Accurate timing prediction at placement stage with look-ahead RC network

Conference Paper · July 2022

DOI: 10.1145/3489517.3530598

5 authors, including:

Xu He
Hunan University
11 PUBLICATIONS **265** CITATIONS
SEE PROFILE

Yao Wang
Independent
27 PUBLICATIONS **239** CITATIONS
SEE PROFILE

Accurate Timing Prediction at Placement Stage with Look-Ahead RC Network

Xu He*, Zhiyong Fu*, Yao Wang†, Chang Liu†, Yang Guo†
*Hunan University, †National University of Defense Technology
Changsha, China

ABSTRACT

Timing closure is a critical but effort-taking task in VLSI designs. In placement stage, a fast and accurate net delay estimator is highly desirable to guide the timing optimization prior to routing, and thus reduce the timing pessimism and shorten the design turn-around time. To handle the timing uncertainty at the placement stage, we propose a fast net delay timing predictor based on machine learning, which extract the fully timing features using a look-ahead RC network. Experimental results show that the proposed timing predictor has achieved average correlation over 0.99 with the post-routing sign-off timing results obtained in Synopsys PrimeTime.

KEYWORDS

Machine Learning, Static Timing Analysis, Timing Estimation, Placement

ACM Reference Format:
Xu He*, Zhiyong Fu*, Yao Wang†, Chang Liu†, Yang Guo†. 2022. Accurate Timing Prediction at Placement Stage with Look-Ahead RC Network . In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC) (DAC '22), July 10–14, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530598

1 INTRODUCTION

Timing closure is a critical but also effort-taking task in VLSI designs, especially for those applications like high-performance processors, systems-on-chip(SoCs), and so on. Timing optimizations are performed throughout the entire design flow, and the earlier the intervention stage, the greater the room for optimization. However, accurate timing estimation at early-stages is difficult due to the lack of essential physical information and process variations. For instance, timing optimization in placement stage is normally falling into a pessimistic estimation on the net delay as routing has not been made yet. Traditionally, circuits are designed using a worst-case assumption, which adds an over-pessimistic timing margin to satisfy the performance specification under process variations. Industrial experience has shown that, the immense gap between the performance predicted by EDA tools and the final silicon performance can be up to 30% [4, 14]. In addition to the pessimistic timing margin, in placement stage, a large arbitrary clock uncertainty is inserted to cover up the net delay estimation "error" before routing. Consequently, the design might be optimized with a large mis-correlation with the net delay information after routing, which could lead to multiple design turn-around iterations and/or poor sign-off *Power-Performance-Area* (PPA) trade-offs.

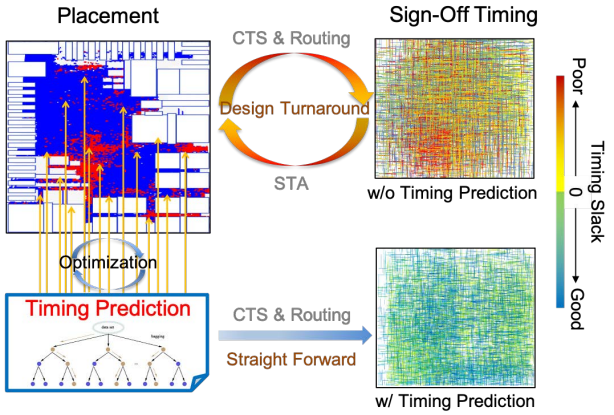

Figure 1: Applying machine learning-based timing model at the placement stage can improve timing prediction and deliver better design performance.

For this reason, an effective machine learning-based timing prediction methodology with Look-Ahead RC network (called "LaRC-Timer") is proposed in this paper, aiming to bridge the gap between the placement & routing steps. More specifically, as illustrated in Fig. 1, a timing model is trained by machine learning method with a look-ahead RC network for features related to both interconnections and gates, and then the endpoint slack and critical paths are calculated by traversing the netlist in topology order. In such a way, the high correlation of path delay with sign-off results at placement stage is achieved. Improved correlation can give "better accuracy for free" that boosts the cost-accuracy trade-offs and reduces the iterative turnaround time in the design flow, and thus stimulates the time-to-market of the product [10].

