Multi-Corner Timing Macro Modeling With Neural Collaborative Filtering From Recommendation Systems Perspective

Kevin Kai-Chun Chang, Guan-Ting Liu, Chun-Yao Chiang, Pei-Yu Lee, and Iris Hui-Ru Jiang

***Abstract*—Timing macro modeling has been widely employed to enhance the efficiency and accuracy of parallel and hierarchical timing analysis. However, existing studies primarily focused on generating an accurate and compact timing macro model for single-corner libraries, making it difficult to adapt these approaches to multi-corner situations. This either incurs substantial engineering effort or results in significant performance degradation. To tackle this challenge, we offer a fresh perspective on the timing macro modeling problem by drawing inspiration from recommendation systems and formulating it as a matrix completion task. We propose a neural collaborative filtering-based framework capable of capturing the convoluted relationships between circuit pins and timing corners. This framework enables the precise identification of timing variant regions across different corners. Additionally, we design several training features and implement various training techniques to enhance precision. Experimental results show that our framework reduces model sizes by more than 10% compared to state-of-the-art single-corner approaches, while maintaining competitive timing accuracy and exhibiting significant runtime improvements. Furthermore, when applied to unseen corners, our framework consistently delivers superior performance, demonstrating its potential for use in off-corner chiplets in a heterogeneous integration system.**

***Index Terms*—Matrix completion, multiple corners, recommendation systems, timing macro modeling.**

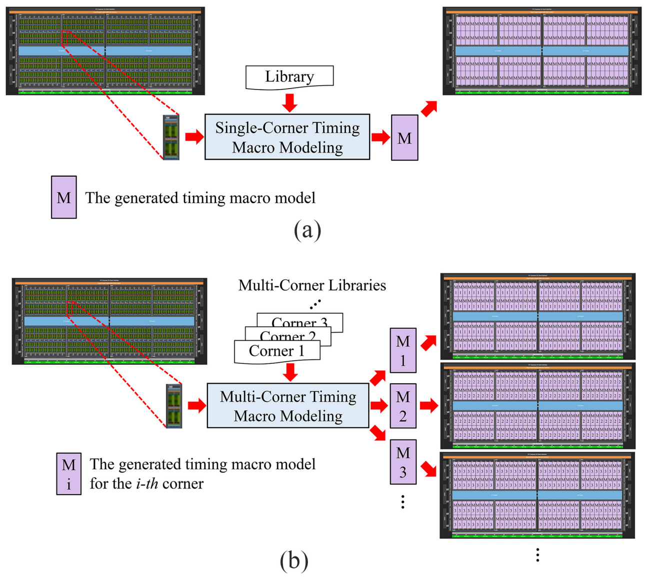

Library
Single-Corner Timing Macro Modeling
M
M The generated timing macro model
(a)
Multi-Corner Libraries
Corner 3
Corner 2
Corner 1
Multi-Corner Timing Macro Modeling
M 1
M 2
M 3
M i The generated timing macro model for the *i-th* corner
(b)

Fig. 1. Flows of timing macro modeling. (a) Single-corner flow. (b) Multi-corner flow. The circuit is the NVIDIA GH100 GPU quoted from [7].

I. Introduction

As THE design complexity continues to grow rapidly, timing analysis has become a significant bottleneck of the IC design flow. To address this issue, parallel and hierarchical timing analysis is widely adopted, which heavily relies on timing macro modeling. As shown in Fig. 1(a), a large design is first partitioned into several blocks; each block is then analyzed once, and a corresponding timing macro model is generated to encapsulate the timing properties of the block. Subsequently, the timing macro model can be reused for the same blocks and thus expedites the timing analysis process. In order to generate an accurate and concise timing macro model, timing variant pins (whose timing is affected by primary input (PI) slews or primary output (PO) loading) in a design should be preserved for accuracy, and timing invariant pins can be reduced for compactness.

Numerous timing macro modeling approaches have been proposed in the literature. Most of the approaches are algorithmic, employing a variety of graph-based algorithms during the extraction of timing macro models [1], [2], [3], [4], [5]. In contrast, we present a novel machine learning-based framework in [6]. By incorporating graph neural networks (GNNs), our framework effectively captures timing variant pins, and achieves timing accuracy comparable to state-of-the-art algorithmic methods while reducing the model size by 10%. Furthermore, our GNN-based framework can easily be applied to various timing analysis models and modes.

With the advancement of semiconductor technology, the timing analysis flow now encompasses the verification of numerous PVT corners (combinations of process, voltage,

Manuscript received 14 August 2023; revised 7 January 2024; accepted 6 March 2024. This work was supported in part by the National Science and Technology Council, Taiwan. This article was recommended by Associate Editor E. R. Keiter. *(Corresponding author: Iris Hui-Ru Jiang.)*

Kevin Kai-Chun Chang is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA.

Guan-Ting Liu is with NVIDIA Research Taiwan, Taipei 114066, Taiwan.

Chun-Yao Chiang is with Synopsys Inc., Taipei 110208, Taiwan.

Pei-Yu Lee is with Synopsys Inc., Hsinchu 300093, Taiwan.

Iris Hui-Ru Jiang is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106319, Taiwan (e-mail: huirujiang@ntu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2024.3383350

and temperature parameters) [8]. To ensure that accurate timing analysis can be performed quickly for all the corners, it is crucial to generate a dedicated timing macro model for each corner, as illustrated in Fig. 1(b). However, the aforementioned previous work, including our GNN-based framework, has mainly focused on single-corner timing macro modeling, making it difficult to extend to multi-corner scenarios. Accomplishing this either demands substantial engineering efforts or results in timing macro models with diminished accuracy and large sizes. The challenges become even more pronounced when conducting timing analysis on off-corner chiplets, where single-corner approaches are rendered impractical due to the absence of characterized libraries.

The main challenge of multi-corner timing macro modeling lies in accurately identifying the difference of timing variance on pins for each corner. This is a critical factor in achieving compact model sizes and high-timing accuracy. To tackle this challenge, we formulate the identification problem as a matrix completion task and utilize principles from recommendation systems. We introduce the concept of collaborative filtering, which effectively captures similarities between pins and utilizes observed data to infer timing variability. Furthermore, we employ neural collaborative filtering (NCF) to model the intricate interactions between pins and corners. Leveraging the neural network layers within the NCF model, we can grasp the complex relationships inherent in the multi-corner timing macro modeling problem.

The main contributions of this work are summarized as follows.

1) To the best of our knowledge, our work is the first to address and formulate the multi-corner timing macro modeling problem.
2) We offer a novel recommendation system-based perspective for timing macro modeling and formulate it as a matrix completion problem. This allows us to effectively capture the relationship between pins and corners.
3) We propose a NCF-based framework to learn the intricate pin-corner interactions. The NCF model not only achieves competitive timing accuracy and reduces macro model sizes by more than 10% compared to the state-of-the-art works in million-gate/instance-scale designs, but also demonstrates a 16X faster model training time. Additionally, the NCF model accelerates the training label generation process by 4.4X.
4) Our framework consistently preserves exceptional performance when applied to unseen corners, demonstrating its potential for utilization in off-corner chiplets.

The remainder of this article is organized as follows: Section II formulates the single-corner timing macro modeling problem, reviews the previous work on single-corner timing macro modeling, and details our GNN-based framework proposed in [6]. Section III discusses the challenges and formulates the multi-corner timing macro modeling problem. Section IV details our multi-corner timing macro modeling framework. Section V shows experimental results. Finally, Section VI concludes this work.

II. Single-Corner Timing Macro Modeling

A. Problem Formulation

In this work, we follow the problem formulation from TAU 2016 and 2017 contests [9], [10], which is also adopted by most previous work. The *single-corner timing macro modeling* problem can be defined as follows.

Given a circuit netlist (**.v* files) along with its parasitics (**.spef* files) and the early and late cell libraries (**.lib* files), the goal is to generate a timing macro model that encapsulates the timing behaviors of the design.

The generated timing macro model is evaluated based on two primary criteria: 1) timing accuracy (the higher, the better) and 2) the macro model size (the lower, the better). Note that there exists a tradeoff between them. Furthermore, the runtime required to generate the timing macro model is also crucial.

B. Previous Work

Interface logic models (ILMs) and extracted timing models (ETMs) [1] are two pioneering single-corner timing macro modeling approaches. ILM preserves circuit netlists from PI or PO ports to the first level of registers (i.e., interface logic) while eliminating register-to-register paths. In contrast, ETM consists solely of context-independent timing arcs between external pins. Subsequent works often build upon either of these two paradigms. Generally, ILM-based approaches [2], [3], [4] achieve exceptionally high-timing accuracy but suffer from larger macro model sizes. Conversely, ETM-based approaches [5] can extract smaller macro models at the expense of timing accuracy. Moreover, ETM-based methods are suitable for IP-reuse scenarios, as they can conceal circuit implementations, while ILM-based methods are more adaptable to advanced timing analysis modes, such as common path pessimism removal (CPPR).

For ILM-based approaches, LibAbs [2] and its following work [4] propose several graph reduction techniques that are applied alternately to timing graphs. This iterative process results in a more concise timing macro model. iTimerM [3] divides the circuit netlist into constant and variant timing regions, where the constant timing region is eliminated to achieve compactness. The separation is based on the propagation of minimum/maximum slew values, designating pins with stabilized slew ranges as constant. On the other hand, ATM [5] builds upon the ETM paradigm. It also adopts slew propagation to identify checkpoint pins, which are then inserted into the ETM model to improve timing accuracy. Fig. 2 summarizes the existing single-corner timing macro modeling frameworks.

To facilitate analysis efficiency, the key objective of timing macro modeling is to strike a balance between timing accuracy and the size of the generated timing macro model. Nevertheless, previous work adopts some heuristic techniques during their model extraction, which may cause degradation on the solution quality. For instance, LibAbs [2], [4] applies in-tree and out-tree graph reductions alternatively, based on the observation on the timing arc forms of cells or nets. Besides, some works need to set a threshold for variant pins

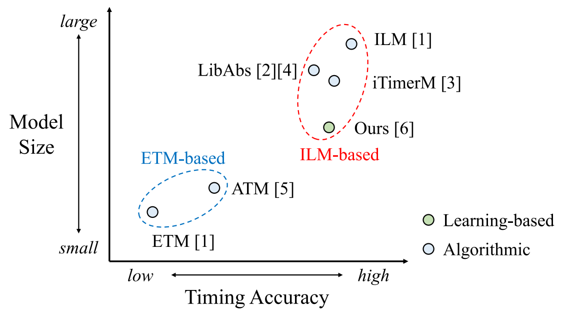


Fig. 2. Comparison of previous work and our framework.

identification, which requires considerable engineering effort, and the same threshold may not be applicable for various circuit designs. For example, iTimerM [3] uses a threshold to separate the variant regions with the constant region, and ATM [5] uses a threshold to determine which pins are dirty. Therefore, there is still room for improvement.

