GTN-Cell: Efficient Standard Cell Characterization Using Graph Transformer Network

Lihao Liu, Yunhui Li, Beisi Lu, Li Shang *Member, IEEE,* Fan Yang *Member, IEEE,*

***Abstract*—Lookup table (LUT)-based libraries of standard cell characterization is crucial to accurate static timing analysis (STA). However, with the continuous scaling of technology nodes and the increasing complexity of circuit designs, the traditional non-linear delay model (NLDM) is progressively unable to meet the required accuracy for cell modeling. The current source model (CSM) offers a more precise characterization of cells at advanced nodes and is able to handle arbitrary electrical waveforms. However, the CSM is highly time-consuming because it requires extensive transistor-level simulations, posing severe challenges to efficient standard cell library design. This work presents GTN-Cell, an efficient graph transformer network (GTN)-based method for library-compatible LUT-based CSM waveform prediction of standard cell characterization. GTN-Cell represents the transistor-level structures of standard cells as graphs, learning the local structural information of each cell. By incorporating the transformer encoder into the model and embedding path-related positional encodings, GTN-Cell captures the global relationships between distant nodes within each cell. Compared with HSPICE, the GTN-Cell achieves an average error of 2.27% on predicted voltage waveforms among different standard cells and timing arcs while reducing the number of simulations by 70%.**

***Index Terms*—Static timing analysis, graph transformer networks, standard cell library, current source model.**

I. Introduction

With the rapid development of very large scale integrated (VLSI) circuits at advanced nodes, billions of transistors are integrated into a single chip. Gate-level static timing analysis (STA) plays an increasingly important role in the efficient design and optimization of VLSI circuits [1]. A high-quality cell library is essential for accurate and efficient STA [2]. Standard cell characterization generates cell libraries, which involves performing accurate transistor-level simulations for each cell [3]. The widely adopted nonlinear delay model (NLDM) stores the delay of a cell in a lookup table (LUT) format indexed by input slews and load capacitances, which provides high efficiency in timing analysis. However, with continuous process technology scaling, the NLDM faces accuracy challenges due to increased wire delays, noise, and Miller effects on smaller transistors [4]. Current source models (CSMs) have been proposed to address these issues, which can model arbitrary waveforms of standard cells with high accuracy [5]. The two widely adopted library-compatible LUT-based CSMs by industry are the effective current source model (ECSM) by Cadence and the composite current source (CCS) model by Synopsys [6].

For high-accuracy CSMs, the primary challenge lies in the high computational cost and extremely long runtime [3]. Firstly, transistor-level simulations are required for each input and output of thousands of standard cells, with a range of input slew and output load configurations, resulting in hundreds of combinations for each cell. Secondly, standard cell characterization must be performed under dozens of process, voltage, and temperature (PVT) corners to meet the demands of advanced node designs. Consequently, the characterization of a standard cell library can take several days to weeks, making it time-consuming and significantly reducing the efficiency of the VLSI design flow.

Traditional approaches have introduced new analytical models to perform cell characterization, improving the accuracy or efficiency of timing modeling [7]–[9]. However, library-compatible LUT-based CSMs are still widely used in the industry to accurately model standard cells at advanced technology nodes. With the rapid development of artificial intelligence (AI), many machine learning (ML)-based methods have been proposed to enhance standard cell library characterization. For instance, ridge regression [10], and multilayer perceptron (MLP)-based [11], [12] methods have been introduced to predict gate delays. Ye et al. [13] leverage graph attention network (GAT) [14] for cell aging-aware delay modeling, and Raslan et al. [6] employs an autoencoder-based method to compress the file size of CSM libraries. The aforementioned ML-based methods primarily focus on delay prediction for standard cell characterization. However, waveform-based models are adopted at advanced technology nodes instead of delay-based models to meet the accuracy requirements of cell characterization.

To tackle the challenge of excessive runtime in constructing accurate CSM libraries, this work proposes a novel standard cell characterization method that leverages a graph transformer network (GTN)-based approach. We focus on predicting standard cell output waveforms for library-compatible LUT-based CSMs, which are widely adopted in the industry. Inspired by the fact that transistor-level standard cells can naturally be represented as graphs, we develop a graph representation method to model the detailed structures of standard cells. In transistor-level standard cells, the relationships between transistors and timing paths significantly influence the output waveforms, making it essential to capture this information for accurate prediction. GTN-Cell leverages transformer-based structures with path-related positional encodings into graph neural networks (GNNs). This enables the model to learn local

This research is supported partly by National Key R&D Program of China 2019YFA0709600, 2019YFA0709602, partly by National Natural Science Foundation of China (NSFC) research projects 92373207 and 62090025.