The main contributions of this paper can be summarized as follows:

- We have proposed a novel feature extraction method based on look-ahead RC network, which helps to increase the prediction accuracy effectively.
- We have presented a net based delay model, which can differentiate multiple input pins of a driver, and distinguish rising and falling delays. Therefore, timing pessimism is reduced significantly.

Xu He is the corresponding author, email: dawn.hx@gmail.com. This work is supported by the National Natural Science Foundation of China, under grant 61872136 and U19A2062.

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530598

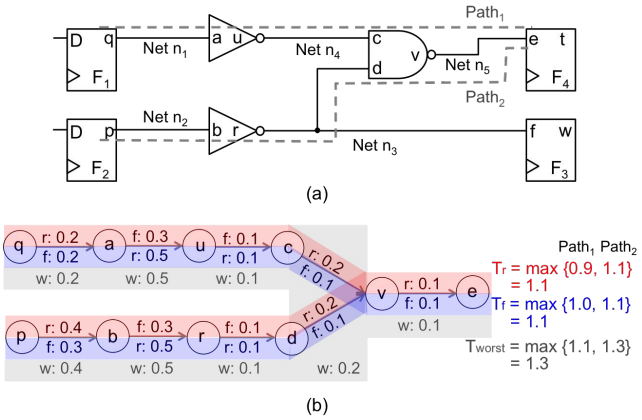

Figure 2: An example to show that, if the simplifications are applied by using the maximum delay among multiple driver input pins, and combining rising and falling delays by the worst, the path arrive time at pin e will be computed pessimistically.

- Experimental results have demonstrated that, under an industry advanced process technology (28nm), our machine learning-based delay model has achieved above 0.99 correlation with the sign-off timing on average.
- We have integrated the proposed timing model into OpenTimer [9] for the computation of critical paths, which demonstrates the compatibility of our model with the design-flow tools for timing optimizations.

The remainder of this paper is organized as follows. Section 2 gives previous works on machine learning techniques used in timing prediction. Section 3 gives the preliminary of timing analysis. Section 4 gives a framework overview of our method. In Section 5, we discuss the details of our timing prediction. Experimental results and conclusions are presented in Section 6 and Section 7, respectively.

2 RELATED WORKS

Currently, machine learning-based time analysis is mainly studied at the post-routing stage. Kahng *et al*. in [11] developed a customized learning technique to calibrate analytical interconnect delay models to fit the STA model, and thus reduce the number of invocations of the incremental static timing analysis (iSTA) tool. Han *et al*. in [8] proposed a machine learning-based golden timer extension (GTX) framework to calibrate the timing divergence between different tools & processes. To reduce the runtime and license costs on timing analysis, Kahng *et al*. in [12] developed a machine learning-based predictor of timing in SI mode based on timing reports from non-SI mode. In [3], Cheng *et al*. proposed a wire timing model trained by XGBoost for tree and non-tree net structures, with the net topological features considered.

However, all of the aforementioned models can only predict or handle timing analysis at post-routing stage. To improve the accuracy of timing prediction at pre-routing stage, Barboza *et al*. [2] made an attempt to use several machine learning engines to predict the net delay and slew rate from hand-crafted features. Their

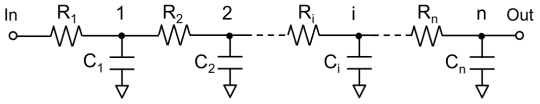

Figure 3: Elmore delay model.

pre-routing features include only the driver and sink capacitance, distance, max driver input slew, and sink locations, which lacks of the information from the net topology and RC network. As a result, it can attain good estimation results only on part of test designs. In [15], Shook *et al*. proposed a machine learning-based parasitic estimation to reduce the error between the pre-layout and the post-layout simulation for analog circuits. Neto *et al*. in [13] proposed a CNN-based binary classifier to identify critical-path at the logic synthesis stage, in which the net delay is not explicitly addressed in the model.

For timing analysis at the placement stage, prior prediction has two major problems: (1) **Insufficient topological feature selection**: The placement layout is not fully explored for potential RC information. Delay prediction at the placement stage is mainly relied on the distance between cells, but in fact, net delay is dominated by topology-dependent RC parasitics in deep-submicron processes. How to extract RC-related features effectively should be studied systematically at the placement stage. (2) **Pessimism-induced over-design**: To simplify the delay model, it does not differentiate among multiple driver input pins, and combines rising and falling delays by predicting the worst between them. Fig. 2 gives an example to show that the pessimism is worsen for the path arrive time at e. Both rising and falling arrive time T_r and T_f at e are 1.1. However, the arrive time at e is enlarged to 1.3 when the above simplifications is applied.