C. Overview of Our GNN-Based Framework

To overcome the deficiencies of prior work, we propose a GNN-based single-corner timing macro modeling framework [6]. GNNs have been developed to apply deep learning methods to graph data [11]. In a typical GNN scheme, node information is aggregated and transformed between neighbors recursively. After several neural network layers, a high-level representation of each node is extracted, which encapsulates the features and structures of the node's neighborhood.

There are several reasons that GNN is suitable for the timing macro modeling problem. First, the evaluation of timing criticality on circuit pins is usually challenging for heuristic-based methods. Nevertheless, graph-learning-based methods could capture implicit properties of circuit pins and thus evaluating timing importance more precisely. Second, the aggregation of node attributes in GNN is similar to the propagation of timing values on timing graphs, as shown in Fig. 3. Consequently, the timing properties of circuit pins could be captured and learned by GNN models smoothly. Third, due to the information exchange mechanism in GNN, the final representations of adjacent nodes tend to become similar. This property is desired in timing macro modeling since neighbor pins are usually of comparable degrees of timing criticality. Lastly, it is natural to represent circuit netlists by graphs, and thus GNNs could be easily embedded into the timing macro modeling framework.

Fig. 4 illustrates the proposed timing macro modeling framework. In the first stage, the timing sensitivity (TS) of each circuit pin is evaluated to reflect the influence of each pin on the overall timing accuracy. Then, the training data is generated accordingly. In the second stage, we adopt GNN models to learn the properties of circuit designs and predict the timing sensitivities of testing data. Finally, starting from the ILM, timing macro models are generated based on our timing sensitivities prediction. Different from previous work, which mainly focuses on nonlinear delay model (NLDM), our framework could also be applied to other advanced node timing analysis models, such as CCS, AOCV, and POCV, or different timing modes like CPPR. The generality of our framework comes from the fact that timing sensitivities could be adaptively evaluated depending on the given timing delay model. Moreover, the GNN models could effortlessly capture the corresponding timing properties.

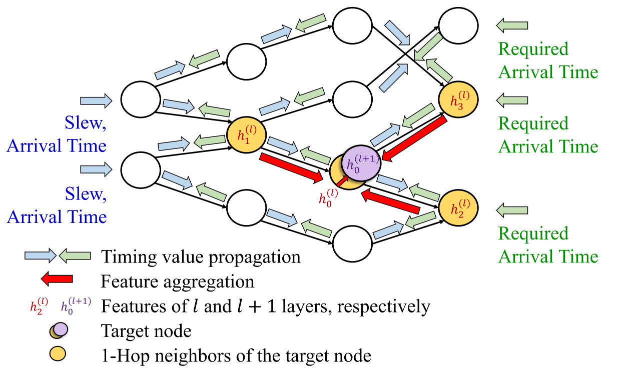


Fig. 3. Analogy between GNN aggregation and timing propagation. Timing values, including slew, arrival time, and required arrival time are propagated through edges (blue and green arrows). On the other hand, node features of layer l, $h_i^{(l)}$, are aggregated through edges and transformed into node features of layer $l+1$, $h_i^{(l+1)}$ (red arrows).

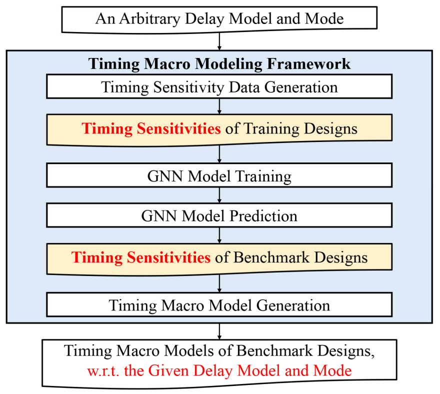

An Arbitrary Delay Model and Mode
Timing Macro Modeling Framework
Timing Sensitivity Data Generation
Timing Sensitivities of Training Designs
GNN Model Training
GNN Model Prediction
Timing Sensitivities of Benchmark Designs
Timing Macro Model Generation
Timing Macro Models of Benchmark Designs, w.r.t. the Given Delay Model and Mode

Fig. 4. Overview of our GNN-based single-corner timing macro modeling framework.

D. Timing Sensitivity Data Generation

1) Timing Sensitivity: In order to generate a high-quality timing macro model, we need to precisely evaluate the influence of each circuit pin on the timing accuracy of the whole design. Then, pins with subtle influences could be waived to reduce the model size, and meanwhile the timing accuracy will not be degraded.

The lower section of Fig. 7 (highlighted by red dashed lines) shows how we evaluate the TS of each pin. Given the input circuit graph, we first randomly generate several sets of boundary timing constraints. For each timing constraint, we store the corresponding timing analysis results of ILM as references. In the TS evaluation stage, we remove a pin from the circuit each time. After the removal, we perform timing propagation based on each set of boundary timing constraints generated and compute the differences between the current and the reference timing values (including slew, arrival time (at), required arrival time (rat), and slack) at the boundary pins. Finally, TS of a pin (for convenience, denoted

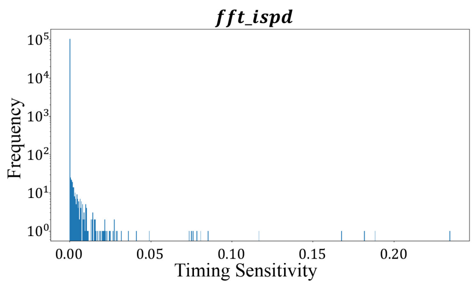


Fig. 5. TS distribution of *fft_ispd*.

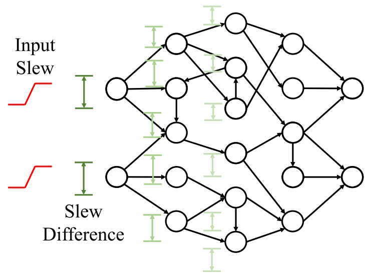


Fig. 6. SD and shielding effect.

as A_i in the following discussion) is set as the average of timing value differences under the different timing constraints. Equations (1) and (2) define the TS of pin A_i, where $\mathbb{B}$ denotes the collection of generated boundary timing constraints, and $slew^b_{P,\text{before}}$ (*resp.* $slew^b_{P,\text{after}}$) denotes the slew value of a boundary pin P under the timing constraint b before (*resp.* after) pin A_i's removal. The definitions of $\Delta at^b_{A_i}$, $\Delta rat^b_{A_i}$, and $\Delta slack^b_{A_i}$ are similar to that of $\Delta slew^b_{A_i}$

$$\text{TS}_{A_i} = \text{AVG}_{b\in\mathbb{B}}\left(\frac{\Delta slew^b_{A_i} + \Delta at^b_{A_i} + \Delta rat^b_{A_i} + \Delta slack^b_{A_i}}{4}\right) \tag{1}$$

$$\Delta slew^b_{A_i} = \frac{1}{|PI \cup PO|}\Sigma_{P\in PI\cup PO}\frac{slew^b_{P,\text{after}} - slew^b_{P,\text{before}}}{slew^b_{P,\text{before}}}. \tag{2}$$

2) Insensitive Pins Filtering: Although the TS evaluation flow could accurately compute the influence of each pin on the overall timing accuracy, running the flow for all the pins is time-consuming as we need to perform timing propagation once in each iteration. To enhance the efficiency, we first observe that the majority of the pins have extremely small or even zero TS. It is due to the nature of timing graph that most of the pins have subtle influences on the overall timing accuracy. For example, the TS distribution of circuit *fft_ispd* is shown in Fig. 5, where 70% pins have zero TS, while only few pins have large TS. Therefore, if we can find a rapid screening method to filter the insensitive pins first, we could perform TS evaluation flow on the potential critical pins only.

Timing value difference propagation is a suitable method for insensitive pins filtering. At each PI or PO port, two timing values, $t_{\min}$ and $t_{\max}$, are set up. We then propagate the timing values through the design and monitor the difference between the two timing values at each pin. According to the shielding effect, as shown in Fig. 6, the difference decays after several levels, and pins with small difference tend to have subtle influence on the overall timing accuracy. Inspired by previous works [3], [5], we choose slew to propagate from each PI. After the propagation, the slew difference (SD) at each pin is standardized, and pins with SD less than a threshold is filtered out. Fig. 7 illustrates the whole training data generation flow.

E. GNN-Based Timing Macro Modeling

1) GNN Model Training and Prediction: With the TS training data, GNN models could learn and predict accordingly. In this work, we adopt GraphSAGE [12] as our main GNN engine. For each node v, (3) first aggregates the node

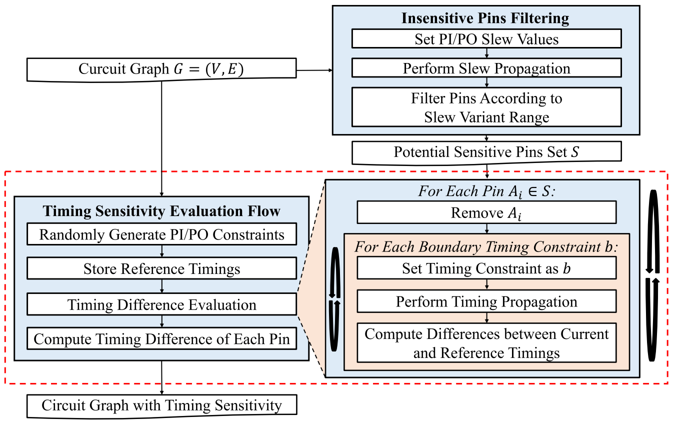

Insensitive Pins Filtering
Set PI/PO Slew Values
Perform Slew Propagation
Filter Pins According to Slew Variant Range
Curcuit Graph $G = (V, E)$
Potential Sensitive Pins Set S
Timing Sensitivity Evaluation Flow
Randomly Generate PI/PO Constraints
Store Reference Timings
Timing Difference Evaluation
Compute Timing Difference of Each Pin
For Each Pin $A_i \in S$:
Remove A_i
For Each Boundary Timing Constraint b:
Set Timing Constraint as b
Perform Timing Propagation
Compute Differences between Current and Reference Timings
Circuit Graph with Timing Sensitivity

Fig. 7. TS training data generation flow.

features from its neighborhood $\mathcal{N}(v)$, then (4) concatenates and encodes the representation of node v with the aggregated vector. In the experiments, only four rounds of aggregations and encodings are performed, as the timing property of a node is mostly influenced by its neighborhood. Other existing GNN models, such as GCN [13] or even self-defined GNN models, could also be embedded with our framework

$$h^k_{\mathcal{N}(v)} \longleftarrow \text{AGGREGATE}_k\left(h^{k-1}_u, \forall u \in \mathcal{N}(v)\right) \tag{3}$$

$$h^k_v \longleftarrow \sigma\left(W^k \cdot \text{CONCAT}\left(h^{k-1}_v, h^k_{\mathcal{N}(v)}\right)\right). \tag{4}$$

We treat the GNN prediction as a classification problem and convert the training labels of pins to {0, 1}. A pin's label is set to 1 if its TS is not zero. In addition, for CPPR mode, labels of multiple-fan-out pins of clock networks are also set to 1, as previous work, e.g., [14], suggests their importance for CPPR calculation.