Lihao Liu, Yunhui Li, Beisi Lu and Fan Yang are with State Key Lab of Integrated Chips and Systems, and School of Microelectronics, Fudan University, Shanghai, China. Li Shang is with State Key Lab of Integrated Chips and Systems, and School of Computer Science, Fudan University, Shanghai, China.

Corresponding author: Fan Yang (yangfan@fudan.edu.cn).

978-3-9826741-0-0/DATE25/© 2025 EDAA

relationships between transistor structures and the distant global information within each cell. With the efficient graph representation and the GTN model, GTN-Cell accurately predicts cell output voltage waveforms across different timing arcs. As a result, GTN-Cell reduces the number of simulations for CSM-based standard cell characterization, improving the efficiency of timing library construction.

The contributions of this work can be summarized as follows:

- A graph representation method for detailed transistor-level structures of standard cells, capturing the relationships between transistors and different timing arcs. Path-related positional encodings are proposed to allow the GTN model to capture global information between distant nodes within standard cells.
- A GTN-based model that efficiently captures the relationships between the detailed structures of each standard cell using a self-attention mechanism, generating graph embeddings to predict cell output voltage waveforms among different timing arcs.
- The predicted voltage waveforms of the proposed GTN-Cell method are compared to HSPICE results across various standard cells and timing arcs. With training on only 30% of the simulations that are typically required for LUT-based CSM standard cell characterization, the model achieved an average relative error of 2.27% in the predicted waveforms for the remaining 70% cases.

The rest of the paper is organized as follows: Section II provides the basic backgrounds of CSMs and GTNs. Section III describes the proposed GTN-based waveform characterization method of standard cells. Section IV presents the experimental results. Section V concludes the work.

II. Background

This section first outlines the basics of CSMs, followed by an overview of GNNs and GTNs.

A. Current Source Models

Standard cell characterization is the process to build accurate and efficient models for each cell, including its characteristics. LUT-based cell libraries are widely adopted to store these models, involving transistor-level simulations to extract relevant features for storage. The NLDM characterizes standard cells by recording delays indexed by input slews and load capacitances, including single receiver capacitance for each cell. LUT-based models use interpolation or extrapolation for unstored values [15]. However, NLDM becomes highly inaccurate for cell modeling in nanometer technologies as waveforms grow more complex. CSMs capture arbitrary output waveforms and offer higher accuracy for cell modeling in advanced nodes. The two widely adopted LUT-based CSMs by industry are the ECSM by Cadence and the CCS model by Synopsys [6].

The ECSM model stores piece-wise linear (PWL) output voltage waveforms indexed by input slews and output loads [16]. Fig. 1 provides an example of an ECSM LUT for standard cells [2]. For the receiver model, ECSM typically stores two different capacitance values corresponding to two delay

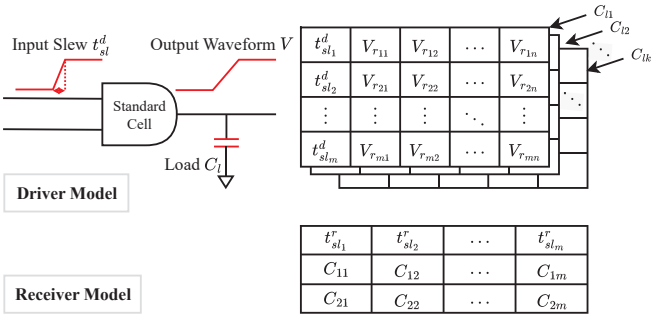

Fig. 1: Example of the ECSM format LUT for standard cells.

thresholds, accounting for the Miller effect [1]. By capturing more complex waveform behaviors of standard cells, ECSM achieves high modeling accuracy.

The CCS format differs from the ECSM format by storing current waveforms generated by the driver [17]. Despite this difference, the ECSM and CCS formats are essentially equivalent and can be transformed into each other [18]. While CSMs offer high modeling accuracy, constructing these libraries still requires a large number of transistor-level simulations, which are time-consuming. This poses significant challenges for standard cell characterization and slows the overall circuit design flow at advanced nodes.