3 PRELIMINARY

3.1 Gate Delay Model

In most cell libraries, table models are included to specify delays and timing checks for various timing arcs of a cell. The table models are referred to as Non-Linear Delay Model (NLDM), and are used to capture the delay through the gate for various combinations of input transition time at the gate input pin and total output capacitance at the gate output.

As the transistor feature size continuously shrinks, one of the shortcomings of the NLDM is that, it is not accurate enough to reflect the timing below 65 nm technology. Therefore, in our timing prediction, the NLDM is only utilized to generate gate-related features, instead of being applied as the gate delay.

3.2 Wire Delay Model

In timing analysis, the Elmore delay metric [7] and the D2M (Delay with 2 Moments) delay metric [1] are applied for the first moment and the second moment of impulse response, respectively.

The Elmore delay metric, or the so-called first moment of the impulse response, is the most widely applied interconnect delay metric. Given a routing result, we can model each net as an RC tree. Fig. 3 gives an RC chain example. Elmore delay can be considered as finding the delay through each segment i, as the segment resistance R_i times the downstream capacitance, and then taking the sum of

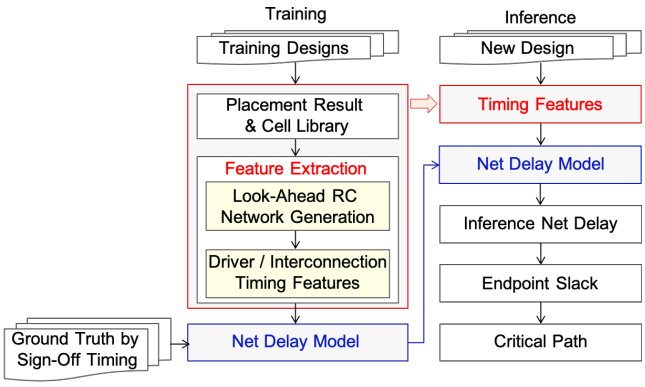

Figure 4: Framework of our method.

the delays from the root to the sink. Segment number equals to the edge number of a path, e.g., n segments from input pin to output pin. Elmore delay of this RC chain is given in Equation (1).

$$Elmore(Out) = \sum_{i=1}^{n} R_i \times \sum_{j=i}^{n} C_j. \tag{1}$$

For tree structure, the Elmore delay computation is similar, in which the downstream capacitance of node i is the sum of capacitance of all its branches.

Besides, the D2M delay metric is applied to alleviate errors of the near-end nodes (nodes relatively close to the driving source) in the Elmore delay. The D2M computation is given in Equation (2).

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

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

4 OVERVIEW OF OUR TIMING PREDICTION

The overview of our timing prediction framework is illustrated in Fig. 4.

In the model training process, given the placement input and technology libraries, we have to do the feature extraction for the net delay model training. At the placement stage, the routing information is still unavailable. To improve the accuracy of timing prediction, we apply a fast routing analysis on the inputted placement layout to generate a look-ahead RC tree for each net. With the help of the generated RC network, the topology-related characteristics like wire-length, as well as the parasitic parameters like the resistance and capacitance of nets, can be extracted and fed to train the net delay model. The training dataset for each net with timing label is extracted from a post-routing sign-off timing report using the PrimeTimer, with the crosstalk considered and the timing analysis performed in signal integrity mode. Moreover, the rising and falling delays for each net are treated separately with different features, which implies edge-related knowledge is considered in our model.

In the model inference process, given the inputted timing features of a net, the trained model is utilized for the prediction of the net delay. Based on the predicted net delay, the arrival time of each circuit node is computed by traversing the circuit graph in topology order. Based on the calculated arrival time, the endpoint slacks and the critical paths are reported and timing optimizations can be performed accordingly, with a good-correlation to the STA timing after routing. Moreover, the model require no knowledge on the design-specific features, it can be safely extended to other designs which utilize the same manufacturing process.

The key techniques of our method are explained in details in the following section.