The training features are listed in Table I. The features are all basic circuit properties which could be extracted within linear time. Features beginning with *"is"* are of {0, 1} Boolean values. For integer type features like *level_from_PI*, *level_to_PO*, and *out_degree*, the values are normalized to [0, 1] so that each feature have the same level of influences.

2) Timing Macro Model Generation: Fig. 8 details the timing macro model generation stage. First, we construct the initial timing graph and capture the interface logic netlist to construct ILM. Second, we apply both serial and parallel merging techniques to simplify the timing graph and retain only the timing variant pins identified by the GNN model. For serial merging, the delay of a merged edge is the sum of the original ones, while the slew inherits the last edge. For parallel

TABLE I
TRAINING FEATURES

Feature	Description
level_from_PI	The minimum level from a PI to the pin
level_to_PO	The min. level from the pin to a PO
is_last_stage_fanout	If the pin is the fanout of a last stage pin
is_last_stage	If the pin is the last stage of the timing graph
is_first_stage	If the pin is the first stage of the timing graph
out_degree	The number of output edges of the pin
is_clock_network	If the pin belongs to clock network
is_ff_clock	If the pin is the clock pin of a flip-flop
cell_type	The enumeration number of the cell type
rise_in_size	The fan-in size of the rise pin
rise_out_size	The fan-out size of the rise pin
fall_in_size	The fan-in size of the fall pin
fall_out_size	The fan-out size of the fall pin
is_CPPR	If the pin is crucial for CPPR

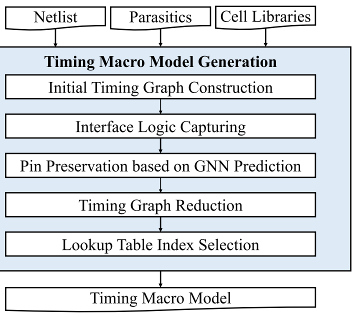

Netlist
Parasitics
Cell Libraries
Timing Macro Model Generation
Initial Timing Graph Construction
Interface Logic Capturing
Pin Preservation based on GNN Prediction
Timing Graph Reduction
Lookup Table Index Selection
Timing Macro Model

Fig. 8. Timing macro model generation.

merging, delay or slew is the minimum (*resp.* maximum) of the original edge values in the early (*resp.* late) mode. Afterward, we apply the lookup table index selection method proposed in [3], where indices that minimize the interpolation timing error are selected. Lastly, the timing macro model is generated.

III. Multi-Corner Timing Macro Modeling

A. Corner Explosion and Challenges

With the continuous advances in the semiconductor industry, design verification now involves examinations under numerous PVT corners [8]. This rigorous verification process is essential to ensure the robustness of a design under possible operating conditions. Several approaches have been proposed to address the challenges associated with multi-corner timing analysis [15], [16], [17], [18], [19]. Nevertheless, none of the previous work has focused on multi-corner timing macro modeling. The groups of timing variant pins may vary a lot across different corners. Consequently, adapting the single-corner timing macro modeling frameworks mentioned in Section II to multi-corner settings presents significant engineering effort or leads to substantial performance degradation. For example, iTimerM [3] and ATM [5] need to fine-tune distinct thresholds for each corner. Similarly, although our single-corner timing macro modeling framework is applicable across different corners, users are still required to extract training data and train an exclusive GNN model for each corner. It becomes even more challenging when dealing with chiplets that deviate from typical corners. To the best of our knowledge, existing approaches lack feasibility in identifying timing variant pins without the presence of a characterized corner library.

B. Problem Formulation

We extend the problem formulation of single-corner timing macro modeling defined in Section II-A to encompass multi-corner settings. The only differences are the input cell libraries are now associated with a set of timing corners, and the framework should generate a dedicated timing macro model for each corner within the corner set or for each unseen corner.

IV. Our Multi-Corner Timing Macro Modeling Framework

A. Problem Modeling

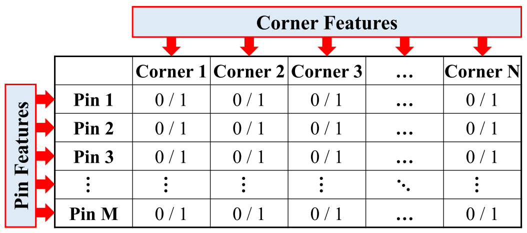
The primary challenge of the multi-corner timing macro modeling problem lies in the precise identification of timing variant pins for each corner. This allows us to build a compact timing macro model for each corner while preserving high-timing accuracy. To effectively tackle the issue of identifying timing variant pins, we have discovered that it can be approached from the perspective of recommendation systems because the relationship between pins and corners resembles the user-item relationship in recommendation systems. In the context of recommendation systems, the goal is to capture user-item interactions and provide personalized suggestions. Likewise, in the multi-corner timing macro modeling problem, our objective is to understand the interactions between pin features and corner properties and determine whether a pin is timing variant under a specific corner. By reframing the problem in this manner, we can leverage the well-established techniques from the field of recommendation systems to improve the identification of timing variant pins.

A common approach to interpret user-item interactions is to construct a user-item matrix, where each element represents the relationship between the corresponding user and item [20]. Then, it becomes a matrix completion problem that aims to determine the missing entries based on known user-item interactions. Similarly, we can formulate pin-corner relationships as a matrix completion problem. Based on this formulation, our framework has the potential to infer the variability of pins in new designs or assess the degrees of variability under unseen corners.

With this understanding, we can formally define the timing variant pin identification problem in the context of matrix completion, where each circuit pin serves as a user, and each corner is considered an item. The pin-corner matrix $Y \in \mathbb{R}^{M\times N}$ for M pins and N corners is defined as

$$y_{A_iC_k} = \begin{cases} 1, & \text{if pin } A_i \text{ is timing variant under corner } C_k \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Fig. 9 illustrates the modeling of the pin-corner matrix. The goal of the matrix completion process is to determine whether each element $y_{A_iC_k}$ is 0 (pin A_i is not timing variant under corner C_k) or 1 (A_i is timing variant under C_k).



	Corner 1	Corner 2	Corner 3	…	Corner N
Pin 1	0 / 1	0 / 1	0 / 1	…	0 / 1
Pin 2	0 / 1	0 / 1	0 / 1	…	0 / 1
Pin 3	0 / 1	0 / 1	0 / 1	…	0 / 1
⋮	⋮	⋮	⋮	⋱	⋮
Pin M	0 / 1	0 / 1	0 / 1	…	0 / 1

Corner Features
Pin Features

Fig. 9. Visualization of a pin-corner matrix.

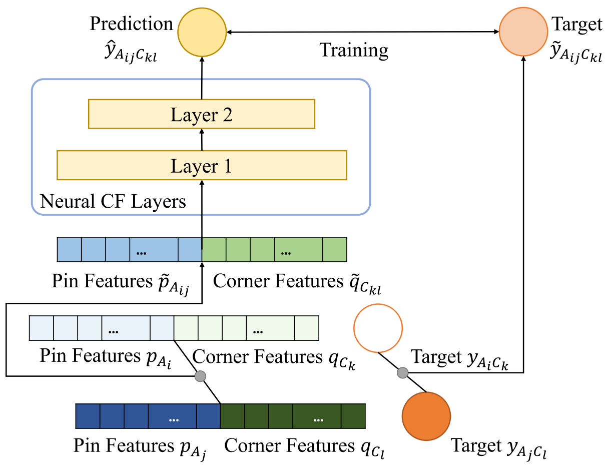

Prediction $\hat{y}_{A_{ij}C_{kl}}$
Training
Target $\tilde{y}_{A_{ij}C_{kl}}$
Layer 2
Layer 1
Neural CF Layers
Pin Features $\tilde{p}_{A_{ij}}$
Corner Features $\tilde{q}_{C_{kl}}$
Pin Features p_{A_i}
Corner Features q_{C_k}
Target $y_{A_iC_k}$
Pin Features p_{A_j}
Corner Features q_{C_l}
Target $y_{A_jC_l}$

Fig. 10. Structure of the NCF network and data augmentation with *mixup*.

B. Collaborative Filtering

To address the pin-corner matrix completion problem, we employ the concept of *collaborative filtering*, which identifies shared interests among users, such as the tendency of users, that purchase item A also purchase item B, and leverages this information to provide similar recommendations to users with similar preferences. This attribute is beneficial in the problem of multi-corner timing variant pin identification, as pins with similar properties often exhibit comparable levels of timing variability under the same corner.

By employing collaborative filtering, we can express the interaction between the pin A_i and the corner C_k as a function of the pin embedding p_{A_i} and the corner embedding q_{C_k}

$$\hat{y}_{A_iC_k} = f\left(A_i, C_k | p_{A_i}, q_{C_k}\right) \tag{6}$$

where f can represent any relationship between pin A_i and corner C_k. The pin embedding p_{A_i} and the corner embedding q_{C_k} can be extracted feature vectors as described in Section IV-C. The interaction function f can be optimized by employing target functions that minimize the discrepancy between $\hat{y}_{A_iC_k}$ and $y_{A_iC_k}$, where $y_{A_iC_k}$ refers to the corresponding element in the pin-corner matrix as described in (5).

C. Neural Collaborative Filtering (NCF) and Training Features

Since the identification of timing variant pins involves many factors (e.g., topological complexity, operating condition), it is crucial to find an interaction function f that can capture nonlinear relationships between pin features and their timing variability under different corners. Thus, we employ the *NCF* model [20], which consists of fully connected neural network layers, as shown in Fig. 10. In contrast to simple interaction functions, such as the inner product, the NCF model demonstrates the potential to capture complex and nonlinear relationships between pins and corners.

TABLE II
CORNER FEATURES FOR TRAINING

Feature	Description
process	The process of the corner (fast: 0.9, typical: 0.6, slow: 0.3)
voltage	The operating voltage of the corner
temperature	The operating temperature of the corner
inv_cell_rise	The flattened cell-rise lookup table of the inverter
inv_rise_transition	The flattened rise-transition lookup table of the inverter
inv_cell_fall	The flattened cell-fall lookup table of the inverter
inv_fall_transition	The flattened fall-transition lookup table of the inverter

In our NCF model, each input vector consists of the pin feature vector p_{A_i} and the corner feature vector q_{C_k}. For pin features, we adopt the same circuit topology features as our single-corner timing macro modeling framework, as described in Table I and Section II-E1. On the other hand, as shown in Table II, corner features include the temperature, voltage, and process of the corner. Furthermore, given that an inverter consists of a pMOS and an nMOS, it can effectively represent the switching characteristics of the corresponding corner library. Thus, we also incorporate the elements in the timing lookup tables of the inverter cell as our corner features. Lookup tables of other primitive cells (e.g., NAND, AND, etc.) might also be included to further enhance the model's learning quality.

