h-head self-attention mechanism. (a) h times linearly projections (b) scaled dot-product attention (c) one time linearly projection

(4) Add and normalization: The add and normalization layer is used to solved the problem of vanishing and exploding gradients. The functions are described as below.

$$\begin{aligned} &LayerNorm(t + MultiHead(t,t,t)) \\ &or \\ &LayerNorm(t + FFN(t)) \end{aligned} \quad (8)$$

where t is the input of multi-head attention layer or the input of feed-forward layer.

V. Experimental Results and Discussion

A. Experiment Setup

The proposed pre-routing path delay prediction framework was implemented in Python with the toolkits of keras and scikit-learn and compared with the competitive learning-based models including RF[1] and CNN[21]. When using RF model, the circuit features were selected to be consistent with those utilized in [1] and the post-routing path delay was estimated by accumulating the predicted net delays as in [1]. For CNN model, it was built with three one-dimensional convolution layers followed by pooling layers as in [21], where the input features are the same with that of the transformer layer in this framework. In this work, heads of each transformer is 4, d_k in timing only transformer is 16 and d_k in timing and physical transformer is 16×7.

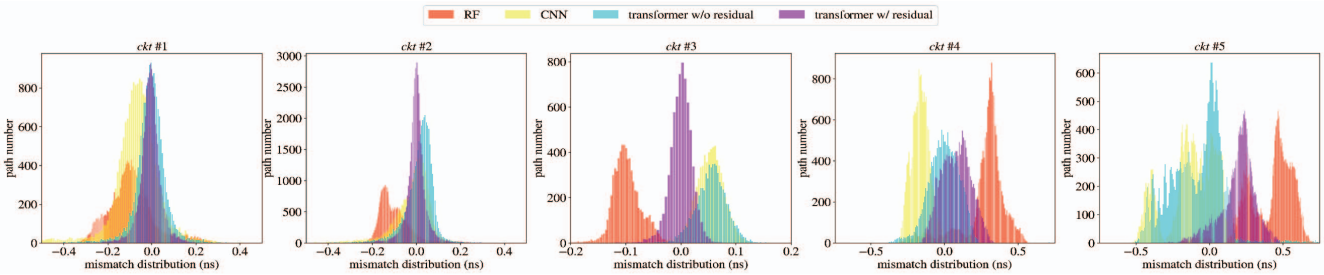

Fig. 6. Mismatch distribution of different models.

Since most benchmark circuits are relatively small-scale for the path delay calibration between the placement and routing stages, several median- and larger-scale industrial circuits were used to validate the prediction of the proposed framework and other models, which were designed with the process of TSMC 28nm technology. As summarized in Table II, 3 designs are used as seen circuits and the other 2 designs are unseen. Over 150K and 105K circuit paths are chosen for training and testing, in which nearly 40K paths are obtained from unseen circuits.

B. *Accuracy Comparison*

The prediction accuracies of the proposed frameworks and the competitive learning-based models are compared in Fig. 7 in terms of rRMSE (relative Root Mean Squared Error) by comparing the predicted post-routing path delays with the actual delay from timing reports after routing. It can be seen that owing to the transformer network and residual model utilized in the proposed framework, the prediction error is less than 1.68% and 3.12% for all seen and unseen circuits respectively, which are reduced by 2.3~5.0 times compared with the RF-based pre-routing prediction model and up to 2.9 times when comparing with CNN model respectively. Moreover, as compared in Fig. 7, the residual model enhances the prediction of transformer network by at least 30% and 10% accuracy improvement for seen and unseen circuits.

TABLE II
Circuit Statistics

Circuits	#Train	#Test	#cell	#net	category
ckt #1	40791	17483	10154	18892	seen
ckt #2	93786	40194	234391	340004	seen
ckt #3	16099	6900	37958	51175	seen
ckt #4	0	16998	6667	9072	unseen
ckt #5	0	23785	11830	15170	unseen
Total	150676	105360	301000	434312	

Fig. 6 gives a global view of the prediction result of the competitive models with the distribution of the mismatch between the predicted delay and the ground truth. The proposed model outperforms the other models with an extremely compact distribution which is centralized at nearly the zero point. The spread of the mismatch distribution of other models are much wider than the proposed model and obviously much more over-pessimistic or over-optimistic, which is consistent with rRMSE in Fig. 7.

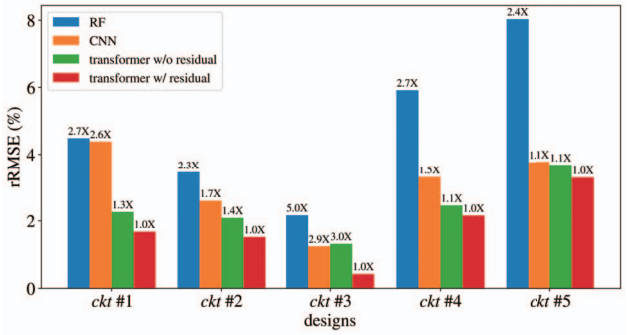

Fig. 7. Error distribution of different models on seen and unseen designs.

C. *Runtime Comparison*

In order to validate the efficiency of the proposed prediction framework, the runtime for the learning-based methods is compared with the common design flow as illustrated in Table III.