Table 1: Feature Summary in Net Delay Model

Type	Feature	Reference
Driver -Related	Driving Strength	Netlist
	Fanout Number	
	Output Load	Look-Ahead RC Network
	NLDM Gate Slew	Cell Libaray, Look-Ahead RC Network
	NLDM Gate Delay	
Interconnection -Related	Distance	Placement Result
	Elmore Delay	Look-Ahead RC Network
	Context Elmore Delay	
	D2M Delay	

5 DETAILS OF THE KEY TECHNIQUES

The key techniques that we utilized to do the delay prediction at placement stage include the generation of the look-ahead RC network, the feature selection & extraction method, the machine learning algorithm, and the inference process.

5.1 The Generation of Look-ahead RC Network

At the placement stage, the routing information is still unavailable. To improve the accuracy of timing prediction, we apply a fast routing analysis to generate a look-ahead RC network for the input placement layout. In such a way, not only the wire-length, but also the parasitic resistance and capacitance of the nets can be obtained for the correlation improvement of net delay with the post-routing results.

To do fast routing analysis, we first partition each multiple-pin net into two-pin nets using Steiner tree algorithm [5]. Each two-pin net is then routed by L-shape routing. This routing analysis can be replaced by other simplified global routers as well for more sophisticated routing analysis.

5.2 The Feature Selection & Extraction Method

In machine learning models, feature selection is of critical importance to the effectiveness of model application. Table 1 lists the features for our net-based delay model. Each net-based sample is a connection between a driver input pin and a target sink, i.e., $b \rightarrow d$ in Fig. 5, the features can be divided into driver-related features and interconnection-related features.

The driver-related features include: (1) Driving strength: It is extracted by the cell type, and it determines the output impedance of the driver. Typically, the higher the driving strength, the smaller the net delay. (2) Output load: It is the load capacitance of driver. (3) Fanout number: The number of sink pins of the net. (4) NLDM gate slew: It represents the speed of rise and fall transitions, and thus influences timing. (5) NLDM gate delay: It is the delay of the timing arc within the driving gate. It should be mentioned that, both gate slew and gate delay are not the golden result from the sign-off

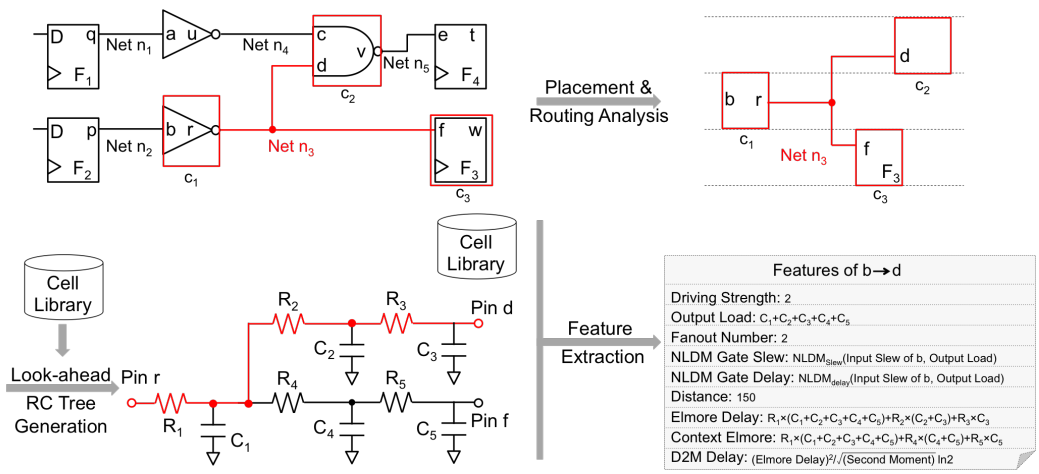

Figure 5: Feature extraction example. For a multi-pin net n_3, it can be partitioned into two-pin nets from input pin b of driver c_1 to sinks d and f, respectively. Each two-pin net-based delay incorporates both driver gate delay and wire delay to the target sink. Based on the placement layout, a fast routing analysis is applied to generate a look-ahead RC tree of n_3. Finally, the timing features are extracted according to the generated RC tree.

timer. Instead, they are obtained by the NLDM look-up table with two input indexes: the input slew and the output load of the driver, both of which are computed with the help of our look-ahead RC network.