After the extraction and concatenation of pin features p_{A_i} and corner features q_{C_k}, the corresponding timing variability $\hat{y}_{A_iC_k}$ can be inferred as follows:

$$\hat{y}_{A_iC_k} = f_\Phi\left(\left[p_{A_i}, q_{C_k}\right]\right) \tag{7}$$

where $[\cdot, \cdot]$ is the concatenation operator, and f_Φ is the interaction function with a set of parameters Φ.

In our NCF model, we use two layers of fully connected neural network. Thus, the interaction function f_Φ can be further expressed as

$$f_\Phi\left(\left[p_{A_i}, q_{C_k}\right]\right) = \sigma\left(f_{\Phi_2}\left(\tanh\left(f_{\Phi_1}\left(\left[p_{A_i}, q_{C_k}\right]\right)\right)\right)\right) \tag{8}$$

where $\Phi = \{\Phi_1, \Phi_2\}$, Φ_1, and Φ_2 correspond to the parameters of the first and the second NCF layer, respectively, $\tanh(\cdot)$ denotes the hyperbolic tangent function which serves as the activation function, and $\sigma(\cdot)$ is the sigmoid function which maps the predicted values to the range of [0, 1].

Since the target $y_{A_iC_k}$ of each training data $[p_{A_i}, q_{C_k}]$ is of a binary value, we adopt the binary classification loss function to optimize the model

$$\mathcal{L} = -\sum_{(A_i,C_k)\in\mathbb{X}} y_{A_iC_k} \log \hat{y}_{A_iC_k} + \left(1 - y_{A_iC_k}\right) \log\left(1 - \hat{y}_{A_iC_k}\right) \tag{9}$$

where $\mathbb{X}$ denotes the training label set which will be discussed in Section IV-D. We use the Adam optimizer [21] to minimize the loss defined by (9). Other design choices (e.g., increasing the number of neural network layers, using different activation functions, etc.) might be adopted to further improve the performance.

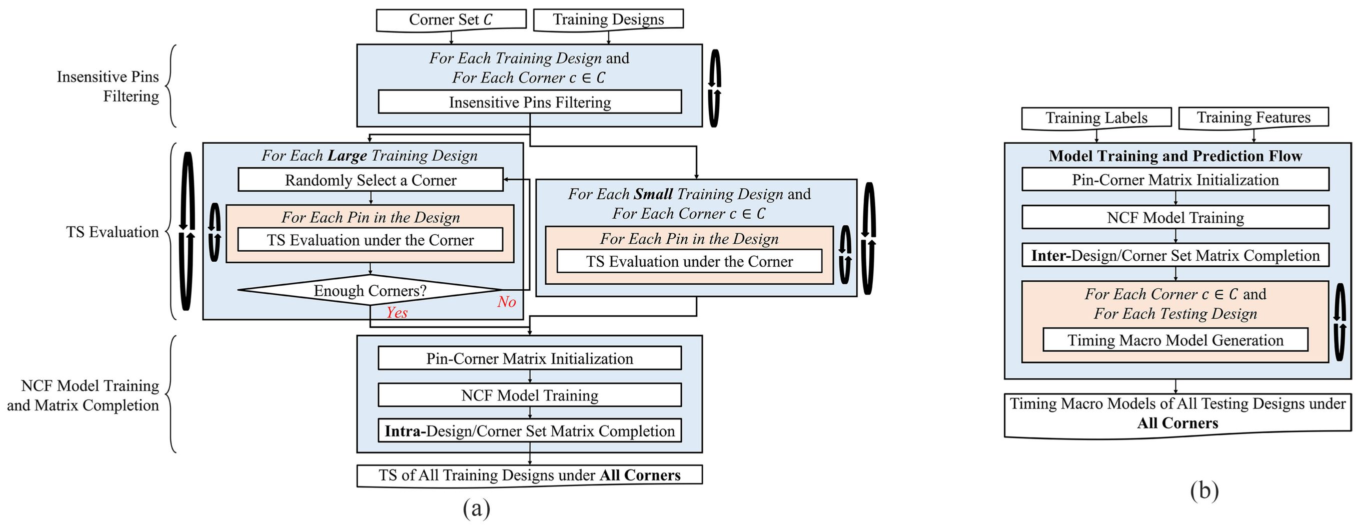

Corner Set C
Training Designs
Insensitive Pins Filtering
For Each Training Design and For Each Corner $c \in C$
Insensitive Pins Filtering
TS Evaluation
For Each Large Training Design
Randomly Select a Corner
For Each Pin in the Design
TS Evaluation under the Corner
Enough Corners?
Yes
No
For Each Small Training Design and For Each Corner $c \in C$
For Each Pin in the Design
TS Evaluation under the Corner
NCF Model Training and Matrix Completion
Pin-Corner Matrix Initialization
NCF Model Training
Intra-Design/Corner Set Matrix Completion
TS of All Training Designs under All Corners
(a)
Training Labels
Training Features
Model Training and Prediction Flow
Pin-Corner Matrix Initialization
NCF Model Training
Inter-Design/Corner Set Matrix Completion
For Each Corner $c \in C$ and For Each Testing Design
Timing Macro Model Generation
Timing Macro Models of All Testing Designs under All Corners
(b)

Fig. 11. (a) Training label generation flow. (b) Flow of NCF model training, prediction, and timing macro model generation on testing data.

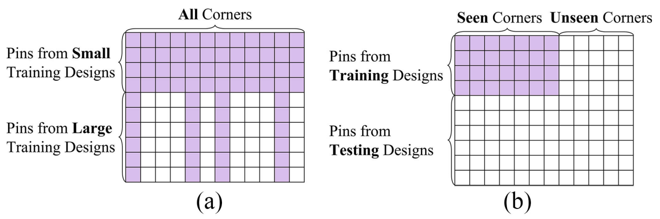

All Corners
Pins from Small Training Designs
Pins from Large Training Designs
(a)
Seen Corners
Unseen Corners
Pins from Training Designs
Pins from Testing Designs
(b)

Fig. 12. (a) Intra-design/corner set matrix completion. (b) Inter-design/corner set matrix completion. Initially, timing variability in the purple region is known, while the goal is to infer the timing variability of the white region.

D. Training Label Generation

To enhance the capability of timing variant pin identification of our NCF model, high-quality training labels are essential. Drawing from the achievements of our single-corner timing macro modeling framework, we leverage two key processes, *insensitive pins filtering* and *TS evaluation*, to generate our training labels. The details of these processes are presented in Fig. 7 and Section II-D. By extending the concepts of (1) and (2), we can define the TS in the context of multi-corner timing macro modeling

$$\mathrm{TS}_{A_iC_k} = \mathrm{AVG}_{b\in\mathbb{B}}\left(\frac{\Delta slew^b_{A_iC_k} + \Delta at^b_{A_iC_k} + \Delta rat^b_{A_iC_k} + \Delta slack^b_{A_iC_k}}{4}\right) \tag{10}$$

$$\Delta slew^b_{A_iC_k} = \frac{1}{|PI \cup PO|}\Sigma_{P\in PI\cup PO}\frac{slew^b_{PC_k,\mathrm{after}} - slew^b_{PC_k,\mathrm{before}}}{slew^b_{PC_k,\mathrm{before}}} \tag{11}$$

where $\mathrm{TS}_{A_iC_k}$ denotes the TS of pin A_i under timing corner C_k, $\mathbb{B}$ denotes the set of boundary timing constraints, $slew^b_{PC_k,\mathrm{before}}$ and $slew^b_{PC_k,\mathrm{after}}$ denote the slew value at boundary pin P under corner C_k before and after the removal of A_i, respectively, and $\Delta at^b_{A_iC_k}$, $\Delta rat^b_{A_iC_k}$, and $\Delta slack^b_{A_iC_k}$ can be obtained similar to (11). After the TS evaluation, we have to convert the TS values into binary to fill in the pin-corner matrix. In our work, a pin A_i is deemed timing variant under a corner C_k if and only if the corresponding TS is larger than a threshold t_v. That is

$$y_{A_iC_k} = \begin{cases} 1, & \text{if } \mathrm{TS}_{A_iC_k} > t_v \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where t_v is set as 0.0001 in our experiments.

However, the computational cost of the TS evaluation flow increases linearly with the number of corners, making it excessively time-consuming to evaluate TS for all pins across all corners, particularly in large-scale training designs. To expedite the generation of training labels, we also incorporate NCF model and matrix completion into our training label generation flow, as depicted in Fig. 11(a). Our approach begins by performing insensitive pins filtering on each corner and design, which involves a one-time timing propagation and therefore does not consume significant time. Subsequently, only the pins that have not been filtered out in at least one of the corners are considered for the TS evaluation step. For each large training design, we randomly select a subset of corners and generate TS values for the remaining pins under those corners, while TS values of small training design are generated across all corners. Then, we convert TS values to timing variability and build the pin-corner matrix. By employing NCF model training and prediction, we can infer the timing variability for the unselected corners. Fig. 12(a) visualizes the matrix completion process. Timing variability for all pins from small training designs is already known, whereas for pins in large training designs, timing variability is only evaluated for a subset of corners. The goal is to infer the timing variability within the white part. Given that the known timing variability spans across all designs and corners, we refer to this approach as *intra-design/corner set matrix completion*.

E. Training Techniques—Data Augmentation With mixup

Now, intuitively, we can utilize the training features from Tables I and II and the training labels generated from Section IV-D to start the training of our NCF model. However, due to the nature of timing graphs, few pins are actually influential in the overall timing accuracy. Consequently, there exists a scarcity of positive elements (i.e., $y_{A_iC_k} = 1$) in the pin-corner matrix (less than 10% in most designs). The overwhelmingly large portion of timing invariant data makes it hard for the model to distinguish critical but rare timing variant pins from hundreds of thousands of pins in a large circuit design. To mitigate the label imbalance issue, we utilize *mixup* [22], a simple yet effective data-augmentation approach that generates pseudo-training data by interpolation.