B. Graph Transformer Networks

GNNs operate on graph data and learn various graph information, which have been applied in many electronic design automation (EDA) tasks [19]–[21]. Traditional GNN models automatically extract high-level embeddings for each node through an information aggregation and update framework. By stacking l GNN layers, the final output for each node from the GNN will have local information from its $l-$hop neighborhood.

Leveraging the message-passing process of GNNs and the self-attention mechanism in transformers [22], GTNs can effectively capture relationships between nodes in a graph. In the self-attention mechanism, attention scores α_{ij} between each pair of nodes v_i and v_j are calculated as follows [23]:

$$\alpha_{ij} = softmax_j\left(\frac{Qh_{v_i} \cdot Kh_{v_j}}{\sqrt{d_k}}\right), \tag{1}$$

where Q and K are learnable matrix parameters, h_{v_i} represents the embedding of node v_i, and d_k is the scaling factor. During the message-passing process, by multiplying the attention scores with node embeddings, the embedding h_{v_i} of node v_i in the graph is formulated as follows [23], [24]:

$$h^{l+1}_{v_i} = \phi\left(W^l_0 h^l_{v_i} + W^l_1 MEAN_{v_j \in N(v_i)}(\alpha^l_{ij} V^l h^l_{v_j})\right), \tag{2}$$

where W^l_0, W^l_1 and V^l are learnable matrix parameters of the $l-$th layer , $N(v_i)$ denotes the neighbors of node v_i and ϕ is the information update function such as MLP. By encoding positional information into the GTN, the model can capture global relationships between distant nodes [25].

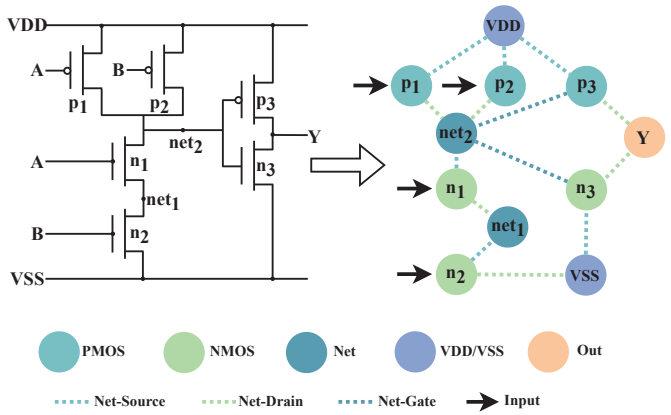

Fig. 2: Graph representation of the 2-input AND standard cell.

III. Graph Transformer-Based Waveform Characterization of Standard Cells

This section provides a detailed explanation of the proposed GTN-Cell method. First, we describe the graph representation method for transistor-level standard cells. Next, we discuss the GTN-based encoder to generate graph embeddings. Finally, we present the approach for predicting output waveforms using the generated graph embeddings and the model training process.

A. Graph Representation With Positional Encodings

Transistor-level structures of standard cells can naturally be represented as graphs. Fig. 2 provides an example of representing a 2-input AND gate as a graph. We define the graph as $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes and $E = \{e_{ij} | v_i, v_j \in V\}$ is the set of edges. During the graph construction, nodes represent transistors, internal nets, and the output pin, while edges are defined based on the connections between transistors. Unlike traditional graph convolutional network (GCN) [26], three levels of features and different types of nodes and edges are defined during the graph construction process, which captures the detailed structures of standard cells.

Firstly, we combine three different encoding methods to form the final node features.

- Type encoding: 6-dimensional one-hot vectors representing different node types, including PMOS, NMOS, Net, VDD, VSS, and Output. In order to further distinguish the relative input (the input pin of the timing arc) from other inputs, the type encoding for PMOS and NMOS is set to 2 when the relative input pin is directly connected to the transistor. Otherwise, the PMOS and NMOS type encoding is set to 1.
- Transistor encoding: 6-dimensional vectors that include channel width, length, and number of fins for transistor nodes. For PMOS, the features occupy the first three values in the vector, while for NMOS, they occupy the last three. For non-transistor nodes, these features are set to zero.
- Positional encoding: 5-dimensional vectors include node degree, the minimum distance to the output node, and the 3 smallest non-trivial Laplacian eigenvectors [23] for each node.

