A Deep-learning-based Statistical Timing Prediction Method for Sub-16nm Technologies

Jiajie Xu, Leilei Jin, Wenjie Fu, Longxing Shi
The National ASIC System Engineering Technology Research Center, Southeast University
Nanjing, China
{jiajiex, jinleilei, wenjfu, lxshi}@seu.edu.cn

***Abstract*—Pre-routing timing estimation is vital but challenging since accurate net information is available only after routing and parasitic extraction. Existing methodologies predict the timing metrics with the help of the placement information of standard cells. However, neglecting the analysis of process variation effects hinders the precision of those methodologies, especially in sub-16nm technologies as delay distributions become asymmetric. Therefore, a deep-learning-based statistical timing prediction method is proposed to model process variation effects in the pre-routing stage. Congestion features and pin-to-pin features are fed into graph neural networks for post-routing interconnect parasitic and arc delay prediction. Moreover, a calibration method is proposed to compensate for the precision loss of the delay propagation. We evaluate our methods using open-source designs and EDA tools, which demonstrate improved accuracy in pre-routing timing prediction methods and a remarkable speed-up compared to traditional routing and timing analysis process.**

***Index Terms*—pre-routing timing estimation, process variation, deep learning, statistical delay propagation**

I. Introduction

During manufacturing of Very Large-Scale Integration designs, timing closure is a vital but effort-taking task. With the increment of chip integration level and the rising density of circuit devices, interconnection increasingly becomes a bottleneck affecting chip performance. Nevertheless, pre-routing timing estimation is normally pessimistic since accurate parasitic resistance and capacitance are available only after routing and parasitic extraction. Consequently, a noticeable divergence emerges in timing reports before and after routing. To bridge the gap, learning-based methods are employed for net delay prediction [1]–[5]. These approaches utilize deep learning methods, such as graph neural network and Transformer models, to predict post-routing delay. However, with technology scaling progressing well into the sub-16nm regime, process variations have a growing impact on circuit delay evaluation [6]. Especially in near-threshold voltage, cell delay distribution becomes asymmetrical due to complicated process variation effects. Unfortunately, existing works do not comprehensively account for the impact of process variations during the pre-routing timing estimation.

Statistical static timing analysis emerges as a solution to effectively model the process variation influences. Existing prediction models can be integrated with Liberty Variation Format (LVF) to account for process variation effects. In LVF, statistical features (sigma and skewness) are stored in look-up tables to model the asymmetric delay distribution. The transition time and the load capacitance are taken as the index of tables to calculate the cell delay. In pre-routing stage, the load capacitance is approximated based on the number of fanout cells. Existing prediction methods predict the effective capacitance to represent the parasitic influence and add it to the original load capacitance. However, the effects of interconnects on cells cannot be simply captured by the effective capacitance [7] as statistical cell and net delays exhibit a strong correlation [8]. Moreover, the path delay is calculated as the accumulation of individual arc delays. Yet in sub-16nm technologies, the propagation of delays from individual cells and nets to overall path is complicated because of non-Gaussian characteristics of statistical delay distribution. This effect hinders the precision of delay calculation in advanced technology nodes.

In this paper, we propose a novel deep-learning-based statistical timing prediction method for sub-16nm technology nodes. As depicted in Fig. 1, our approach begins with extracting placement information from training designs to determine congestion and pin-to-pin connectivity features. These features are then utilized for training the prediction model of parasitic parameters, as shown in blue region of Fig. 1. Afterwards, cell features are extracted and integrated with the parasitic features to form the timing arc features, which is then fed into the timing prediction model (green region). Finally, the well-trained model is implemented in test designs for arc delay prediction, followed by a delay propagation method (red dashed box) to bridge the gap between statistical arc delay and path delay. The key contributions of this work can be

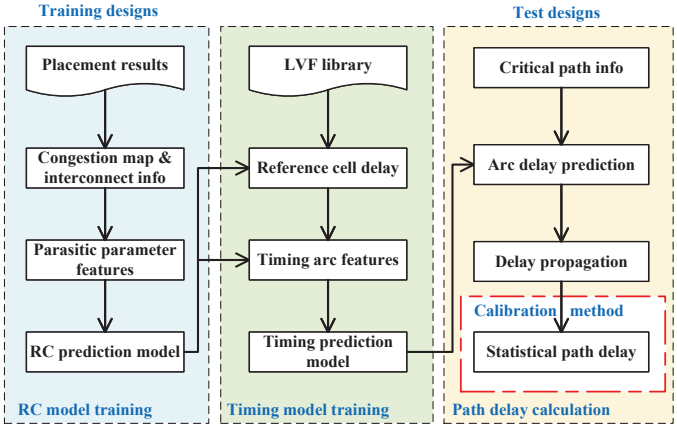

Fig. 1: Flowchart of our framework.