We first group all pin-corner pairs with training label 1 as the positive set $\mathbb{X}^+ : \{(A_i, C_k)|(A_i, C_k) \in \mathbb{X}$ and $y_{A_iC_k} = 1\}$ and those with training label 0 as the negative set $\mathbb{X}^- : \{(A_j, C_l)|(A_j, C_l) \in \mathbb{X}$ and $y_{A_jC_l} = 0\}$. In each training iteration, for each pin-corner pair (A_i, C_k) in $\mathbb{X}^+$, we sample one pin-corner pair (A_j, C_l) from the negative set $\mathbb{X}^-$. Then, the corresponding pin and corner features form a feature tuple $(p_{A_i}, q_{C_k}, p_{A_j}, q_{C_l})$. After that, we interpolate each data pair with a coefficient λ sampled from the Beta distribution to generate the augmented data

$$\tilde{p}_{A_{ij}} = \lambda p_{A_i} + (1-\lambda)p_{A_j}$$

$$\tilde{q}_{C_{kl}} = \lambda q_{C_k} + (1-\lambda)q_{C_l}$$

$$\tilde{y}_{A_{ij}C_{kl}} = \begin{cases} 1, & \text{if } \lambda y_{A_iC_k} + (1-\lambda)y_{A_jC_l} > 0.5 \\ 0, & \text{otherwise.} \end{cases}$$

After generating the augmented data $(\tilde{p}_{A_{ij}}, \tilde{q}_{C_{kl}}, \tilde{y}_{A_{ij}C_{kl}})$, we can optimize the NCF model by rewriting the loss function (9) as follows:

$$\mathcal{L} = -\sum_{(A_i,C_k)\in\mathbb{X}^+} \left[\tilde{y}_{A_{ij}C_{kl}} \log \hat{y}_{A_{ij}C_{kl}} \right.$$
$$\left. +\left(1-\tilde{y}_{A_{ij}C_{kl}}\right) \log\left(1-\hat{y}_{A_{ij}C_{kl}}\right)\right]$$

where

$$\hat{y}_{A_{ij}C_{kl}} = f_\Phi\left(\left[\tilde{p}_{A_{ij}}, \tilde{q}_{C_{kl}}\right]\right).$$

As shown in Fig. 10, by interpolating the pin and corner features with opposite timing variability, the model is forced to learn more comprehensive and distinguishable features for timing variant pins and thus make more precise inferences on new layouts or unseen corners.

F. Testing Data Prediction

After completing the training label generation and data augmentation, we perform NCF model training and prediction to identify timing variant pins in the testing designs, as shown in Fig. 11(b). Unlike the intra-design/corner set matrix completion discussed in Section IV-D, where the timing variability of pins from all designs is known, in this case, the matrix completion process needs to infer timing variability for pins from completely unseen testing designs. Furthermore, we assume that there may exist unseen corners in the testing data. Therefore, we refer to this approach as *inter-design/corner set matrix completion*. Fig. 12(b) visualizes this matrix completion process.

Once we have identified the timing variant pins, we proceed to generate timing macro models for each testing design under each corner, employing a process similar to single-corner timing macro modeling, as depicted in Fig. 8. The distinction lies in the pin preservation step, which is now determined based on the pin-corner matrix.

TABLE III
TRAINING DESIGN STATISTICS. IN EXPERIMENTS REGARDING THE ACCELERATION OF TRAINING LABEL GENERATION IN SECTION V-C4, THE LEFT 13 DESIGNS SERVE AS "SMALL" TRAINING DESIGNS WHILE THE RIGHT SIX DESIGNS ARE "LARGE" TRAINING DESIGNS

Design	#Pins	#Cells	#Nets	Design	#Pins	#Cells	#Nets
s1196_eval	2181	823	791	usb_funct	48243	15992	15911
s1494_eval	2584	976	951	systemcaes	21971	6873	6790
s27_eval	94	40	36	wb_dma	13125	4627	4419
s344_eval	638	251	225	tv80	17038	5331	5317
s349_eval	651	261	235	ac97_ctrl	42438	14473	14435
s386_eval	641	245	232	pci_bridge32	59879	19426	19274
s400_eval	760	287	260				
s510_eval	980	382	369				
s526_eval	1002	381	354				
crc32d16N	1283	527	495				
usb_phy_ispd	2545	957	938				
systemcdes	10282	3638	3596				
aes_core	66751	23327	23199				

TABLE IV
TESTING DESIGNS STATISTICS

Design	#Pins	#Cells	#Nets
mgc_edit_dist_iccad_eval	581319	224113	224101
vga_lcd_iccad_eval	768050	286597	286498
leon3mp_iccad_eval	4167632	1534489	1534410
netcard_iccad_eval	4458141	1630171	1630161
leon2_iccad_eval	5179094	1892757	1892672
mgc_edit_dist_iccad	450354	164266	164254
vga_lcd_iccad	679258	259251	259152
leon3mp_iccad	3376832	1248058	1247979
netcard_iccad	3999174	1498565	1498555
leon2_iccad	4328255	1617069	1616984
mgc_matrix_mult_iccad	492568	176084	174484

V. EXPERIMENTAL RESULTS

A. Experimental Settings

In our framework, the training and prediction of both NCF models and GNN models are implemented in the Python3 programming language, while the insensitive pins filtering, TS evaluation, and timing macro model generation are implemented in the C++ programming language. The experiments are conducted on a Linux workstation with a 3.7-GHz CPU, 192 GB RAM, and an NVIDIA RTX 3090 GPU.

For the experiments regarding single-corner timing macro modeling, we adopt the benchmark suite provided by the TAU 2016 [9] and TAU 2017 [10] contests as listed in Tables III and IV. For the experiments regarding multi-corner timing macro modeling, we also adopt the circuit designs from these two contests. As for the multi-corner library, we adopt the Synopsys SAED 32/28nm Digital Standard Cell Library [23], as the TAU libraries do not support multi-corner functionality. We use 27 base characterization corners in our experiments, as listed in Table V. The first and second characters in a corner's name denote its nMOS and pMOS processes, respectively, (f, t, and s represent fast, typical, and slow, respectively). The central segment of a corner's name signifies the voltage, while the final segment denotes the temperature. For instance, "tt1p05vn40c" corresponds to typical nMOS and pMOS processes, 1.05 V, and −40°C. In each corner library, a total of 314 cells are characterized. We

TABLE V
CORNER LIST

ff0p85v125c	ff0p85v25c	ff0p85vn40c	ff0p95v125c	ff0p95v25c
ff0p95vn40c	ff1p16v125c	ff1p16v25c	ff1p16vn40c	ss0p7v125c
ss0p7v25c	ss0p7vn40c	ss0p75v125c	ss0p75v25c	ss0p75vn40c
ss0p95v125c	ss0p95v25c	ss0p95vn40c	tt0p78v125c	tt0p78v25c
tt0p78vn40c	tt0p85v125c	tt0p85v25c	tt0p85vn40c	tt1p05v125c
tt1p05v25c	tt1p05vn40c			

have aligned the cell names in TAU 2016 and TAU 2017 circuit netlists with those in the SAED libraries.

In our experiments, we assess timing accuracy by comparing the timing analysis results of the timing macro model with the flat circuit design. We adopt iTimerC 2.0 [14] as the reference timer. On the other hand, the evaluation of macro model size is based on the size of the early library associated with the timing macro model.

B. Results on Single-Corner Timing Macro Modeling

First, we evaluate our single-corner timing macro modeling framework and compare the results with state-of-the-art works. Note that in our experiments, we utilize only the first eight basic features in Table I, along with the *is_CPPR* feature. This selection is made to enhance the overall performance.

Table VI shows the results on TAU 2016 [9] and TAU 2017 [10] benchmarks considering CPPR and the comparisons with two state-of-the-art ILM-based works iTimerM [3] and [4]. Among all the criteria, max error and model file size are viewed as the most crucial ones. Our framework achieves extremely high-timing accuracy as all the max errors are less than 0.1 ps, which is same as iTimerM [3] and 9 times better than [4]. As for model file size, our result is about 10% smaller than iTimerM [3] and 45% smaller than [4]. To summarize, our framework preserves the highest-timing accuracy in terms of max errors among the state-of-the-art works, while further improving the model size by 10% than the same-accuracy-level work. Our framework also achieves similar or even better results in terms of model generation performance and model usage performance. The average errors of our framework are slightly higher than those of iTimerM [3]; however, the difference is only a few femtoseconds and thus can be neglected.

In Table VII, we demonstrate how domain knowledge can enhance GNN model training across various timing models or modes, using CPPR as a case study. As mentioned in Section II-E, multiple-fan-out pins of clock networks are crucial for CPPR calculation. Thus, we could add a dedicated training feature for CPPR to indicate this kind of pins, called *is_CPPR*. We adopt the results of iTimerM [3] as the baseline and calculate the differences and ratios as described in Table VI. Before adding *is_CPPR*, our framework could already achieve the same timing accuracy as iTimerM [3] while reducing the model size by 6%. After the *is_CPPR* feature is included, our framework still preserves the same timing accuracy while improving the model size by 10%. The result tells that our framework could achieve superior quality with only the basic features, while the dedicated features could capture the timing properties of designs more precisely.

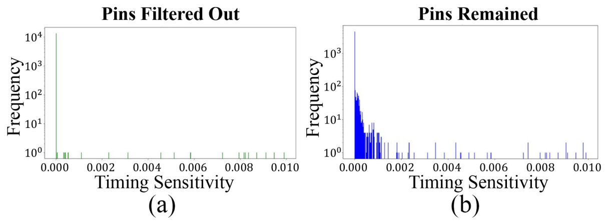


Fig. 13. Separated TS distribution of *systemcaes* based on the insensitive pins filtering.

Table VIII displays the results on the TAU 2017 [10] benchmark without CPPR. Our results are compared with the ILM-based work iTimerM [3] and the ETM-based work ATM [5]. In comparison with ATM [5], our framework achieves 9 times better-max error and 25 times better-average error, but it suffers from a larger model size. It is as our expectation since our framework is ILM-based while ATM [5] is ETM-based. Besides, we also achieve 17 times faster model generation runtime than ATM [5]. As for the ILM-based work iTimerM [3], we preserve the same timing accuracy while improving the model size by 9%. The result demonstrates the applicability and generality of our framework on different timing modes (CPPR on and CPPR off), and it may be further inferred to various timing delay models and modes.

As mentioned in Section II-D, the goal of the insensitive pins filtering is to exclude noncritical pins rapidly, under the premise that the timing accuracy is not degraded. Fig. 13 shows the timing sensitivities of pins in the training design *systemcaes*. TS of pins that are filtered out are shown in the left histogram, and those of the potential sensitive pins are shown in the right histogram. It can be seen that a majority of filtered pins indeed have zero TS, while many remained pins have nonzero TS. It confirms the consistency between the insensitive pins filtering and the TS evaluation, which implies the insensitive pins filtering is suitable for accelerating the training data generation flow. To further ensure the timing accuracy is not degraded by the insensitive pins filtering, we conduct an experiment in which the training labels of all the remained pins after the insensitive pins filtering are set to 1. The result is shown in Table IX. The results of iTimerM [3] are adopted as the baseline, and the differences and ratios are calculated as described in Table VI. The results achieve the same timing accuracy as iTimerM [3] which is of the best accuracy among the previous works. Therefore, it is supported that the insensitive pins filtering does not degrade the resulting timing accuracy.