The interconnection-related features include: (1) Distance: It is the Manhattan distance between the locations of the driving output pin and the target sink pin. (2) Elmore delay: We select the capacity and resistance from the RC tree, and compute the Elmore delay between the driving output pin and the target sink pin. (3) Context Elmore delay: Besides the target sink of the net, the other sinks are called context sinks, and the sum of Elmore delays between driving pin and context sinks is also applied. (4) D2M delay: The second moment of impulse response is extracted.

Fig. 5 gives an example of feature extraction. Given a net n_3, we do routing analysis by partitioning n_3 into two-pin nets $b \rightarrow d$ and $b \rightarrow f$ using Steiner tree algorithm. Each two-pin net is then routed by L-shape routing. A look-ahead RC tree can be generated from the routing analysis and the cell library. The features of each two-pin net are then extracted. The extracted features of the two-pin net $b \rightarrow d$ are given in Fig. 5 as an example.

It should be mentioned that, to consider rising and falling delays in the same predictor, for each two-pin net, different data samples are generated according to the combinations of rising and falling signals, i.e., at most $2 \times 2 = 4$ combinations. For each two-pin net, its rising and falling samples are varied by different related-features and timing labels.

5.3 Machine Learning Algorithm

The regression $y = f(\vec{x})$ is performed in our net-based model, where $\vec{x}$ is the vector of input features, and y represents the net delay at the target sink. After evaluating several machine learning methods like neural-networks, linear regression, etc., we choose the Random Forest (RF) algorithm [2, 16] for our timing prediction problem since it offered the best prediction accuracy based on our experiments.

5.4 Timing Inference

The circuit design in STA is modeled as a directed acyclic graph (DAG) $G = (P, E)$, where a vertex represents a pin in the design and an edge denotes a timing arc between two vertices. Typically, in our net-based model, each $p \in P$ is an input pin of a gate or a Flip-Flop (FF), and each $e \in E$ is a two-pin connection from a driver input pin to its target sink. Because of the special property of DAG, every pin $p \in P$ in the circuit graph can be levelized, such that the topological order among different pins are maintained.

In our method, the delay predictions of all nets can be obtained by calling the timing model inference only once, since we can extract features of all nets when the look-ahead RC network is constructed. After the prediction of the net delay, by traversing the graph G in topology order, the endpoint slacks can be computed, and the critical paths with negative slacks are identified. In our implementation, the inference results from our trained timing model are integrated into OpenTimer [9] for the computation of the critical paths.

6 EXPERIMENTAL RESULT

The proposed approach is implemented in C++17 and trained with Pytorch 1.8.0, and then evaluated on a PC machine with an Intel Core i7 (@3.00 GHz) and 16 GB DDR4. The experiment process and result analysis are illustrated and discussed in detail as follows.

6.1 Dataset Generation

The experiments are performed on the ITC'99 benchmark circuits [6] with a 28-nm industry process. Each design in the ITC'99 benchmark suite is synthesized using Synopsys Design Compiler at first. Physical design (P&R) is then performed by Cadence Innovus, and the output placement database serves as testcase for timing prediction. Its routing database is fed to Synopsys StarRC for parasitic extraction. Finally, Synopsys PrimeTime is called in signal integrity mode to generate the ground truth. The circuit information is summarized in Table 2.

Table 2: Benchmark Suite Information

Circuit	#Cells	#Nets	#FFs	#Training Data	#Testing Data
b11	420	436	33	53188	52880
b12	829	840	119	102084	101868
b13	216	228	48	21312	21320
b14	3889	3951	252	539230	539210
b20	8477	8553	517	1207702	1204146
b21	8343	8437	526	1200268	1201242
b17	15014	15252	1521	-	2228422
b22	12569	12743	757	-	1931542

The "#Cells", "#Nets", and "#FFs" are the average numbers of all implementations.