979-8-3503-4859-0/DATE24/© 2024 EDAA

summarized as follows:

1) We introduce a novel graph learning technique that effectively integrates both physical proximity information and pin-to-pin connectivity information of a target net.

2) We treat interconnect and its driver cell as a unified entity to predict timing arc delay, mitigating inaccuracies caused by their correlation.

3) We propose a calibration method to compensate for the discrepancy introduced by the propagation of non-Gaussian statistical delay of timing arcs to overall path.

The rest of this paper is structured as follows. Section II provides a brief overview of prior works and their limitations. Section III details the statistical timing prediction approach. The experimental results are presented in Section IV and Section V concludes this paper.

II. Preliminaries

A. Pre-routing Timing Prediction

Pre-routing timing prediction is essential because the routing process is time-consuming. Traditional methods estimate the interconnect delay based on the wire length. Their accuracy is insufficient due to the lack of routing details. Machine learning offers improved predictions. Random forest model is utilized by Barboza et al. [1] and He et. al [2] to predict delay of each net. Yang et. al [4] treat path delays as sequences and fed them into transformer network for path level timing metrics, without analysis of delay propagation. A timing engine inspired graph neural network is proposed by Guo et al. [3] to predict total negative slacks and worst negative slacks for timing optimization at routing stage, losing accuracy in predicting detailed path delay. Chang et al. [5] employ convolutional neural networks to extract placement features and subsequently use artificial neural networks for net delay prediction. Interconnect features and cell features are considered separately, compromising the prediction accuracy even with complicated physical characterization extraction. Furthermore, circuit delay evaluation is increasingly affected by process variations as technology scales into sub-16nm regime. The cell delay distribution becomes asymmetrical at near-threshold voltage. Currently, process variations are not adequately accounted for during pre-routing timing estimation [3]. As illustrated in Fig. 2 (a), traditional corner-based timing methodologies (SS/TT/FF) yield errors of up to 65% in 0.4V when compared to the golden results of Monte Carlo (MC) simulation. Such discrepancies can result in excessive number of iterative optimizations which is unacceptable to circuit designers. Therefore, a prediction approach that accounts for process variation effects is needed for delay estimation prior to routing.

B. Statistical Delay Propagation

In sub-16nm manufacturing technologies, cell delay distributions exhibit non-Gaussian features like skewness, especially at near-threshold or sub-threshold voltages [9]. These features lead to increased inaccuracy and make statistical delay of cells and nets hard to model. Researchers often turn to

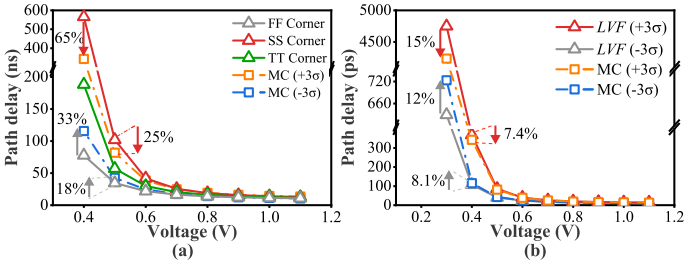

Fig. 2: (a) Corner-based path delay results comparing to MC simulation under different voltages. (b) LVF-based path delay results comparing to MC simulation under different voltages.

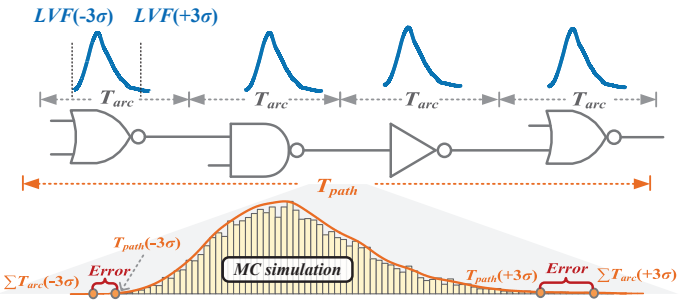

Fig. 3: The path delay propagation.

complicated expressions to capture cell-level asymmetric delay distributions [10]–[12]. LVF emerges as a widely accepted industrial approach [13], modeling the cell delay asymmetry as extra sigma values (σ_{early} and σ_{late}). However, as shown in Fig. 2 (b), a increasing error existed between the path delay calculated from LVF based LVF(+/-3σ) delay and golden delay MC(+/-3σ) from MC simulations as voltage decreases. This additional error is introduced by the propagation of individual statistical delays of timing arcs to overall path, as depicted in Fig. 3. It remains unsolved especially at low voltages as shown in Fig. 2 (b). For example, in 0.3V, the discrepancy between LVF and the MC highly up to 15% and 12% for +3σ and -3σ path delay, separately.