Finally, when we encounter new benchmarks under the same NLDM libraries, we only need to consider the GNN model inference runtime and the model generation runtime since our framework is available on general designs under the NLDM. The GNN model inference time is usually much less than the model generation time listed in the above tables. Thus, our framework spends comparable or even shorter runtime than previous work for unseen test data under the NLDM. As for other timing delay models, such as AOCV, POCV, and CCS, we need to further consider the training data generation time and the GNN model training time. However, since our

TABLE VI
SINGLE-CORNER TIMING MACRO MODELING EXPERIMENTAL RESULTS ON TAU 2016 [9] AND TAU 2017 [10] BENCHMARKS WITH CPPR. FOR THE MODEL FILE SIZE, WE ADOPT THE SIZE OF THE LIBRARY FOR LATE TIMING. DIFFERENCE 1 AND RATIO 1 ARE COMPARED WITH iTIMERM [3]. DIFFERENCE 2 AND RATIO 2 ARE COMPARED WITH [4]. DIFFERENCE = COMPARED RESULT − OUR RESULT. RATIO = COMPAREDF RESULT/OUR RESULT. NOTE THAT [4] IS ONLY EVALUATED ON TAU 2016 BENCHMARK IN THEIR WORK

Design		Avg. Error (ps)	Max Error (ps)		Model File Size (MB)	Generation Runtime (s)	Generation Memory (MB)	Usage Runtime (s)	Usage Memory (MB)
mgc_edit_dist_iccad_eval	Ours	0.0000	0.007	Ours	56	11	1087	8	475
	iTimerM	0.0000	0.007	iTimerM	64	10	1043	8	550
	[4]	N.A.	0.158	[4]	79	15	5	4	5
vga_lcd_iccad_eval	Ours	0.0006	0.040	Ours	45	12	1204	6	383
	iTimerM	0.0006	0.040	iTimerM	50	13	1208	7	402
	[4]	N.A.	0.255	[4]	72	24	399	4	5
leon3mp_iccad_eval	Ours	0.0004	0.052	Ours	35	50	4908	5	324
	iTimerM	0.0004	0.052	iTimerM	45	58	4807	6	395
	[4]	N.A.	0.220	[4]	86	78	5	5	5
netcard_iccad_eval	Ours	0.0000	0.004	Ours	213	89	6609	29	1757
	iTimerM	0.0000	0.004	iTimerM	220	65	6513	29	1822
	[4]	N.A.	0.203	[4]	372	101	12616	23	4332
leon2_iccad_eval	Ours	0.0002	0.016	Ours	369	89	8298	64	3034
	iTimerM	0.0002	0.016	iTimerM	372	82	7865	61	3056
	[4]	N.A.	0.241	[4]	676	105	15299	38	5315
TAU 2016 Average	Difference 1	0.0000	0.000	Ratio 1	1.116	0.961	0.975	1.099	1.094
	Difference 2	N.A.	0.192	Ratio 2	1.809	1.448	0.818	0.738	0.851
mgc_edit_dist_iccad	Ours	0.0029	0.052	Ours	60	16	1054	8	514
	iTimerM	0.0003	0.052	iTimerM	66	12	1063	9	537
vga_lcd_iccad	Ours	0.0024	0.080	Ours	56	16	1455	7	474
	iTimerM	0.0023	0.080	iTimerM	58	15	1429	8	487
leon3mp_iccad	Ours	0.0031	0.046	Ours	37	68	5407	5	332
	iTimerM	0.0016	0.046	iTimerM	46	67	5281	6	406
netcard_iccad	Ours	0.0013	0.029	Ours	239	101	7814	35	1938
	iTimerM	0.0003	0.029	iTimerM	248	98	7545	33	1993
leon2_iccad	Ours	0.0027	0.095	Ours	438	125	8171	62	3613
	iTimerM	0.0013	0.095	iTimerM	440	109	8049	64	3625
TAU 2017 Average	Difference	-0.0013	0.000	Ratio	1.084	0.903	0.984	1.070	1.065

TABLE VII
SINGLE-CORNER TIMING MACRO MODELING EXPERIMENTAL RESULTS WITH AND WITHOUT CPPR-DEDICATED FEATURES

Benchmark		Avg. Error	Max Error		Model File Size	Generation Runtime	Generation Memory	Usage Runtime	Usage Memory
TAU2016 (avg.)	Difference Before	0.0000	0.000	Ratio Before	1.064	1.055	0.959	1.133	1.048
	Difference After	0.0000	0.000	Ratio After	1.116	0.961	0.975	1.099	1.094
TAU2017 (avg.)	Difference Before	-0.0001	0.000	Ratio Before	1.060	0.828	0.994	1.115	1.037
	Difference After	-0.0013	0.000	Ratio After	1.084	0.903	0.984	1.070	1.065

framework could be directly applied to perform timing macro modeling no matter which timing model is chosen, users do not need to spend a great deal of time designing specific algorithms for different timing delay models and tuning a bunch of parameters. As a consequence, our framework still shows high applicability and efficiency on the timing macro modeling problem.

C. Results on Multi-Corner Timing Macro Modeling

1) Effectiveness of Our Multi-Corner Framework: In this section, we compare our multi-corner timing macro modeling framework with the multi-corner extensions of state-of-the-art single-corner timing macro modeling frameworks. The results of iTimerM [3] are obtained by directly conducting the iTimerM algorithm flow on each specific corner. For the experiments of our single-corner timing macro modeling framework (hereafter referred to as "[6]" to avoid ambiguity in this section), training data for each corner is generated first, and subsequently distinct GNN models are trained for each individual corner.

To ensure fair comparisons, for experiments in Sections V-C1 and V-C2, all the training designs from Table III are categorized as "small" during the training label generation process (as shown in Fig. 11); that is, the timing variability of all the pins is evaluated under all the corners. Note that LibAbs [2], [4] and ATM [5] do not support the multi-corner SAED library, and thus they are not included in the comparison in this section.

Table X presents the experimental results on TAU 2016 and TAU 2017 benchmarks listed in Table IV over the 27 corners. Here, the "average" max error (*resp.* model size) represents the mean values of the maximum timing errors (*resp.* timing macro model size) across the 27 corners, while the "max" max error (*resp.* model size) indicates the highest values of the maximum timing errors (*resp.* timing macro model size) across the 27 corners. Note that the results in Table X are without CPPR because the SAED library does not provide separate early and late cell libraries. Nevertheless, incorporating CPPR into our framework would pose no significant challenge, as we can easily integrate the *is_CPPR* feature in our pin feature vector.

Compared to the state-of-the-art algorithmic work iTimerM [3], our framework demonstrates a 16% improvement in model size while maintaining the same level of timing accuracy, with an average max error difference of less than

TABLE VIII
SINGLE-CORNER TIMING MACRO MODELING EXPERIMENTAL RESULTS ON TAU 2017 BENCHMARK WITHOUT CPPR. DIFFERENCE 1 AND RATIO 1 ARE COMPARED WITH ITIMERM [3]. DIFFERENCE 2 AND RATIO 2 ARE COMPARED WITH ATM [5]. DIFFERENCE = COMPARED RESULT − OUR RESULT. RATIO = COMPARED RESULT/OUR RESULT. WE ADDITIONALLY INCLUDE THE CIRCUIT *mgc_matrix_mult_iccad* TO EVALUATE SINCE ATM [5] ALSO ADOPTS IT AS ONE TEST CASE

Design		Avg. Error (ps)	Max Error (ps)		Model File Size (MB)	Generation Runtime (s)	Generation Memory (MB)	Usage Runtime (s)	Usage Memory (MB)
mgc_edit_dist_iccad	Ours	0.0033	0.052	Ours	59	14	1069	9	563
	iTimerM	0.0007	0.052	iTimerM	65	13	1062	9	523
	ATM	0.0960	0.402	ATM	2	833	N.A.	0.36	N.A.
vga_lcd_iccad	Ours	0.0026	0.080	Ours	52	18	1457	7	442
	iTimerM	0.0023	0.080	iTimerM	55	17	1420	9	450
	ATM	0.0400	0.160	ATM	0.3	85	N.A.	0.06	N.A.
leon3mp_iccad	Ours	0.0033	0.046	Ours	31	78	5392	5	275
	iTimerM	0.0018	0.046	iTimerM	31	102	5257	4	286
	ATM	0.1070	0.460	ATM	0.6	740	N.A.	0.09	N.A.
netcard_iccad	Ours	0.0033	0.029	Ours	226	124	7804	32	1795
	iTimerM	0.0005	0.029	iTimerM	229	104	7539	33	1838
	ATM	0.0540	0.246	ATM	1.6	618	N.A.	0.27	N.A.
leon2_iccad	Ours	0.0027	0.095	Ours	408	193	8156	60	3378
	iTimerM	0.0013	0.095	iTimerM	410	152	7782	59	3390
	ATM	0.0400	0.240	ATM	2.4	1055	N.A.	0.34	N.A.
mgc_matrix_mult_iccad	Ours	0.0032	0.054	Ours	124	27	1106	18	924
	iTimerM	0.0020	0.054	iTimerM	171	29	1114	24	1098
	ATM	0.1300	0.450	ATM	12	629	N.A.	1.63	N.A.
Average	Difference 1	-0.0016	0.000	Ratio 1	1.093	0.980	0.978	1.085	1.033
	Difference 2	0.0748	0.267	Ratio 2	0.028	17.910	N.A.	0.029	N.A.

TABLE IX
VALIDATION ON INSENSITIVE PINS FILTERING

Benchmark	Avg. Error	Max Error	Model File Size
TAU2016	0.0000	0.000	1.040
TAU2017	0.0000	0.000	1.009

0.2 ps. Moreover, considering the maximum of max error and model size, our framework reduces the timing macro model size by over 20% while maintaining a timing error difference of less than 0.1 ps. Compared to the learning-based approach [6], our framework exhibits a similar trend, showcasing a 10% improvement in average model size while keeping the timing error difference below 0.2 ps. Similarly, the results are even better when considering the maximum of max error and model size. The experimental results validate the precise identification of timing variant pins in each corner by our framework, leading to highly compact timing macro models. Furthermore, the superior performance observed in terms of the maximum of max error and model size suggests that our framework effectively captures the relationship between each pin and each specific corner, ensuring stable performance across all corners.

