One-for-All: An Unified Learning-based Framework for Efficient Cross-Corner Timing Signoff

Linyu Zhu
linyuzhu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yichen Cai
cai_yichen@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Xinfei Guo*
xinfei.guo@sjtu.edu.cn
Shanghai Jiao Tong University &
State Key Laboratory of Integrated Chips and Systems (SKLICS)
Shanghai, China

ABSTRACT

In advanced technology nodes, the proliferation of process corners poses significant challenges in timing signoff, particularly in estimating wire-induced interconnect delay across process corners. This paper proposes a learning-based framework to perform cross-corner timing prediction efficiently and accurately. It seamlessly integrates learning-based reference corner selection and topology-aware interconnect timing prediction into the broader timing signoff steps and Engineering Change Order (ECO) processes. Unlike previous methods, it only requires information about one single known corner while accurately predicting all unknown corners. Evaluated on two mainstream industry processes, the proposed framework surpasses alternative machine learning models and existing strategies, with an impressive average accuracy enhancement of 62.6% and 95.3% respectively, and maintains a low mean absolute error (MAE) under 0.37ps and 0.01ps. Additionally, a faster version of the framework is developed to predict interconnect delay directly from a single extracted SPEF, yielding over 2× speedup. Moreover, the single-corner approach featured by the framework significantly accelerates ECO processes by over 10× compared to standard timing signoff flows. The proposed framework is also set to be open-sourced at a later date.

CCS CONCEPTS

• **Hardware → Static timing analysis**; **Electronic design automation**.

1 INTRODUCTION

The intricacy and difficulty of timing signoff have surged in advanced technology nodes [8, 9]. This complexity is largely a consequence of the exploded number of corners and scenarios [6, 11]. Traditionally, operational space could be effectively constrained by examining timing at a select few design points, such as worst-case conditions and best-case conditions for process, voltage, and temperature (PVT) variations. However, in advanced technology nodes, the growing significance of wire delay in timing paths, combined with the diminishing routing pitch and the impact of coupling capacitance, has heightened the influence of process variations in metallization on overall timing. The RC extraction corners consist of worst C, worst RC, best C, best RC, and typical, which worsen the whole corner explosion problem. It is time-consuming to check timing for over hundreds of corners or scenarios now given very limited computation resources. Additionally, when timing failures occur in specific corners or scenarios, designers must address these cross-corner or cross-mode violations through labor-intensive ECO iterations before the tape-out phase. Each iteration typically involves a distinct extraction run for each RC corner and a separate analysis run for each corner/mode combination in static timing analysis (STA). In complex designs, the proliferation of hundreds or thousands of ECOs to address various design issues leads to numerous iterations for extraction and timing analysis.

In modern timing signoff tools, interconnect delay estimation from extracted parasitic usually consumes a huge amount of runtime as it relies on analytical approaches or intricate solvers [1, 3]. Consequently, it is highly desired to boost the efficiency for estimating interconnect delay while ensuring reliable final signoff to guide subsequent ECO processes. By harnessing the power of machine learning (ML) and artificial intelligence (AI) [7], various techniques have been proposed to enhance the efficiency of net delay estimation, generally falling into two main categories. The first category primarily focuses on enhancing prediction accuracy and efficiency under a single RC corner. In [3], an ML-based wire delay estimator was introduced, considering the intricate net structure from the extracted RC network as part of the features for the XGBoost model. Another approach proposed a machine learning model for swift timing prediction, enabling the rapid generation of ECOs to rectify timing violations [18]. Several works have employed graph neural networks (GNN) to address wire delay estimation challenges during signoff. In works [19, 20], a fast and accurate wire timing estimator was proposed based on a novel graph learning architecture capable of generating wire path representations by aggregating local structure information and global relationships across entire RC nets. [22] further proposed a customized GNN model to streamline the tedious feature extraction process and achieve ultra-fast net delay estimation in signoff. While these studies have achieved promising results in predicting timing for a single corner, they do not specifically address the challenges associated with cross-corner timing prediction.