Table 3: Prediction Results of Net-Based Delay

Circuit	DAC'19 [2]				LaRC-Timer			
	Mean (ps)	Max (ps)	Cor	Test CPU (s)	Mean (ps)	Max (ps)	Cor	Test CPU (s)
b11	10	93	0.618	11	1	66	**0.989**	13
b12	9	85	0.748	10	1	23	**0.994**	12
b13	7	45	0.770	8	1	19	**0.994**	10
b14	9	86	0.891	15	2	58	**0.994**	19
b20	9	102	0.889	24	2	72	**0.995**	30
b21	9	105	0.891	24	2	70	**0.995**	30
Avg.	9	86	0.801	15	1	51	**0.993**	19
b17	13	121	0.832	36	3	100	**0.987**	47
b22	10	118	0.843	31	2	68	**0.993**	40
Avg.	11	119	0.837	34	2	84	**0.990**	44

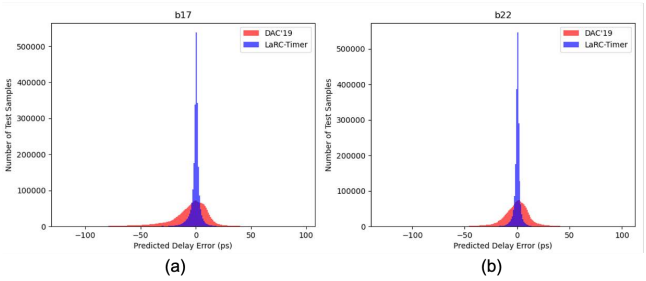

Figure 6: Comparison of net delay prediction on b17 (a) and b22 (b).

In order to obtain sufficient net samples for training, we variate several tool parameters such as the floorplan, the clock frequency, and so on in the logic and layout synthesis, and generate 32 different design implementations for circuits, in which 16 design implementations are used for training and the rest are used for testing. There is no intersection of circuit implementation between training and testing data. Please note that, for cross validation, there is no training data for circuit b17 and b22.

6.2 Comparison on Net-Based Delay Prediction

To evaluate the performance, we compare our results with the counterpart in recent work, namely the "DAC'19 [2]". It has to be mentioned that, the net-based model used in work [2] is different from ours. In work [2], it takes the maximum delay for each gate with multiple input pins, and it does not differentiate the rising and falling delays but uses the worst one of them. However, in our experiments, it is found that these simplifications may lead to a great prediction pessimism. Even though the ground truth is directly used, these simplifications will enlarge the path arrival time by 30.8% on average, and 65.4% for the maximum mismatch. Therefore, to be fair to their results, all the results for method "DAC'19 [2]" are produce by using their proposed features but our net-based delay model for comparison.

The comparison on the net-based delay is given in Table 3. The columns "Mean", "Max", and "Cor" are the mean error, maximum error, and correlation with respect to the sign-off timing from PrimeTime, respectively. The column "Test CPU" presents the inference run-time in seconds. In our implementation, the look-ahead RC network generation takes up about 10.7% of total inference run-time averagely, in which the Steiner tree construction accounts for about half of it. Normally, the larger a net is, the longer runtime it takes. The training time for "DAC'19" and "LaRC-Timer" are 1081 and 1097 seconds for the total 3.12M training data, respectively. As one can observe from Table 3, "Cor" of our method is above 0.99 on average, including cross validation results on b17 and b22. Moreover, the "Mean" and "Max" error predicted by our method are much smaller than "DAC'19".

Error distributions of the delay predictions are depicted in Fig. 6 for the cross validation results on b17 and b22. Results on other circuits and designs are similar. Based on the RF algorithm, compared with "DAC'19", our predicted delay errors are generally concentrated near ±0 with much smaller spread, which indicates the effectiveness of the feature selections of our model. Furthermore, it is found in our experiments that the features "NLDM Gate Slew" and "NLDM Gate Delay", which are generated from the Look-Ahead RC network, have the largest contribution to the delay among all the extracted features. Generally, the driver-related features are accounting for more than half of the total net delay. By contrast, the features in "DAC'19" are mainly related to interconnection features, which seems to be not fully taking the driver-related knowledge into consideration.

Table 4: Comparison on Endpoint Slack

Circuit	DAC'19 [2]			LaRC-Timer		
	Mean (ps)	Max (ps)	Cor	Mean (ps)	Max (ps)	Cor
b11	66	344	0.871	4	32	**0.995**
b12	38	129	0.899	4	23	**0.986**
b13	17	55	0.952	3	18	**0.999**
b14	110	516	0.862	13	47	**0.986**
b20	138	574	0.828	12	47	**0.971**
b21	140	473	0.840	15	56	**0.965**
Avg.	85	348	0.875	9	37	**0.984**
b17	188	416	0.823	15	87	**0.921**
b22	232	497	0.833	14	52	**0.963**
Avg.	210	456	0.828	15	70	**0.942**

6.3 Path Timing Analysis

6.3.1 Slack Comparison. The slack is defined as the difference between the required time and the arrival time at an endpoint, e.g., an FF (Flip-Flop) or a primary output. The slack value is used to identify the violated paths for further timing optimization. The overall predictions of the endpoint slacks are summarized in Table 4. Compared with "DAC'19", our slack errors are much less, and our correlation is higher than "DAC'19" by 11.0% on average.