III. Statistical Timing Prediction

Fig. 4 depicts our proposed statistical timing prediction model, comprising two stages. In the first stage, a graph learning-based model is utilized to predict the net resistance (R) and capacitance (C). These predictions are fed into the second stage, where a timing arc delay prediction model is developed. Integrating the predicted R and C values in the second stage enhances accuracy and reliability in timing prediction, yielding more accurate timing estimation results.

A. RC Network Features

Data representation: Two distinct graph data structures are used to represent physical proximity information and pin-to-pin connection information for a target net. For physical proximity information, a directed graph is employed, denoted as $\mathcal{G}_1$. It represents the target net and its physically adjacent nets as nodes, all connected to the target net. The node feature

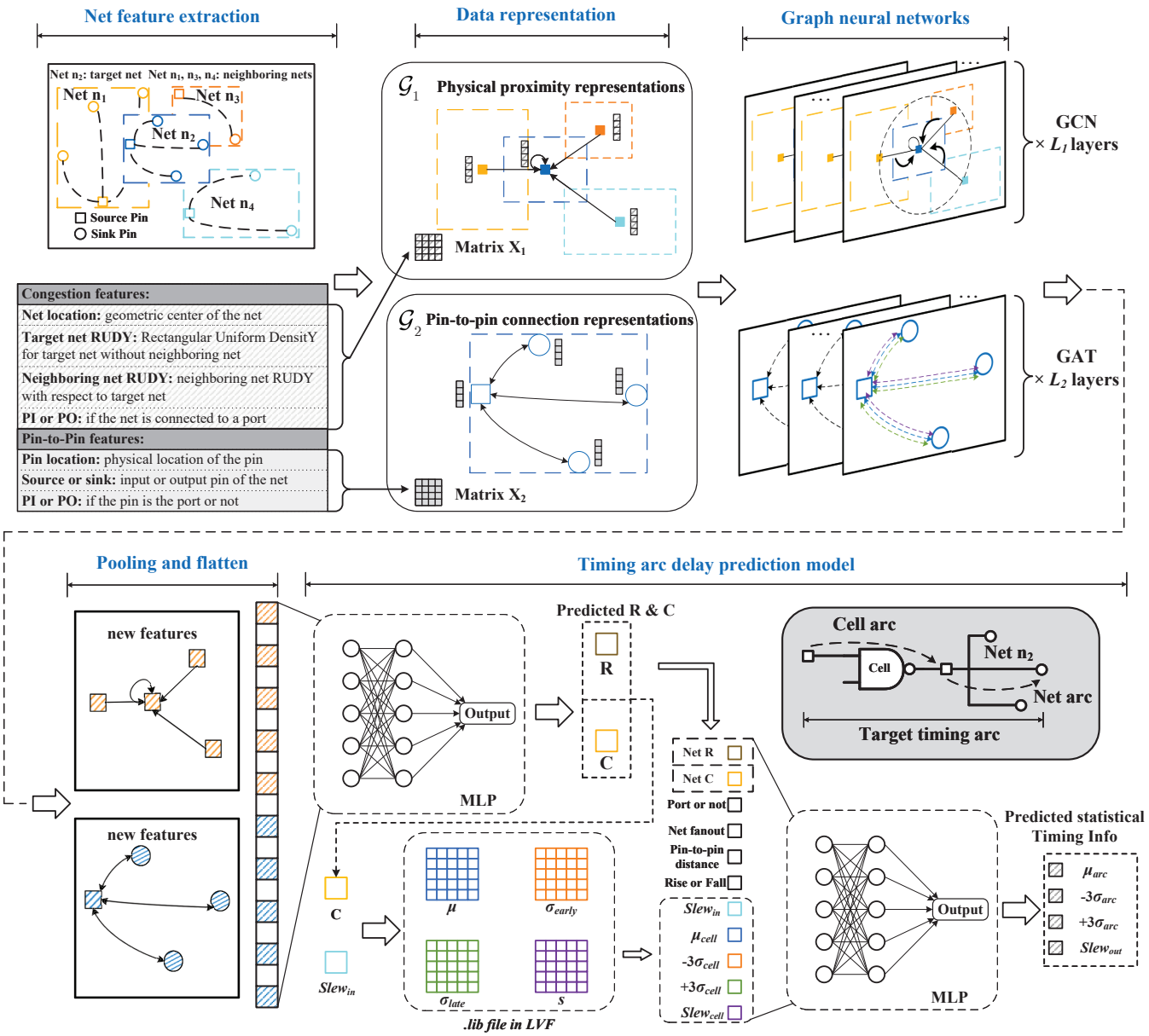

Fig. 4: Statistical timing prediction model.