*Corresponding author.

ICCAD '24, October 27–31, 2024, New York, NY, USA

ACM ISBN 979-8-4007-1077-3/24/10…$15.00
https://doi.org/10.1145/3676536.3676656

The second category of prior work has focused on cross-corner timing estimation, utilizing observed RC corners to predict unknown corners and further improve signoff efficiency. In [14], a novel method was proposed for fast and accurate hold timing analysis covering all corners. The objective was to determine the worst-case hold slack at all corners using only a small set of full STA runs. [4] introduced a branch-and-bound method to identify the worst-case process corner for chip sign-off timing analysis. Similarly, [13] derived worst-case timing results much faster than a normal multi-corner STA by estimating the worst-case timing among various process corners. This allowed for evaluating the upper bound of critical path timing with a one-pass analysis only. However, finding a strictly tight upper bound in advanced technology nodes remains a challenge. [10] investigated a data-driven approach based on multivariate linear regression to predict timing analysis at unobserved corners using results from observed corners. The approach employed a backward stepwise selection strategy to determine which corners to observe and which to predict. However, this approach required multiple observed corners for relatively good coverage, making it time-consuming and lacking universality across designs. In [21], under a similar assumption, authors introduced nonlinear regression models for STA prediction with improved accuracy. They proposed using a statistical analysis tool to select key STA results as model inputs to save additional STA runs. While accurate, this approach was less feasible for numerous RC corners. In [17], a novel nonlinear machine learning-based timing prediction model was proposed, considering all iterative STA optimizations in a unique space and leveraging their close correlations. An interesting observation from this work was that path delays at different corners are strongly correlated, with correlations dependent on the topology and structure of the path. However, this work focused on coarse-grained path features and did not explore net structures or RC corners.

In summary, prior cross-corner timing prediction work typically required multiple corners as "known" or "observed" corners to estimate timing for the remaining "unknown" corners. This approach incurred high runtime overhead by invoking signoff tools multiple times to generate reference data. Additionally, previous efforts primarily focused on enhancing model accuracy and efficiency, often neglecting challenges in integrating these prediction models into a standard STA flow. In contrast, our work introduces an unified learning-based framework for cross-corner timing signoff, featuring a streamlined "one-for-all" approach that utilizes only one known corner to predict interconnect timing for the rest. The main contributions of this work are summarized as follows:

- The proposed framework incorporates two distinct single-corner-based interconnect delay estimation approaches: one emphasizes precision, while the other delivers rapid results. This duality allows for tailored accuracy and efficiency trade-offs to suit varying design scenarios.
- To intelligently account for the unique characteristics of different designs, the framework leverages an innovative learning-based approach to select the known corner and operates effectively on a limited dataset.
- Designed for seamless integration with signoff and timing ECO steps, the framework is demonstrated across two industry-standard technology nodes, showing improved accuracy, efficiency, and integration compared to state-of-the-art methods.

Table 1: Notations and Definitions

Notations	Definitions
N	Total number of RC corners
M	Total number of nets
c_n	n^{th} RC corner
$\mathcal{I}$	observed corners set
$\hat{\mathcal{I}}$	unobserved corners set
Matrix $\mathbf{Y}(z) = \mathbf{y}_{i,j}(z)$	Timing of the j-th net under the i-th RC corner for circuit z
Matrix $\mathbf{P}(z) = \mathbf{p}_{i,j}(z)$	Parasitic parameter of the j-th net under the i-th RC corner for circuit z
GM	Graph mapping function
Matrix $\mathbf{P}^g(z)$	Output graph matrix by GM
GF	Graph filtering function
Matrix $\mathbf{P}^h(z)$	Output matrix by GF