6.3.2 Critical Path Classification. One important application of timing analysis or slack estimation is to identify those timing critical

Table 5: Critical Path Classification

Circuit	PrimeTime		DAC'19 [2]			LaRC-Timer w/o Correction			LaRC-Timer w/ Correction		
	#Critical	TNS (ns)	#Critical	TNS (ns)	TPR/TNR	#Critical	TNS (ns)	TPR/TNR	#Critical	TNS (ns)	TPR/TNR
b11	144/511	-2.640	268	-26.196	0.986/0.657	145	-2.985	0.958/0.981	161	-4.208	0.986/0.948
b12	313/1542	-5.155	617	-44.984	1.000/0.753	344	-6.207	1.000/0.975	403	-9.235	1.000/0.927
b13	36/762	-0.867	32	-1.330	0.778/0.994	36	-0.884	1.000/1.000	43	-1.197	1.000/0.990
b14	163/1537	-2.399	773	-126.920	1.000/0.556	408	-7.286	0.988/0.820	481	-10.885	1.000/0.769
b20	99/1548	-1.090	906	-158.024	1.000/0.443	202	-2.703	0.990/0.928	279	-4.611	1.000/0.876
b21	364/1500	-6.351	946	-173.084	1.000/0.488	560	-12.017	0.986/0.823	642	-16.870	0.997/0.754
Avg.	187/1233	-3.084	590	-88.423	0.961/0.648	283	-5.347	0.987/0.921	335	-7.834	0.997/0.877
b17	810/1532	-25.439	1364	-291.236	1.000/0.233	925	-37.399	0.951/0.785	1010	-45.153	0.973/0.693
b22	746/1476	-14.693	1332	-334.986	1.000/0.197	1015	-27.453	0.996/0.627	1106	-35.971	1.000/0.507
Avg.	778/1504	-20.066	1348	-313.111	1.000/0.215	970.000	-32.426	0.973/0.706	1058.000	-40.562	0.986/0.600

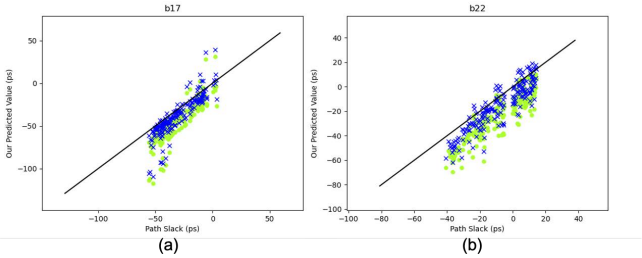

Figure 7: Analysis of our endpoint slack on b17 (a) and b22 (b). The blue "×" is our slack value, and the green "·" is the slack after correction.

paths with negative endpoint slacks. The results of the critical paths classification are summarized in Table 5. The column "TNS" denotes the total negative slack. The column "TPR"and "TNR" are the true positive and negative ratio, which represent the ratio of the number of truly critical and non-critical paths correctly identified by a classifier versus the total number of critical and non-critical paths reported from PrimeTime, respectively. A perfect TPR or TNR is 1.

In Table 5, one can observe that, based on our accurate prediction on net delay, our "TNS" is closer than "DAC'19" to the PrimeTime's results, and both of our "TPR" and "TNR" are generally equal or close to 1, which means our method can be served as an effective timing estimator to do timing optimization at the placement stage, with a good correlation with the post-routing results.