matrix $\mathbf{X}_1$ in $\mathcal{G}_1$ captures characteristics of both the target net and its neighboring nets, stored in the graph dataset $\mathbf{X}$. Regarding the pin-to-pin connection information, a bidirectional graph denoted as $\mathcal{G}_2$ is used. The nodes in this network correspond to the source or sink pins of the target net. The features of these pins are also represented as a node feature matrix $\mathbf{X}_2$ and stored in the graph dataset $\mathbf{X}$.

Net feature extraction: There are two types of features: congestion features and pin-to-pin features. On the one hand, the congestion features correspond to the characteristics that describe the relationship between the target net and its neighboring nets. These features are utilized by graph $\mathcal{G}_1$. On the other hand, the pin-to-pin features represent the attributes of pins in target nets, and they are employed in graph $\mathcal{G}_2$. Detailed explanations of these features are provided in the grey table of Fig. 4. The target net Rectangular Uniform DensitY [14] ($\mathrm{RUDY}(i)$) for target net i and neighboring net RUDY ($\mathrm{RUDY}(i,j)$) for neighboring net j are calculated by:

$$\mathrm{RUDY}(i) = \frac{\mathrm{HPWL}(i)}{\mathrm{Area}(i)}, \tag{1}$$

$$\mathrm{RUDY}(i,j) = \frac{\mathrm{HPWL}(j)}{\mathrm{Area}(j)} * \frac{\mathrm{overlapArea}(i,j)}{\mathrm{Area}(i)}, \tag{2}$$

where $\mathrm{HPWL}(i/j)$, $\mathrm{Area}(i/j)$ and $\mathrm{overlapArea}(i,j)$ stand for half perimeter net length of net i/j, bounding box area of net i/j and the overlap area of the bounding boxes for net i and net j.

B. Net RC Prediction

Graph neural networks: Our RC prediction model comprises two components: a graph convolutional network (GCN) model for $\mathcal{G}_1$ and a graph attention network (GAT) model for $\mathcal{G}_2$. Their key distinction lies in the method they employ to aggregate information from the one-hop neighborhood. In the case of GCN, a graph convolution operation is performed, which computes the normalized sum of the node features from neighboring nodes. Since $\mathcal{G}_1$ is specifically designed to capture proximity information related to the target network, we leverage GCN to aggregate congestion features within $\mathcal{G}_1$. Furthermore, the self-loop feature of $\mathcal{G}_1$ aids GCN in aggregating the self-feature of the target net. As an alternative to statically normalized convolution, GAT introduces an attention mechanism through the calculation of the coefficient in (5).

The node representations of our GCN model are calculated by:

$$\mathcal{H}i^{(l+1)} = \sigma\left(\sum_{j\in N_i} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}} W^{(l)}\mathcal{H}_j^{(l)}\right). \quad (3)$$

Meanwhile, the node representations of our GAT model are calculated using:

$$\mathcal{H}_i^{(l+1)} = \sigma\left(\sum_{j\in N_i} \alpha_{ij}^{(l)} W^{(l)}\mathcal{H}_j^{(l)}\right). \quad (4)$$

Here, $\mathcal{H}_i^{(l)}$, σ, N_i and $W^{(l)}$ indicate the representation vector for node i in the l-th layer, the activation function (e.g., ReLU), the set of neighboring nodes of node i and the weight matrix of the l-th layer, respectively. $\deg(i)$ represents the degree of the node i, while $\alpha_{ij}^{(l)}$ denotes the attention coefficient between node i and j in l-th layer, calculated as follows:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l)T}[W^{(l)}\mathbf{h}_i^{(l)}||W^{(l)}\mathbf{h}_j^{(l)}]\right)\right)}{\sum_{k\in N_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l)T}[W^{(l)}\mathbf{h}_i^{(l)}||W^{(l)}\mathbf{h}_k^{(l)}]\right)\right)}, \quad (5)$$

where LeakyReLU refers to the Leaky Rectified Linear Unit, $\mathbf{a}^{(l)}$ represents the weight vector which is used for the attention mechanism and $||$ is the concatenation operation.