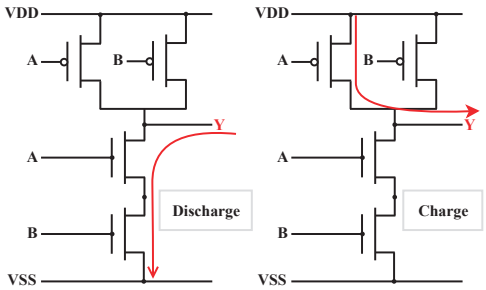

Fig. 3: An illustration of various timing paths in standard cells.

Type encoding and transistor encoding aim to capture various characteristics of the graph nodes that influence the shapes of standard cell output waveforms. Path-related positional encoding is introduced based on the observation that different timing arcs result in varying output delays for standard cells. As illustrated in Fig. 3, in a 2-input NAND cell, the delay of discharging and charging the load differs due to the current flowing through different numbers of transistors and paths. Path-related encoding is applied to each node to enable the model to learn such global path-related information, incorporating the minimum distance to the output and the node degree. To further enhance the model to learn graph-level information, the 3 smallest non-trivial Laplacian eigenvectors [23] for each node are also added to the positional encoding.

Next, for edge features, a 3-dimensional one-hot vector is proposed to represent different edge types, such as nets connected to the drain, source, or gate of the transistor. Finally, input slews, output loads, and whether the input waveform is rising or falling are collectively defined as slew-load encoding using 3-dimensional vectors. This encoding is assigned to each graph, enabling the model to learn the relationships between these parameters and the structure of the standard cells.

B. Graph Transformer-Based Encoder

Based on the graph representation enriched with detailed information of standard cells, the GTN-based encoder can generate high-level graph embeddings. The overall flow of the proposed GTN-Cell method is shown in Fig. 4, with Fig. 4(c) representing the GTN-based encoder. Node and edge features are first transformed into the same higher dimension and then passed through several graph transformer layers to produce the final node embeddings. Inspired by [23] and [22], the proposed structure of each GTN layer is depicted in Fig. 5. Multi-head attention is first applied to the input embeddings of nodes and edges, allowing the model to attend to different graph representations. The multi-head attention mechanism is shown on the right side of Fig. 5, where attention scores between each pair of nodes are calculated using Eq. (1). After multi-head attention, residual connections are applied to facilitate the training of deep neural networks. After the message-passing process of the GNN, based on Eq. (2), node and edge embeddings are