In addition to timing accuracy and timing macro model size, our framework also exhibits outstanding performance in terms of runtime. As shown in Table XI(a), our framework achieves approximately a 16X faster model training time compared to [6]. The main reason is that the GNN models used in our single-corner framework [6] are specific to each corner. Consequently, experiments of [6] necessitated training 27 separate GNN models. In contrast, our NCF model in our multi-corner framework accounts for all the designs and corners involved and thus requires only one model that merely takes 4.12 s per training epoch. It is important to note that the time spent on model tuning has not been considered here, and it is obvious that fine-tuning the parameters for 27 models is much more time-consuming than for a single model. The speedups will be even more significant for advanced technology nodes with hundreds or even thousands of corners. Furthermore, Table XI(b) compares the time required to identify and generate a file containing the timing variant pins for the 11 designs listed in Table IV for one corner. All three frameworks exhibit similar performance in this aspect. Additionally, it is worth noting that our framework is able to identify timing variant pins for unseen designs. Therefore, it is reasonable to compare the model inference time of our framework with the overall runtime of iTimerM [3].

To further analyze the results of multi-corner timing macro modeling, we present 3-D surface plots that illustrate variations in maximum errors and model sizes for various corners and designs, as depicted in Fig. 14. These visualizations are based on the experimental results presented in Tables X and XII. Each of the surface plots contains 297 data points (27 corners × 11 designs). In comparison with model sizes, the max errors show more drastic changes across various corners. This phenomenon aligns with the inherent characteristics of timing macro models, wherein the mere addition or removal of a small number of timing-critical pins can induce significant alterations in timing behavior. Nevertheless, our multi-corner framework maintains a max error of less than 2.0 ps for all the corners and designs, which is comparable to the existing works. Additionally, we can observe from Fig. 14(f) that the model sizes of our single-corner framework [6] fluctuate wildly across different corners. It is as expected given that this approach necessitates a unique GNN model for each corner. Consequently, this model-by-model strategy may result in unstable performance due to the uncertainties inherent in the training processes for each GNN instance. In contrast, as shown in Fig. 14(g),

TABLE X
MULTI-CORNER TIMING MACRO MODELING EXPERIMENTAL RESULTS ON TAU 2016 AND TAU 2017 BENCHMARKS OVER 27 CORNERS. AVERAGE ERROR AND MAX ERROR ARE COMPARED BY DIFFERENCE, WHERE DIFFERENCE = COMPARED RESULT − OUR RESULT. MACRO MODEL SIZE IS COMPARED BASED ON RATIO, WHERE RATIO = COMPARED RESULT/OUR RESULT

Design	iTimerM [3]					Our Single-Corner [6]					Our Multi-Corner				
	Average Error (ps)	Max Error (ps)		Model Size (MB)		Average Error (ps)	Max Error (ps)		Model Size (MB)		Average Error (ps)	Max Error (ps)		Model Size (MB)	
		average	max	average	max		average	max	average	max		average	max	average	max
mgc_edit_dist_iccad_eval	0.0005	0.217	1.293	66.7	69.0	0.0008	0.183	0.613	87.4	101.0	0.0188	0.702	1.855	54.1	55.0
vga_lcd_iccad_eval	0.0005	0.064	0.309	65.4	78.0	0.0017	0.284	0.878	65.2	68.0	0.0005	0.067	0.309	80.2	96.0
leon3mp_iccad_eval	0.0004	0.118	1.031	47.0	47.0	0.0004	0.133	1.031	31.8	33.0	0.0058	0.491	1.031	36.0	36.0
netcard_iccad_eval	0.0005	0.188	1.838	219.7	220.0	0.0008	0.219	1.838	198.3	225.0	0.0021	0.352	1.838	203.9	205.0
leon2_iccad_eval	0.0004	0.146	1.771	375.0	376.0	0.0008	0.183	1.771	345.5	360.0	0.0046	0.309	0.599	371.6	373.0
mgc_edit_dist_iccad	0.0002	0.006	0.008	68.0	73.0	0.0003	0.017	0.072	74.9	86.0	0.0096	0.185	0.343	60.0	60.0
vga_lcd_iccad	0.0004	0.042	0.225	88.9	101.0	0.0016	0.232	0.969	71.9	77.0	0.0006	0.096	0.225	90.3	112.0
leon3mp_iccad	0.0003	0.078	0.132	46.0	47.0	0.0003	0.078	0.132	33.5	36.0	0.0013	0.132	0.249	36.3	37.0
netcard_iccad	0.0002	0.028	0.044	247.2	248.0	0.0003	0.031	0.062	246.8	258.0	0.0014	0.124	0.226	232.3	237.0
leon2_iccad	0.0003	0.085	0.112	435.5	436.0	0.0003	0.102	0.146	412.7	462.0	0.0010	0.141	0.240	432.9	433.0
mgc_matrix_mult_iccad	0.0003	0.034	0.132	231.4	278.0	0.0005	0.127	0.264	229.7	259.0	0.0074	0.516	0.871	118.4	119.0
Diffrence / Ratio	**-0.0045**	**-0.192**	**-0.081**	**1.168**	**1.203**	**-0.0041**	**-0.139**	**-0.001**	**1.103**	**1.177**	**0.0000**	**0.000**	**0.000**	**1.000**	**1.000**

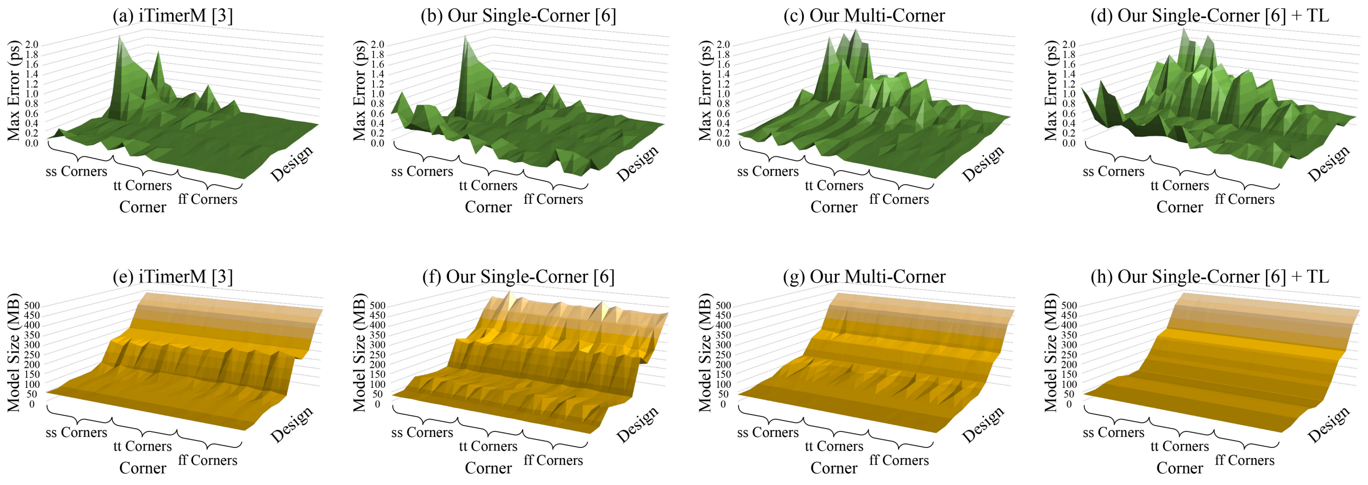


Fig. 14. 3-D surface plots of max errors and model sizes across corners and designs.

our multi-corner framework consistently demonstrates similar model sizes across diverse corners, indicating its adaptability to various, and potentially unseen, timing corners.

2) Exploration of Transfer Learning-Based Approach: As described in Section V-B, our single-corner framework [6] achieves timing accuracy comparable to the state-of-the-art works while reducing model sizes by 10%. Therefore, to enhance the performance of multi-corner timing macro modeling, an intuitive and reasonable approach is to leverage transfer learning (TL) [24] and transfer our single-corner results to diverse corners. Specifically, starting from a source corner (we select "tt0p78v25c" in our experiments), we implement our GNN-based single-corner framework for 20 000 epochs on this source corner. Subsequently, we fine-tune the generated timing macro model across the remaining 26 corners (listed in Table V) for 5000 epochs. The results are listed in Table XII, where the differences and ratios are compared against our multi-corner framework.

Compared to the results in Table X, the TL-based approach achieves the most compact macro model sizes for most designs. However, it suffers from a significant degradation in timing accuracy. As illustrated in Fig. 14(h), we can observe the TL-based timing macro models exhibit strikingly uniform model sizes across various corners. We hypothesize that this uniformity arises due to the heavy reliance of the transferred timing macro models on the source timing macro model. Consequently, these transferred models fail to discern and retain pins that are uniquely sensitive to the specific corner. This deficiency leads to significant accuracy fluctuations across corners, as illustrated in Fig. 14(d). We further observe that the transferred timing macro models tend to consider only clock pins during TS evaluation while ignoring all other pins, resulting in extremely small model sizes but with high-timing errors.

To implement the TL-based approach, we still need to generate training data for every corner, as described in Section IV-D, and fine-tune for each target corner. In contrast, our multi-corner framework can produce timing macro models that are accurate and compact for various corners, even those we have not seen before, without any additional fine-tuning. Therefore, our multi-corner framework is more efficient and generic than the TL-based approach.

3) Results on Unseen Corners: As discussed in Section III-A, a challenging situation occurs when dealing with chiplets that deviate from typical corners. To validate the applicability of our framework in such scenarios, we hide 3 corners, "ff0p95v125c," "ss0p75vn40c," and "tt0p85v125c," from our training data. Then, we train our NCF model based on the 24 known corners and identify timing variant pins in testing designs under both the seen and unseen corners.

TABLE XI
MODEL INFERENCE RUNTIME FOR THE 11 TESTING DESIGNS FOR ONE CORNER

(a) Model training runtime for all the corners.