Pooling and flatten: After L_1 layers of GCN and L_2 layers of GAT, our graph-based prediction model is followed by a pooling and flatten module, which uses a global mean pooling layer for all the node representations. The overall net representation $\mathcal{F}_n$ is calculated as follows:

$$\mathcal{F}_n = (\frac{1}{N_1}\sum_{q\in\mathcal{G}_1} X_q^{L_1})||(\frac{1}{N_2}\sum_{p\in\mathcal{G}_2} X_p^{L_2}), \quad (6)$$

where N_1 and N_2 represent number of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$, separately. Afterward, a multiple layer perceptron (MLP) is employed to predict the target net resistance and capacitance.

The prediction models for resistance and capacitance are trained separately. In each model, both types of graph neural networks are trained together, and the output RC values serve as input features for delay prediction.

TABLE I: Cell and net features used for timing prediction.

Type	Name	Description
Net	Net R	Predicted net resistance
	Net C	Predicted net capacitance
	Port or not	If the net is the output port
	Net fanout	The fanout number of the net
	Pin-to-pin distance	The HPWL distances between the pins in the timing arc
Cell	Rise or fall	Rise or fall trend of the signal
	$Slew_{in}$	Input slew of the cell
	μ_{cell}	Mean value of the cell delay
	-3σ_{cell}	-3σ point of the cell delay
	+3σ_{cell}	+3σ point of the cell delay
	$Slew_{cell}$	Output slew of the cell

C. Timing Arc Delay Prediction

Timing arc delay prediction model: Our timing prediction model considers the target net and its driver cell as a unified entity, accounting for both cell and net characteristics (grey box of Fig. 4). The first step involves gathering statistical cell delay information using predicted capacitance and input slew values from LVF, which can be expressed as:

$$[\mu_{cell}, \sigma_{early}, \sigma_{late}, Slew_{cell}] = LUT(C_{pre}, Slew_{in}). \quad (7)$$

The +/-3σ point of cell delay (+/-3σ_{cell}) is then calculated through the looked up values. Next, we combine cell and net features, including predicted net RC, as inputs for another MLP to obtain timing information estimation, such as the delay mean, +/-3σ point delay and the output slew of the timing arc. Table I provides detailed explanations of all the input features used in this prediction model. Since our prediction model only contains the target net after the driver cell, the effect of the RC network for the input port requires additional calibration. To address this, an extra MLP that utilizes specific net features is employed to predict the RC network effect for the input port.

D. Statistical Delay Propagation Method

As illustrated in the second section, precisely characterized arc delay models are insufficient in accurately formulating statistical path delays. As a result, a path delay calibration method is proposed as follows.

To begin with, based on the easily obtained μ, σ_{early} and σ_{late} in LVF, a coefficient denoted as λ is introduced to measure arc delay asymmetry:

$$\lambda = \frac{\sigma_{late} - \sigma_{early}}{\mu}, \quad (8)$$

λ characterizes the extent to which the arc delay distribution diverges from a Gaussian distribution. A positive λ suggests a long right-hand tail in the distribution, and vice versa.

Following that, to represent the overall path delay deviation, an arithmetic mean value of λ is calculated for all arcs in the path. Mathematically, this can be expressed as follows:

$$\overline{\lambda} = \frac{1}{d_{path}} * \sum_{i=1}^{d_{path}} \lambda_i, \quad (9)$$

where d_{path} represents the depth of the path.

Lastly, employing a polynomial regression enables a calibration to reduce the difference between the sum of individual arc delay sigma points $\Sigma T_{arc}(\pm 3\sigma)$ and the actual path delay sigma point $T_{path}(\pm 3\sigma)$:

$$T_{path}(\pm 3\sigma) = \Sigma T_{arc}(\pm 3\sigma) * \sum_{i=0}^{n} \xi_n \overline{\lambda}^n. \quad (10)$$

The regression coefficients ξ_n can be easily acquired by fitting experimental data of circuit path delays and stored into library files. Commercial timing analysis tools can thus be flexibly integrated with this method.

IV. Experimental Results

The proposed approach is implemented in Python 3.10 and trained with Pytorch 2.0.1 using one NVIDIA GeForce RTX 4090 GPU, and then evaluated on a Linux machine with an Intel Core i9 (@4.7GHz) and 64GB DDR5 memory.