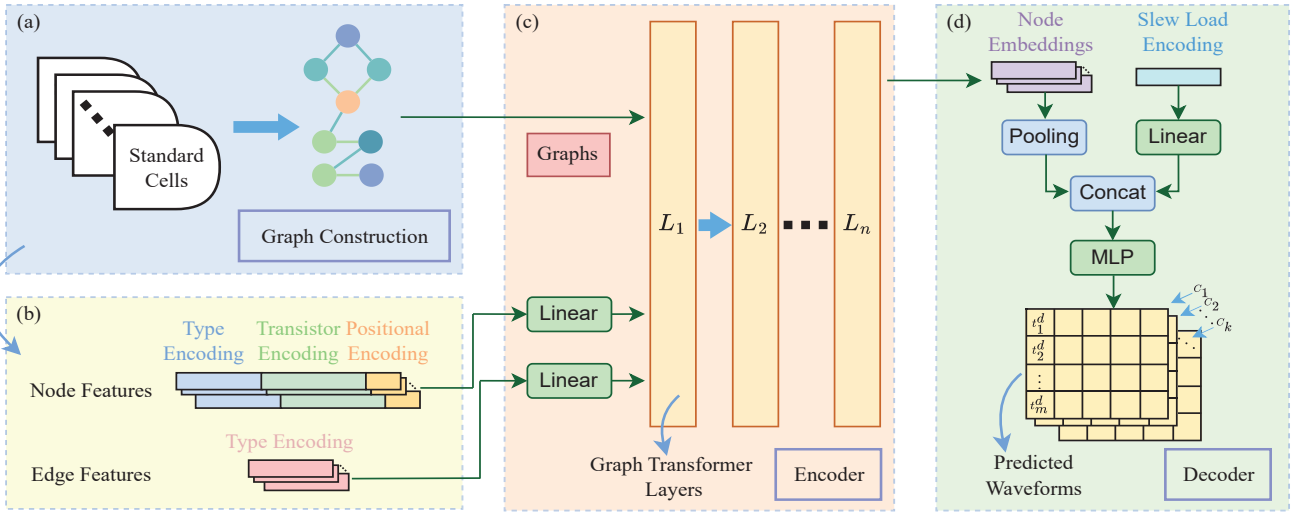

Fig. 4: The framework of GTN-Cell for output waveforms prediction of standard cells. (a) Graph construction process for various standard cells. (b) Feature assignment to nodes and edges. (c) Encoder generating embeddings using graph transformer layers. (d) Decoder predicting the final output waveforms based on the node embeddings from the encoder.

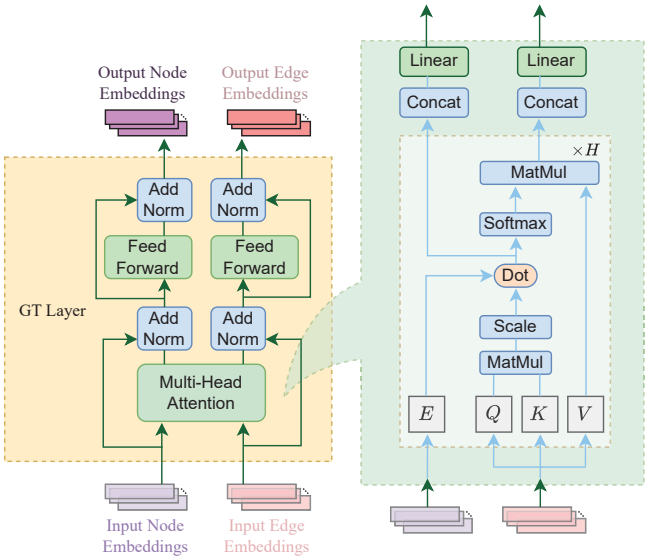

Fig. 5: The detailed structure of the graph transformer layer used in GTN-Cell.

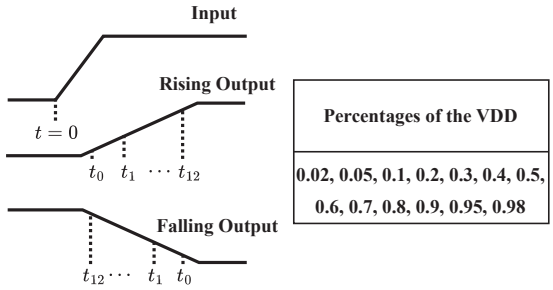

Fig. 6: An illustration of predicted time points in output waveforms reaching specific voltage percentage thresholds.

produced, capturing both local structural information and path-related global information.

C. Decoder for Waveform Prediction and Model Training

The GTN-based encoder generates powerful node embeddings that capture the structural information of standard cells. The decoder component of GTN-Cell, shown in Fig. 4(d), combines these embeddings with slew-load encoding and feeds the result into an MLP to predict the final output waveforms for standard cell characterization. This approach allows the model to learn the relationships between standard cell structures and input-load configurations.

Our framework uses two GTN layers, and all MLPs have one hidden layer of 128 neurons. The linear and feedforward layers follow the same configuration as the MLP. The number of attention heads is set to 8. We predict 13 time points of the output waveform, corresponding to the moments when the output voltage reaches specific percentage thresholds. Fig. 6 illustrates the prediction of the voltage waveform by time indices, with time starting from the input transition. Notably, if the output waveform is a falling edge, the 13 time points are stored in decreasing order, allowing the model to predict whether the output is rising or falling automatically. The mean square error (MSE) is used as the loss function during model training based on the predicted waveforms and the HSPICE simulated results. The learning rate is set to 0.001, and the batch size is set to 2100 based on the capability of the GPU.

IV. Experiments

In this section, we first outline the experimental setup. Next, we compare the predicted results with HSPICE simulations. Finally, we compare the results with other ML techniques.