	Our Single-Corner [6]	Our Multi-Corner
Total Training Runtime	80 minutes	5 minutes

(b) Model inference runtime for the 11 testing designs for one corner.

	iTimerM [3]	Our Single-Corner [6]	Our Multi-Corner
Inference Runtime	7 minutes	6 minutes	7 minutes

TABLE XII
ABLATION STUDY ON TRANSFERRING OUR SINGLE-CORNER FRAMEWORK

Design	Our Single-Corner [6] + Transfer Learning				
	Average Error (ps)	Max Error (ps)		Model Size (MB)	
		average	max	average	max
mgc_edit_dist_iccad_eval	0.0188	0.702	1.855	54.0	55.0
vga_lcd_iccad_eval	0.0053	0.558	1.361	43.0	43.0
leon3mp_iccad_eval	0.0061	0.491	1.031	36.0	36.0
netcard_iccad_eval	0.0095	0.825	1.396	213.0	214.0
leon2_iccad_eval	0.0062	0.606	1.246	371.2	372.0
mgc_edit_dist_iccad	0.0096	0.185	0.343	60.0	60.0
vga_lcd_iccad	0.0039	0.290	0.541	51.0	51.0
leon3mp_iccad	0.0039	0.163	0.249	36.0	36.0
netcard_iccad	0.0062	0.268	0.369	241.0	241.0
leon2_iccad	0.0015	0.202	0.271	432.0	432.0
mgc_matrix_mult_iccad	0.0073	0.516	0.871	119.4	120.0
Diffrence / Ratio	**0.0023**	**0.154**	**0.159**	**0.925**	**0.904**

Table XIII shows the evaluation results across all the corners, where the values are averaged across the 11 testing designs in Table IV. It can be observed that the performance on the three unseen corners is comparable to that of the 24 seen corners. Moreover, the max error of the corner "ff0p95v125c" (0.0744 ps) falls in a similar range as the max errors of other fast/fast process corners (between 0.0595 ps and 0.0826 ps), and similar trends can also be observed on unseen corners "ss0p75vn40c" and "tt0p85v125c." The results indicate that our framework is able to be generalized to arbitrary corners and demonstrates the potential for use in off-corner chiplets.

4) Acceleration of Training Label Generation: As discussed in Section IV-D, the training label generation on large designs takes an extremely long runtime to complete and therefore necessitates using the NCF to speed up the training label generation process. To validate the effectiveness of the NCF-based training label generation flow, we set the left 13 designs in Table III as the "small" training designs and the right 6 designs as the "large" training designs. For each small training design, the TS and the timing variability of all pins under all corners are evaluated, while those of large training designs are evaluated under only 6 corners, ff0p85vn40c, ff1p16v125c, ss0p75v25c, ss0p95v125c, tt0p85v125c, and tt1p05v25c. Table XIV shows that the NCF model realizes a 4.4X speedup. It is worth noting that within the 16.8 h runtime after acceleration, the NCF model training and matrix completion only takes about 10 min, while most of the time is spent on the TS evaluation for the six corners. Thus, the acceleration will become more significant if the total number of corners increases. To further testify the quality of the NCF-generated labels, we compare the predicted labels with the golden labels (obtained by performing TS evaluation on the remaining 21 corners). On average, the NCF model can achieve almost 95% prediction accuracy. Thus, the accelerated training label generation flow can significantly reduce label generation time while maintaining the accuracy of the generated labels.

TABLE XIII
EVALUATION RESULTS ON UNSEEN CORNERS. THE TRAINING DATA CONSISTS SOLELY OF THE UPPER 24 CORNERS, WITH THE BOTTOM THREE CORNERS BEING ENCOUNTERED FOR THE FIRST TIME DURING THE EVALUATION OF THE TESTING DESIGNS

Corner	Model Size (MB)	Avg. Error (ps)	Max Error (ps)
ff0p85v125c	166.3636	0.0018	0.0769
ff0p85v25c	164.1818	0.0020	0.0764
ff0p85vn40c	164.0000	0.0022	0.0826
ff0p95v25c	162.4545	0.0017	0.0693
ff0p95vn40c	159.8182	0.0018	0.0698
ff1p16v125c	166.2727	0.0017	0.0751
ff1p16v25c	161.4545	0.0015	0.0638
ff1p16vn40c	160.6364	0.0014	0.0595
ss0p75v125c	171.7273	0.0042	0.3410
ss0p75v25c	168.6364	0.0044	0.3107
ss0p7v125c	173.7273	0.0045	0.5170
ss0p7v25c	167.5455	0.0045	0.3058
ss0p7vn40c	161.4545	0.0044	0.5778
ss0p95v125c	160.0909	0.0033	0.1218
ss0p95v25c	162.2727	0.0040	0.1500
ss0p95vn40c	162.7273	0.0043	0.1891
tt0p78v125c	165.1818	0.0032	0.1179
tt0p78v25c	171.4545	0.0041	0.1621
tt0p78vn40c	172.4545	0.0044	0.2278
tt0p85v25c	166.2727	0.0036	0.1346
tt0p85vn40c	169.0909	0.0041	0.1606
tt1p05v125c	157.5455	0.0023	0.1015
tt1p05v25c	158.4545	0.0023	0.0919
tt1p05vn40c	158.6364	0.0026	0.0978
ff0p95v125c	166.4545	0.0017	0.0744
ss0p75vn40c	162.4545	0.0046	0.2858
tt0p85v125c	162.0909	0.0028	0.1015

TABLE XIV
ACCELERATION OF TRAINING LABEL GENERATION

	Performance
Runtime Speedup	4.4X (= 74.7 hours / 16.8 hours)
Correctness Ratio	94.96%

VI. CONCLUSION

In this work, we present a novel learning-based single corner timing macro modeling framework that is applicable to various timing analysis models and modes. By leveraging our proposed TS metric and drawing analogies between GNN and timing macro modeling, our approach achieves exceptionally high-timing accuracy while further improving the model size than the most accurate state-of-the-art work. Furthermore, we address and formulate the multi-corner timing macro modeling problem. We view the problem from the perspective of recommendation systems and transform it into a matrix completion task. We introduce the concept of collaborative filtering and propose an NCF-based framework to capture the complex interactions between pins and corners. Through our framework, high-quality timing macro models are generated for each corner. Experimental results based on Synopsys SAED multi-corner libraries [23] and TAU 2016 [9] and TAU 2017 [10] benchmarks show our framework preserves extremely high-timing accuracy while reducing more than 10% of model sizes compared to state-of-the-art works. Moreover, our framework achieves a 16X faster model training time and accelerates the training label generation process by 4.4X. Additionally, based on the evaluation of our prediction results on unseen corners, the applicability of our framework for off-corner chiplets is also validated. Future work includes timing macro modeling for multi-corner multi-mode (MCMM), the development of unified timing macro models for multiple corners, and timing macro modeling for multiple-chiplet integration or subsystems.

References

[1] A. J. Daga, L. Mize, S. Sripada, C. Wolff, and Q. Wu, "Automated timing model generation," in *Proc. Design Autom. Conf. (DAC)*, 2002, pp. 146–151.

[2] T.-Y. Lai, T.-W. Huang, and M. D. F. Wong, "LibAbs: An efficient and accurate timing macro-modeling algorithm for large hierarchical designs," in *Proc. 54th Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[3] P.-Y. Lee and I. H.-R. Jiang, "iTimerM: A compact and accurate timing macro model for efficient hierarchical timing analysis," *ACM Trans. Design Autom. Electron. Syst. (TODAES)*, vol. 23, no. 4, pp. 1–21, 2018.

[4] T.-Y. Lai and M. D. F. Wong, "A highly compressed timing macro-modeling algorithm for hierarchical and incremental timing analysis," in *Proc. Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2018, pp. 166–171.

[5] K.-M. Lai, T.-W. Huang, P.-Y. Lee, and T.-Y. Ho, "ATM: A high accuracy extracted timing model for hierarchical timing analysis," in *Proc. 26th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2021, pp. 278–283.

[6] K. K.-C. Chang, C.-Y. Chiang, P.-Y. Lee, and I. H.-R. Jiang, "Timing macro modeling with graph neural networks," in *Proc. 59th Design Autom. Conf. (DAC)*, 2022, pp. 1219–1224.

[7] M. Andersch et al. "NVIDIA hopper architecture in-depth." 2022. [Online]. Available: https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

[8] B. Bailey. "Taming corner explosion in complex chips." 2023. https://semiengineering.com/taming-corner-explosion-in-complex-chips/

[9] J. Hu, S. Chen, X. Zhao, and X. Chen, "TAU 2016 timing contest on macro modeling," 2016. [Online]. Available: https://sites.google.com/site/taucontest2016/

[10] S. Chen and A. Khandelwal, "TAU 2017 timing contest on macro modeling," 2017. [Online]. Available: https://sites.google.com/site/taucontest2017/

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[12] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1025–1035.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.

[14] P.-Y. Lee, I. H.-R. Jiang et al., "iTimerC 2.0: Fast incremental timing and CPPR analysis," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2015, pp. 890–894.

[15] S. Onaissi, F. Taraporevala, J. Liu, and F. Najm, "A fast approach for static timing analysis covering all PVT corners," in *Proc. 48th Design Autom. Conf. (DAC)*, 2011, pp. 777–782.

[16] A. B. Kahng, U. Mallappa, L. Saul, and S. Tong, ""Unobserved corner" prediction: Reducing timing analysis effort for faster design convergence in advanced-node design," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, 2019, pp. 168–173.

[17] X. Jiao, D. Ma, W. Chang, and Y. Jiang, "TEVoT: Timing error modeling of functional units under dynamic voltage and temperature variations," in *Proc. 57th Design Autom. Conf. (DAC)*, 2020, pp. 1–6.

[18] W. Fu et al., "A cross-layer power and timing evaluation method for wide voltage scaling," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–6.

[19] S. Bian, M. Hiromoto, M. Shintani, and T. Sato, "LSTA: Learning-based static timing analysis for high-dimensional correlated on-chip variations," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, 2017, pp. 173–182.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2018, *arXiv:1710.09412*.

[23] (Synopsys Inc., Sunnyvale, CA, USA). *SAED_EDK32/28_CORE—SAED 32/28nm Digital Standard Cell Library*. (2012). [Online]. Available: https://solvnetplus.synopsys.com

[24] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

Kevin Kai-Chun Chang received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2022. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA.

His research interests lie in the design and verification of cyber–physical systems, autonomous vehicles, and electronic design automation.

Guan-Ting Liu received the B.S. degree in electronics engineering with cross-discipline in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2018, and the Ph.D. degree from the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan, in 2024.

He is currently a Research Scientist with NVIDIA Research Taiwan, Taipei, Taiwan. His research interests include machine learning, reinforcement learning, and electronic design automation.

Chun-Yao Chiang received the B.S. degree in electrical engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2019, and the M.S. degree in electronic design automation from the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, in 2021.

He is currently a Software Research and Development Engineer with Synopsys Inc., Taipei, Taiwan. His current research interests include physical design, timing analysis, and IR drop analysis.

Pei-Yu Lee received the Ph.D. degree in electronics engineering from the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, in 2018.

He is currently a Staff Engineer with Synopsys Inc., Hsinchu, Taiwan, and his research interests include physical design, timing analysis, distributed computing, and machine-learning for EDA.

Dr. Lee received Four First Place Awards from TAU Timing Contests and the 2017 TSIA Doctoral Graduate Student Award. He has served on the Program Committees for ASP-DAC and ISPD, and the Contest Chair for CADathlon at ICCAD.

Iris Hui-Ru Jiang received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan.

She is currently a Professor with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. Her current research interests include timing analysis and optimization and design for manufacturability.

Dr. Jiang has served as the General Chair for ISPD, the Program Chair for ISPD and ASP-DAC, and the Associate Editor for IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.