A. Dataset Generation

The experiments are conducted on the ISCAS85 benchmark circuits and OpenCores [15] designs with commercial 14nm technology. The benchmark information are summarized in Table II. Firstly, each design is synthesized using Design Compiler. Subsequently, physical design is performed using IC Compiler II, with the output results used for timing prediction. Moreover, Siliconsmart is used to generate the LVF library, in a supply voltage of 0.4V, representing the worst-case sub-threshold region, and a temperature of 25°C. The output slew and delay of every timing arc is extracted from post-routing PrimeTime timing report. In order to validate the path delay model, the 10% critical paths in each circuit is individually simulated. 10k MC simulations are performed to construct the timing database as golden output.

TABLE II: Benchmark suite information.

	Circuits	#Cells	#Nets	RC Data	Timing Data
train	c432	526	519	1.4 MB	356 KB
	c1355	955	923	2.6 MB	720 KB
	c1908	1117	1092	3.1 MB	844 KB
	c2670	1592	1452	8.0 MB	1000 KB
	c3540	2263	2241	7.0 MB	1712 KB
	c6288	5276	5244	14.0 MB	4344 KB
	c7552	4165	4057	17.0 MB	3124 KB
	aes	15696	14487	61.8 MB	9184 KB
	nova	329046	318295	1274.0 MB	68628 KB
	total	360636	348310	1388.9 MB	89912 KB
test	c499	966	934	2.7 MB	700 KB
	c880	1014	988	2.9 MB	700 KB
	c5315	3097	2974	13.0 MB	2248 KB
	des	4480	4232	19.2 MB	2560 KB
	wb_con	67746	52795	146.3 MB	17072 KB
	vga	119183	118273	421.6 MB	36076 KB
	total	196486	180130	605.7 MB	59356 KB

B. Parameter Selection

For the parameters of our RC prediction model (L_1: layer of GCN, L_2: layer of GAT), three distinct configurations are used for evaluation: PlanA (L_1=10, L_2=3), PlanB (L_1=5, L_2=5), and PlanC (L_1=3, L_2=10). The corresponding results for each configuration are presented in Table III. Better generalization is demonstrated with PlanB, so it is chosen as the default configuration for RC prediction.

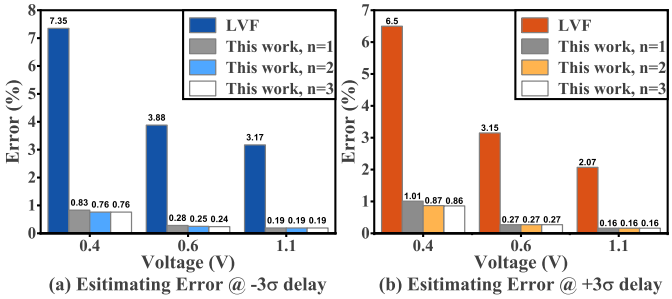

Fig. 5: Errors of estimating +/-3σ path delay.

TABLE III: Parasitic prediction accuracy.

	R/C prediction accuracy (R^2 score)		
	PlanA	PlanB	PlanC
train	0.9927/0.9989	**0.9988/0.9995**	0.9861/0.9968
test	0.9927/0.9527	**0.9988/0.9545**	0.9860/0.9532

In the path delay calibration model, we compare the errors of polynomial regression with different orders and error of path delay calculated from LVF. The results in Fig. 5 show an average error less than 1%, indicating a strong correlation between path delay deterioration and the proposed $\overline{\lambda}$. Polynomials of order 2 effectively bridge the gap between calculated delay and actual path delay, with an average error of 0.42%. Order 3 shows only marginal improvement compared to order 2, so we choose n=2 as the default order.

C. Accuracy of Timing Arc Delay Prediction

The proposed prediction method is compared with two recent works, "DAC19" [1] and "DAC22" [2]. However, it's important to note that our prediction target differs from the net-based models used in both "DAC19" and "DAC22", as they only focus on net delay prediction. During the evaluation

TABLE IV: Prediction results of +/-3σ point of timing arcs.

Circuit	Cell and net timing prediction (R^2 score)					
	DAC19		DAC22		Ours	
	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ
c432	0.9960	0.9924	**0.9992**	**0.9982**	0.9962	0.9946
c1355	0.9970	0.9939	**0.9994**	**0.9987**	0.9962	0.9953
c1908	0.9941	0.9916	**0.9989**	**0.9984**	0.9944	0.9923
c2670	0.9934	0.9913	**0.9986**	**0.9979**	0.9911	0.9947
c3540	0.9953	0.9906	**0.9992**	**0.9980**	0.9940	0.9931
c6288	0.9956	0.9917	**0.9993**	**0.9985**	0.9952	0.9929
c7552	0.9965	0.9929	**0.9994**	**0.9985**	0.9960	0.9945
aes	0.9967	0.9974	**0.9997**	0.9979	0.9976	**0.9981**
nova	0.9935	0.9953	**0.9986**	**0.9992**	0.9974	0.9978
Avg.	0.9953	0.9930	**0.9991**	**0.9983**	0.9953	0.9948
c499	0.9481	0.8470	0.9822	0.9702	**0.9890**	**0.9775**
c880	0.9383	0.8405	0.9804	0.9687	**0.9887**	**0.9787**
c5315	0.9334	0.7962	0.9789	0.9617	**0.9849**	**0.9672**
des	0.9756	0.8965	0.9782	0.9704	**0.9796**	**0.9832**
wb_con	0.8715	0.9198	0.9720	0.9392	**0.9822**	**0.9926**
vga	0.8935	0.8952	0.9785	0.9793	**0.9875**	**0.9879**
Avg.	0.9273	0.8659	0.9784	0.9649	**0.9853**	**0.9812**