TABLE I: Error of Predicted Results Compared with HSPICE

Methods		HSPICE	GTN-Cell
Waveforms	**Mean (%)**	0.00	2.27
	Std (%)	0.00	4.92
Delay	**Mean (%)**	0.00	1.79
	Std (%)	0.00	2.42
Transition Time	**Mean (%)**	0.00	3.26
	Std (%)	0.00	4.64
# of Simulations	**Values**	52430	**15729**
	Ratio	1	**0.30**

A. Experiment Setup

The GTN-Cell is trained and tested on a Linux server with an Nvidia Tesla V100 GPU and the Intel Xeon Silver 8358 CPU at 2.60GHz. The method is implemented in Python using the PyTorch framework and the PyTorch-geometric toolkit [27].

We build the dataset for training and testing based on the ASAP $7nm$ FinFET PDK [28]. Various types of standard cells, along with corresponding input slews and output lumped capacitive loads, are selected according to the PDK. The target standard cells include simple and complex gates, such as AND, NAND, OR, NOR, INV, AOI, AO, OA, and OAI, with varying inputs for each type, resulting in a total of 126 standard cells. The input slew values range from $5ps$ to $320ps$, and the output load capacitances range from $1.44fF$ to $92.16fF$. We select 7 input slews and 7 load capacitances for each gate within this range. Different timing arcs are also considered during dataset generation. For each input of the standard cell, two different rising or falling conditions are configured. A graph is based on the method discussed in Section III-A for each combination of standard cells, input slew, output load, timing arc, and rising/falling condition. HSPICE simulations using netlists with extracted parasitics are then conducted to extract the reference output waveforms for standard cell characterization. A total of 52430 graphs are constructed for the following experiments.

The training scheme follows the approach discussed in Section III-C. To demonstrate the ability of GTN-Cell to capture the relationships between standard cell structures and load slew configurations during the characterization process, we use only 30% of the data for training and testing, while the remaining 70% is completely unseen by the model.

B. Waveform Prediction Result Compared with HSPICE

To evaluate the accuracy of the trained model, we use HSPICE simulation results of the output waveforms from standard cells as the golden reference. The mean absolute relative error (MARE) is employed to calculate the error between predicted and reference values. The MARE is calculated as follows:

$$MARE = \frac{1}{N}\sum_{i=1}^{N}(|1 - \frac{y_o}{y}|), \tag{3}$$

where N is the number of predicted values, y_o means the predicted values and y means the reference values. Firstly, as shown in Fig. 7a, we compare the predicted and reference time points at which the waveforms reach specific voltage percentage thresholds, as previously discussed in Fig. 6. Next, based on the predicted waveforms, we evaluate the delay and transition times of each standard cell, as shown in Fig. 7b and Fig. 7c. The delay is defined as the time difference between the 50% crossing points of the input and output waveforms, while the rising/falling transition time is defined as the time difference between the 10% and 90% crossing points of the waveforms. The predicted values for waveforms, delay, and transition time are analyzed using the 70% of data unseen during the training and testing process. Furthermore, Fig. 7d shows the predicted rising and falling waveforms with 2% MARE compared with HSPICE. The figures in Fig. 7 demonstrate that the predicted values align closely with the HSPICE reference values.

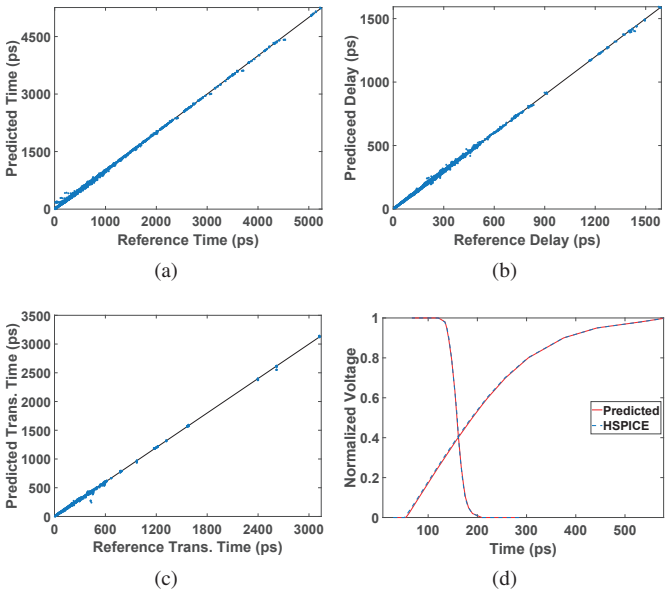

Fig. 7: Comparison between predicted results and reference HSPICE results. (a) Waveform time points at specific voltage percentages. (b) Computed delay. (c) Computed rising/falling transition time. (d) Rising and falling waveforms.