TABLE III
Runtime analysis

		Prediction Runtime (s)				
Model		*ckt* #1	*ckt* #2	*ckt* #3	*ckt* #4	*ckt* #5
Traditional IC flow	CTS	1378	31896	109	193	620
	Routing	1818	411968	143	254	816
	PrimeTime	655	1608	276	680	951
	Total	3851	445472	528	1127	2387
RF		11.2	26.5	10.6	15.7	28.4
CNN		1.28	2.68	0.57	1.22	1.66
This work		1.02 (3775X)	2.16 (206237X)	0.40 (1320X)	1.02 (1105X)	1.38 (1730X)

The required runtime in traditional design flow consists of the time used in clock tree synthesis, routing and timing analysis to perform physical design after placement and calculate path delay, while for learning-based models, the runtime consumes as the inference time of the trained model. The design tools are implemented on a 72-core 2.6GHz Linux machine with 1024 GB memory while the learning-based models are trained and inferenced on a Linux machine equipped with four NVIDIA GPUs Tesla V100 and 256GB memory. As shown in Table III, it can be seen that the

proposed pre-routing path delay prediction framework produces at least three orders of magnitude speedup compared with the traditional flow, which would even increase to six orders of magnitude for the large-scale *ckt* #2, indicating remarkable efficiency improvement for the design flow. The proposed prediction framework outperforms the random forest-based prediction model by at least one order of magnitude because the path-based prediction is much faster than the net-based approach by eliminating the iterative invoking and accumulation for each net along the path.

VI. Summary and Conclusions

An efficient and accurate pre-routing path delay prediction framework is proposed in this work by employing transformer network and residual model. Experimental results show that the proposed framework achieves significant prediction precision improvement compared with other learning approaches and speedup compared with traditional design flow.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2019YFB2205004), and in part by the National Natural Science Foundation of China under Grant(62174031) and in part by the Jiangsu Natural Science Foundation (Grant No. BK20201233).

References

[1]E. C. Barboza, N. Shukla, Y. Chen, 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.

[2]R. Liang, Z. Xie, J. Jung, V. Chauha, Y. Chen, J. Hu, ... G. J. Nam, “Routing-free crosstalk prediction,” *in 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1-9, IEEE, 2020.

[3]J. Qiu, S. Reda, S. Hassoun, “Fast, accurate a priori routing delay estimation,” *in Proceedings of the 12th ACM/IEEE international workshop on System level interconnect prediction (SLIP)*, pp. 77-82, IEEE, 2010.

[4]Q. Liu, M. Marek-Sadowska, “Pre-layout wire length and congestion estimation,” *in Proceedings of the 41st annual Design Automation Conference (DAC)*, pp. 582-587, IEEE, 2004.

[5]S. Shah, N. Mansouri, A. Nunez-Aldana, “Pre-layout estimation of interconnect lengths for digital integrated circuits,” *in 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06)*, pp. 38-38, IEEE, 2006.

[6]S. Bodapati, F. N. Najm, “Pre-layout estimation of individual wire lengths,” *in Proceedings of the 2000 international workshop on System-level interconnect prediction (SLIP)*, pp. 93-98, IEEE, 2000.

[7]D. Stroobandt, *A Priori Wire Length Estimates for Digital Design*, Springer Science & Business Media, 2001.

[8]A. B. Kahng, U. Mallappa, L. Saul, S. Tong, “"Unobserved corner" prediction: Reducing timing analysis effort for faster design convergence in advanced-node design,” *in 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 168-173, IEEE, 2019.

[9]A. B. Kahng, U. Mallappa, L. Saul, “Using machine learning to predict path-based slack from graph-based timing analysis,” *in 2018 IEEE 36th International Conference on Computer Design (ICCD)*, pp. 603-612, IEEE, 2018.

[10] A. B. Kahng, M. Luo, S. Nath. “SI for free: machine learning of interconnect coupling delay and transition effects,” *in 2015 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1-8, IEEE, 2015.

[11] H. H. Cheng, I. H. R. Jiang, 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.

[12] B. Shook, P. Bhansali, C. Kashyap, C. Amin, S. Joshi, “MLParest: Machine learning based parasitic estimation for custom circuit design,” *in 2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1-6, IEEE, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, I. Polosukhin, “Attention is all you need,” *in Advances in neural information processing systems*, pp. 5998-6008, 2017.

[14] Y. Zhuang, Y. Chen, J. Zheng, “Music Genre Classification with Transformer Classifier,” *in Proceedings of the 2020 4th International Conference on Digital Signal Processing*, pp. 155-159, 2020.

[15] Y. Han, L. Li, J. Zhang, “A Coordinated Representation Learning Enhanced Multimodal Machine Translation Approach with Multi-Attention,” *in Proceedings of the 2020 International Conference on Multimedia Retrieval*, pp. 571-577, 2020.

[16] P. H.Anh, N. X. Bach, T. M. Phuong, “Session-Based Recommendation with Self-Attention,” *in Proceedings of the Tenth International Symposium on Information and Communication Technology*, pp. 1-8, 2019.

[17] J. Luo, M. Ye, C. Xiao, F. Ma, “Hitanet: Hierarchical time-aware attention networks for risk prediction on electronic health records,” *in Proceedings of the 26th ACM/SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 647-656, 2020.

[18] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, V. Kumar, “Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles,” *in Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 558-566, 2019.

[19] F. Hamilton, A. L. Lloyd, K. B. Flores, “Hybrid modeling and prediction of dynamical systems,” *PLoS computational biology*, Vol. 13(7), pp. e1005655, 2017.

[20] T. Xu, A. J. Valocchi, “Data-driven methods to improve baseflow prediction of a regional groundwater model,” *Computers & Geosciences*, Vol. 85, pp. 124-136, 2015.

[21] J. P. A. Vieira, R. S. Moura, “An analysis of convolutional neural networks for sentence classification,” i*n 2017 XLIII Latin American Computer Conference (CLEI)*, pp. 1-5, IEEE, 2017.