TABLE V: The path delay prediction results based on ISCAS85 benchmarks and OpenCores circuits.

Circuit	Critical path delay (ps)								Errors of path delay (%)						Runtime (s)				
	MC		DAC22+LVF		DATE23+LVF		Ours		DAC22+LVF		DATE23+LVF		Ours		Routing	STA	Total	Ours	Speed-up
	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ	-3σ	+3σ					
c432	792	2422	683	2731	732	2540	790	2428	13.8	12.8	7.5	4.9	**0.2**	**0.3**	12	1.12	13.12	**0.525**	**25**×
c1355	777	2416	665	2575	727	2480	781	2401	14.4	6.6	6.4	2.7	**0.5**	**0.6**	17	1.17	18.17	**0.532**	**34**×
c1908	977	3136	871	3587	918	3296	986	3200	10.7	14.4	6.0	5.1	**0.9**	**2.0**	18	1.15	19.15	**0.513**	**37**×
c2670	981	2783	856	3346	903	2928	976	2782	12.7	20.2	8.0	5.2	**0.5**	**0.1**	22	1.18	23.18	**0.531**	**44**×
c3540	1131	3695	972	3918	1047	3926	1132	3524	14.0	6.1	7.5	6.3	**0.1**	**5.0**	35	1.19	36.19	**0.515**	**70**×
c6288	2230	7264	1921	7596	1918	7886	2286	7042	13.9	4.6	14.0	8.5	**2.5**	**3.0**	94	1.47	95.47	**0.531**	**180**×
c7552	857	2617	754	2839	794	2671	852	2590	11.9	8.5	7.4	2.1	**0.5**	**1.0**	63	1.34	64.34	**0.498**	**129**×
aes	2568	8107	2217	9666	2421	9028	2580	8014	13.6	19.2	5.7	11.4	**0.4**	**1.1**	596	1.99	597.99	**0.493**	**1176**×
nova	4104	9486	3696	11878	3780	11305	3838	9174	9.9	25.2	7.9	19.2	**6.5**	**3.3**	8333	18.87	8351.87	**1.364**	**6123**×
Avg.	-	-	-	-	-	-	-	-	12.8	13.1	7.8	7.3	**1.3**	**1.8**	1021	3.28	1024.28	**0.611**	**869**×
c499	858	2597	718	2916	772	2679	835	2544	9.9	12.3	9.9	6.9	**2.6**	**2.0**	15	1.14	16.14	**0.510**	**32**×
c880	724	2220	606	2520	636	2403	688	2085	16.3	13.5	12.1	8.2	**4.9**	**6.1**	17	1.14	18.14	**0.520**	**35**×
c5315	926	3028	780	3266	830	3458	895	2822	15.7	7.8	10.3	14.2	**3.2**	**6.8**	46	1.27	47.27	**0.509**	**93**×
des	2283	6568	1947	7190	2016	7160	2116	6361	14.7	9.5	11.7	9.0	**7.3**	**3.2**	416	1.48	417.48	**0.492**	**849**×
wb_con	1993	5407	1759	6120	1688	6028	1875	5228	11.7	13.2	15.3	10.3	**5.9**	**3.3**	1695	4.58	1699.58	**0.681**	**2496**×
vga	1024	2469	872	2632	934	2746	955	2622	14.8	6.6	8.8	11.2	**6.7**	**6.2**	3129	7.10	3136.10	**0.782**	**4010**×
Avg.	-	-	-	-	-	-	-	-	11.8	9.7	13.5	10.8	**5.1**	**4.6**	886	2.79	889.12	**0.582**	**1253**×