Detailed statistics are summarized in TABLE I, including the average and standard deviation of the MARE values between the predicted and reference waveforms. The results indicate that the predicted waveforms, compared to HSPICE, have an average error of 2.27% and a standard deviation of 4.92%. For the calculated delay and transition times, the average errors are only 1.79% and 3.26%, respectively. This demonstrates that the model can perform standard cell characterization tasks accurately, thereby improving the efficiency of constructing standard cell characterization libraries with fewer simulations.

Notably, the model requires only 30% of the simulations to predict all the remaining waveforms. Training can be completed within several minutes, which is negligible compared to the hours required for simulation. The average inference time of our model for a graph is $0.059ms$. GTN-Cell is capable of producing highly accurate output waveforms for standard cell characterization while significantly reducing the number of simulations.

TABLE II: Comparison Between Predicted Delay and Transition Time with HSPICE Results

Methods		MLP		GCN		GAT		GTN-Cell w/o pos		GTN-Cell	
		Value	Ratio	Value	Ratio	Value	Ratio	Value	Ratio	Value	Ratio
Waveforms	Mean (%)	7.38	3.25	5.37	2.37	3.81	1.68	3.38	1.49	**2.27**	**1**
	Std (%)	12.13	2.47	8.09	1.64	5.93	1.21	5.77	1.17	**4.92**	**1**
Delay	Mean (%)	5.93	3.31	4.51	2.52	3.17	1.77	2.73	1.53	**1.79**	**1**
	Std (%)	8.73	3.61	5.54	2.29	3.96	1.64	3.39	1.40	**2.42**	**1**
Transition Time	Mean (%)	13.42	4.12	7.02	2.15	4.91	1.51	4.56	1.40	**3.26**	**1**
	Std (%)	19.59	4.22	10.72	2.31	6.67	1.44	6.37	1.37	**4.64**	**1**

C. Comparison with Other ML Techniques

MLP and GAT are two primary ML techniques adopted for standard cell timing characterization with good results [10]–[13]. However, most existing works focus on predicting only delays for standard cell timing characterization, while CSM-based models with waveforms are required for accurate timing analysis in advanced process nodes. To show the effectiveness of the GTN-Cell, we implement MLP, GCN, and GAT-based methods for predicting output waveforms for standard cell characterization and compare our method with them. The results are summarized in TABLE II. For the MLP-based method, we train a model for each standard cell, which is commonly adopted in existing methods. The training time and 30% training data for other methods remain the same as ours, and testing on the remaining 70% data. The results show that our GTN-Cell method achieves the highest accuracy in predicting waveforms. Compared to MLP, GCN, and GAT-based methods, GTN-Cell leverages the transformer structure with path-related positional encoding, which captures relationships between both graph nodes and edges and learns distant information from a global perspective. We also evaluate the impact of positional encoding on GTN-Cell, as shown in TABLE II. The results indicate that without positional encoding, the error in predicted waveforms increases by 49%, highlighting the importance of positional encoding in enabling the model to learn distant information and improve prediction accuracy.

V. Conclusion

This paper presents GTN-Cell, an efficient GTN-based method for standard cell characterization, which focuses on output waveform prediction of standard cells for library-compatible LUT-based CSMs. By modeling the transistor-level structures of standard cells as graphs and encoding rich information at the node, edge, and graph levels, the GTN-based encoder effectively captures the relationships between cell structures and load slew configurations. Consequently, the decoder can accurately predict output waveforms. Compared to HSPICE, the GTN-Cell method achieves an average error of 2.27% in predicted waveforms while reducing the number of required simulations by 70%. Note that our methods can also be applied to accelerate standard cell characterization library construction with multi-PVT by training a model for each PVT corner.

References

[1] Dimitrios Garyfallou, Stavros Simoglou, Nikolaos Sketopoulos, Charalampos Antoniadis, Christos P Sotiriou, Nestor Evmorfopoulos, and George Stamoulis. Gate delay estimation with library compatible current source models and effective capacitance. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(5):962–972, 2021.