As mentioned in [2], it is important to perform a result correction for RF-algorithm based classifiers due to the oscillations of the classifier results near the threshold. To optimize the "TPR" and identify truly critical paths as many as possible, we make scaling-based correction on slack. After correction, our "TPR" of almost circuits is equal to 1, while our "TNR" is about 30% higher than that of "DAC'19". The distributions of our slack predictions w/o and w/ correction are plotted in Fig. 7 for b17 and b22, respectively. The solid lines indicate a perfect match of the prediction with the results of PrimeTime. Referring to the PrimeTime results, more than 52.8% of the endpoint slacks are within ±25ps on average. The larger the slack distribution concentrated around 0ps, the more sensitive the classification will be. Therefore, most of our data sets are "Hard-to-Classify" paths, and increasing slack pessimism by scaling-based correction will improve "TPR" at the expense of "TNR". An effective way to improve "TPR" is to reduce the under-estimation of the net delay, which will be an interesting direction for future work.

7 CONCLUSION

In this work, we introduce an efficient and accurate timing prediction method at the placement stage. To fully explore the potential routing information at the placement stage, a fast routing analyzer is integrated. This routing analyzer can not only provide predicted wire-length, but also be utilized to build a look-ahead RC network for nets. With the help of the look-ahead RC network, the features of interconnections and gates are extracted for net delay prediction. Experiments show that the proposed timing model can greatly improve the correlation of pre-routing timing prediction with the sign-off timing results.

REFERENCES

[1] C. J. Alpert, A. Devgan, and C. Kashyap. 2001. A Two Moments RC Delay Metric for Performance Optimization. In *International Conference on Computer-Aided Design*. 69–74.
[2] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. 2019. Machine Learning-Based Pre-Routing Timing Prediction with Reduced Pessimism. In *56th ACM/IEEE Design Automation Conference*. 1–6.
[3] Hsien-Han Cheng, Iris Hui-Ru Jiang, and Oscar Ou. 2020. Fast and Accurate Wire Timing Estimation on Tree and Non-Tree Net Structures. In *57th ACM/IEEE Design Automation Conference*. 1–6.
[4] D. G. Chinnery and K. Keutzer. 2000. Closing the Gap between ASIC and Custom: An ASIC Perspective. In *Proceedings of the 37th Annual Design Automation Conference*. 637–642.
[5] C. Chu. 2004. FLUTE: fast lookup table based wirelength estimation technique. In *IEEE/ACM International Conference on Computer Aided Design*. 696–701.
[6] S. Davidson. 1999. Characteristics of the ITC'99 Benchmark Circuits. In *IEEE International Test Synthesis Workshop*.
[7] William C Elmore. 1948. The transient response of damped linear networks with particular regard to wideband amplifiers. In *Journal of Applied Physics*. 55–63.
[8] Seung-Soo Han, Andrew B. Kahng, and et al. 2014. A Deep Learning Methodology to Proliferate Golden Signoff Timing. In *Proceedings of the Conference on Design, Automation and Test in Europe*. Article 260, 6 pages.
[9] Tsung-Wei Huang and Martin D. F. Wong. 2015. OpenTimer: A high-performance timing analysis tool. In *IEEE/ACM International Conference on Computer-Aided Design*. 895–902.
[10] Andrew Kahng. 2018. Machine Learning Applications in Physical Design: Recent Results and Directions. In *Proceedings of the International Symposium on Physical Design*. 68–73.
[11] Andrew B. Kahng and et al. 2013. Learning-based approximation of interconnect delay and slew in signoff timing tools. In *ACM/IEEE International Workshop on System Level Interconnect Prediction*. 1–8.
[12] Andrew B. Kahng, Mulong Luo, and Siddhartha Nath. 2015. SI for free: machine learning of interconnect coupling delay and transition effects. In *2015 ACM/IEEE International Workshop on System Level Interconnect Prediction*. 1–8.
[13] Walter Lau Neto and et al. 2021. Read Your Circuit: Leveraging Word Embedding to Guide Logic Optimization. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 530–535.
[14] Michael Orshansky and Kurt Keutzer. 2002. A General Probabilistic Framework for Worst Case Timing Analysis. In *Proceedings of the 39th Annual Design Automation Conference*. 556–561.
[15] B. Shook, P. Bhansali, and et al. 2020. MLParest: Machine Learning based Parasitic Estimation for Custom Circuit Design. In *57th ACM/IEEE Design Automation Conference*. 1–6.
[16] M. Magdon-Ismail Y. S. Abu-Mostafa and H.-T. Lin. 2012. *Learning from Data*. Vol. 4. AMLBook New York, NY, USA.