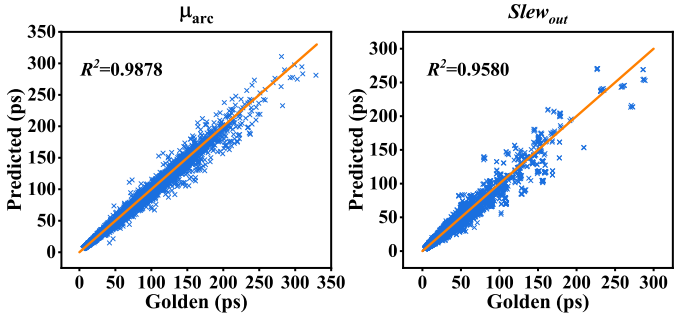

Fig. 6: Delay mean and output slew prediction.

process, their proposed net-based features are utilized along with cell delay features derived from their predicted RC values.

The results in Table IV show that the random forest model used in these works fits the training data well, and the lookahead RC network in "DAC22" improves prediction accuracy as expected. However, without accurate information of parasitic parameters, these models show low accuracy in inference. In contrast, our approach exhibits better stability in predicting both +3σ and -3σ points with the predicted RC information. Fig. 6 presents our results of the predicted delay mean and output slew of the timing arcs in the test benchmarks.

D. Accuracy and Runtime of Path Delay Prediction

Our statistical prediction method is compared to "DAC22" [2] and "DATE23" [5] for the path delay estimation. The net delay prediction models are integrated with accurate cell delay LVF models. The results are presented in Table V. Although previous net delay estimation techniques demonstrate adequate accuracy, the path delay is inaccurate due to unsolved propagation issues. In contrast, our methodology demonstrates superior accuracy in our benchmark circuits. On average, our prediction model achieves an error of 1.6% for the training set and 4.8% for the test set. Furthermore, our method demonstrates a runtime speed-up of average 1000 times compared to the traditional routing and static timing analysis process.

V. Conclusion

To address the discrepancy between timing results obtained before and after routing stage, we present a novel statistical timing prediction method in 14nm FinFET technology. Leveraging graph learning techniques, our approach achieves superior accuracy in estimating the +/-3σ delay of timing arcs. Additionally, a calibration method is introduced to minimize errors during delay propagation. Experimental results demonstrate higher precision of the proposed method and a remarkable runtime speed-up.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No.62274034).

References

[1] Barboza E C, Shukla N, et al. Machine learning-based pre-routing timing prediction with reduced pessimism[C]//DAC 2019: 1-6.

[2] He X, Fu Z, et al. Accurate timing prediction at placement stage with look-ahead RC network[C]//DAC 2022: 1213-1218.

[3] Guo Z, Liu M, et al. A timing engine inspired graph neural network model for pre-routing slack prediction[C]//DAC 2022: 1207-1212.

[4] Yang T, He G, et al. Pre-routing path delay estimation based on transformer and residual framework[C]//ASP-DAC 2022: 184-189.

[5] Chang K, Ahn J, Park H, et al. DTOC: integrating Deep-learning driven Timing Optimization into the state-of-the-art Commercial EDA tool[C]//DATE 2023: 1-6.

[6] Blaauw D, Chopra K, et al. Statistical timing analysis: From basic principles to state of the art[J]. IEEE transactions on computer-aided design of integrated circuits and systems, 2008, 27(4): 589-607.

[7] Huynh-Bao T, Ryckaert J, Tőkei Z, et al. Statistical timing analysis considering device and interconnect variability for BEOL requirements in the 5-nm node and beyond[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(5): 1669-1680.

[8] Jin L, Xu J, Fu W, et al. A Novel Delay Calibration Method Considering Interaction between Cells and Wires[C]//DATE 2023: 1-6.

[9] Ghanta P, Keller I. Importance of modeling non-Gaussianities in STA in sub-16nm nodes[C]//TAU 2016, 35.

[10] Frustaci F, Corsonello P, Perri S. Analytical delay model considering variability effects in subthreshold domain[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2012, 59(3): 168-172.

[11] Balef H A, Kamal M, Afzali-Kusha A, et al. All-region statistical model for delay variation based on log-skew-normal distribution[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 35(9): 1503-1508.

[12] Moshrefi A, Aghababa H, Shoaei O. Statistical estimation of delay in nano-scale CMOS circuits using Burr Distribution[J]. Microelectronics Journal, 2018, 79: 30-37.

[13] Kahng A B. New game, new goal posts: A recent history of timing closure[C]//DAC 2015: 1-6.

[14] Liang R, Xie Z, Jung J, et al. Routing-free crosstalk prediction[C]//ICCAD 2020: 1-9.

[15] 2021 OpenCores [online]. Available: https://opencores.org/