[2] Baljit Kaur, Naushad Alam, Sanjeev Kumar Manhas, and Bulusu Anand. Efficient ECSM characterization considering voltage, temperature, and mechanical stress variability. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(12):3407–3415, 2014.

[3] Kenza Charafeddine and Faissal Ouardi. Fast timing characterization of cells in standard cell library design based on curve fitting. In *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pages 1–6. IEEE, 2017.

[4] Xu He, Zhiyong Fu, Yao Wang, Chang Liu, and Yang Guo. Accurate timing prediction at placement stage with look-ahead RC network. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, pages 1213–1218, 2022.

[5] Behnam Amelifard, Safar Hatami, Hanif Fatemi, and Massoud Pedram. A current source model for CMOS logic cells considering multiple input switching and stack effect. In *Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 568–573, 2008.

[6] Waseem Raslan and Yehea Ismail. Deep learning autoencoder-based compression for current source model waveforms. In *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 1–6. IEEE, 2021.

[7] Sandeep Miryala, Baljit Kaur, Bulusu Anand, and Sanjeev Manhas. Efficient nanoscale vlsi standard cell library characterization using a novel delay model. In *2011 12th International Symposium on Quality Electronic Design*, pages 1–6. IEEE, 2011.

[8] Weiwei Shan, Yuqiang Cui, Wentao Dai, Xinning Liu, Jingjing Guo, Peng Cao, and Jun Yang. An efficient path delay variability model for wide-voltage-range digital circuits. *Science China. Information Sciences*, 66(2):129401, 2023.

[9] Tiansong Cui, Yanzhi Wang, Xue Lin, Shahin Nazarian, and Massoud Pedram. Semi-analytical current source modeling of finfet devices operating in near/sub-threshold regime with independent gate control and considering process variation. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 167–172. IEEE, 2014.

[10] Florian Klemme and Hussam Amrouch. Machine learning for on-the-fly reliability-aware cell library characterization. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(6):2569–2579, 2021.

[11] Dimitrios Garyfallou, Anastasis Vagenas, Charalampos Antoniadis, Yehia Massoud, and George Stamoulis. Leveraging machine learning for gate-level timing estimation using current source models and effective capacitance. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 77–83, 2022.

[12] Seyed Milad Ebrahimipour, Behnam Ghavami, Hamid Mousavi, Mohsen Raji, Zhenman Fang, and Lesley Shannon. Aadam: A fast, accurate, and versatile aging-aware cell library delay model using feed-forward neural network. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[13] Yuyang Ye, Tinghuan Chen, Zicheng Wang, Hao Yan, Bei Yu, and Longxing Shi. Fast and accurate aging-aware cell timing model via graph learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2023.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[15] Qin Tang, Amir Zjajo, Michel Berkelaar, and Nick van der Meijs. RDE-based transistor-level gate simulation for statistical static timing analysis. In *Proceedings of the 47th Design Automation Conference (DAC)*, pages 787–792, 2010.

[16] OVS Shashank Ram and Sneh Saurabh. Modeling multiple-input switching in timing analysis using machine learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 40(4):723–734, 2020.

[17] Igor Keller, King Ho Tam, and Vinod Kariat. Challenges in gate level modeling for delay and SI at 65nm and below. In *Proceedings of the 45th annual Design Automation Conference (DAC)*, pages 468–473, 2008.

[18] Peter Feldmann, Soroush Abbaspour, Debjit Sinha, Gregory Schaeffer, Revanta Banerji, and Hemlata Gupta. Driver waveform computation for timing analysis with multiple voltage threshold driver models. In *Proceedings of the 45th annual Design Automation Conference (DAC)*, pages 425–428, 2008.

[19] Pengju Chen, Dan Niu, Zhou Jin, Changyin Sun, Qi Li, and Hao Yan. TSA-TICER: A two-stage TICER acceleration framework for model order reduction. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.

[20] Haoxing Ren, George F Kokai, Walker J Turner, and Ting-Sheng Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[21] Yunfan Zuo, Yuyang Ye, Hongchao Zhang, Tinghuan Chen, Hao Yan, and Longxing Shi. A graph-learning-driven prediction method for combined electromigration and thermomigration stress on multi-segment interconnects. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.

[22] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[23] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[24] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[25] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[26] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[27] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[28] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016.