2 ONE-FOR-ALL CROSS-CORNER TIMING SIGN-OFF FRAMEWORK

The proposed framework, depicted in Fig. 1, consists of two key components (highlighted in purple): a learning-based strategy for selecting the optimal RC corner as a reference, and a model for predicting cross-corner interconnect timing from the selected corner. Initially, an RC network is extracted into a parasitic Standard Parasitic Extraction Format (SPEF) file from the selected corner, followed by a standard STA check for known timing information. A faster alternative skips STA, using the SPEF to estimate timing swiftly. Utilizing the predicted timing for interconnect across all RC corners, the framework estimates cell delay for every PVT condition. As the design matures, this framework accommodates various ECOs, enhancing the adaptability of the design flow. Unlike traditional signoff processes that require extensive parasitic extraction and timing analysis across all RC corners, this framework reduces the runtime and computing resource demand, speeding up the ECO iteration. In this section, we will begin with the cross-corner timing prediction, followed by details on the RC corner selection and how the signoff framework integrates with remaining timing signoff procedures. For easier navigation, the vital notations utilized in this paper are explicitly defined in Table 1.

2.1 Cross-Corner Interconnect Timing Prediction

In this subsection, we assume that the known corner is given, the primary challenge revolves around predicting the interconnect timing across corners, which can be further divided into two problems:

- **Problem 1**: Given the observed timing for the known corner, the objective is to accurately capture the correlations between different corners and estimate the interconnect timing for the remaining $N-1$ unobserved corners. This corresponds to the *Accurate Version* as depicted in Fig. 1.
- **Problem 2**: Based on only the parasitic information for a known corner, the objective is to accurately estimate the interconnect timing for all N unobserved corners. This corresponds to the *Fast Version* as illustrated in Fig. 1.

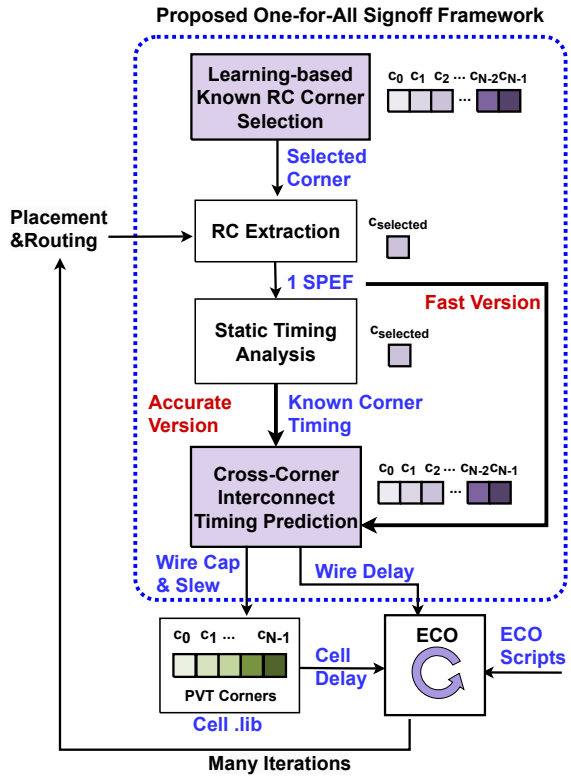


Figure 1: Overview of the proposed one-for-all learning-based cross-corner timing signoff framework. The inputs are represented by text highlighted in blue.

To resolve these problems, we propose the cross-corner interconnect timing prediction strategy that converts complex RC networks into graph representations. This approach then utilizes graph theory to tackle the challenges associated with cross-corner variations. The overall strategy is shown in Fig. 2. We firstly define two matrices denoted as $\mathbf{Y}(z) = \mathbf{y}_{i,j}(z)$ and $\mathbf{P}(z) = \mathbf{p}_{i,j}(z)$, which store the timing and parasitic parameters of the j-th net under the i-th RC corner for the circuit z respectively. Specifically, parasitic parameter $\mathbf{p}_{i,j}(z)$ is stored in SPEF file and timing $\mathbf{y}_{i,j}(z)$ is stored in timing report file. The overarching objective of the timing signoff process is to obtain a complete matrix $\mathbf{Y}(z)$ with the assistance of $\mathbf{P}(z)$. Previous work [10] made an hypothesis that many columns of matrix $\mathbf{Y}(z)$ are linearly dependent, which utilizes the internal correlation of matrix $\mathbf{Y}(z)$. Linear regression model f_w is thus employed to predict timing of unobserved corner in this work, as follows:

$$f_w(\mathbf{Y}(z)^{i\in\mathcal{I}}) = w^\top \mathbf{Y}(z)^{i\in\hat{\mathcal{I}}} \tag{1}$$

where $\mathcal{I} = \{observed\ l\ corners\}$, $\hat{\mathcal{I}} = \{unobserved\ corners\}$ and w is coefficient weights. Given data $\mathcal{D}_M = \{\mathbf{Y}(z)^{i\in\mathcal{I}}, \mathbf{Y}(z)^{i\in\hat{\mathcal{I}}}\}$, where $\mathbf{Y}(z)^{i\in\mathcal{I}} \in \mathbb{R}^{M\times l}$, and $\mathbf{Y}(z)^{i\in\hat{\mathcal{I}}} \in \mathbb{R}^{M\times(N-l)}$, the model is then trained by:

$$w_t \leftarrow w_{t-1} - \mathbf{Y}(z)^{i\in\mathcal{I}\top}\left(\mathbf{Y}(z)^{i\in\mathcal{I}} w - \mathbf{Y}(z)^{i\in\hat{\mathcal{I}}}\right) \tag{2}$$

This strategy requires obtained signoff timing data from observed corners (typically larger than 2). The selection of these observed corners is crucial for prediction accuracy. However, there is a strong desire to minimize the number of observed corners in order to save runtime and computation resources. Furthermore, the assumption that the selection of these corners remains constant across designs overlooks the possibility that new designs may introduce new RC topologies. $\mathbf{P}(z)$ encompasses unique interconnect topology features for circuit z and is indispensable for timing prediction.

To address the limitations from previous studies, as shown in Fig. 2, we initially introduce a Graph Mapping (GM) function $\mathcal{GM}(\cdot) : \mathbf{P}(z) \rightarrow \mathbf{P}^g(z)$, which maps the original euclidean matrix $\mathbf{P}(z)$ to a graph matrix $\mathbf{P}^g(z)$. The mapping process is detailed in Algorithm 1. Within this framework, for each net in $\mathbf{P}(z)$, a heterogeneous graph $G(\mathcal{V}_{i,j}, \mathcal{E}_{i,j})$ is constructed, where capacitance and resistance are represented as nodes $\mathcal{V}$ and edges $\mathcal{E}$, respectively. Capacitances that function as output nodes are grouped into the node set $\mathcal{V}^o$. Since each net features a varying number of output nodes, there are differing numbers of timing paths and labels associated with each. Node attributes $a(v)$, including capacitance values and Elmore delay values, are assigned to the nodes. In advanced technology nodes, due to the complexities of parasitics, the condition $|\mathcal{V}i, j| < |\mathcal{E}i, j| + 1$ might exist, indicating a cyclic graph, a loop-breaking procedure is required. Ultimately, this results in the formation of the Matrix $\mathbf{P}^g(z)$. This GM function effectively simplifies complex wire topologies into a higher-dimensional feature space.

For each heterogeneous graph $G \in \mathbf{P}^g(z)$, output nodes $\mathcal{V}^o$ and non-output nodes do not uniformly influence each other. Given the homophily principle of Graph Neural Networks (GNNs) as discussed in [12], graphs with nodes carrying diverse labels (heterophilous graphs) tend to be less effectively processed by standard GNN architectures like Graph Convolutional Networks (GCN). To address this, instead of using trainable GNN layers, we employ non-trainable Graph Filters (GF) inspired by convolution kernel filters from Convolutional Neural Networks (CNN). These filters are designed to capture local structural information around each node without the need for training. As illustrated in Fig. 2, we introduce a Graph Filtering (GF) function $\mathcal{GF}(\cdot) : \mathbf{P}^g(z) \rightarrow \mathbf{P}^h(z)$, which extracts low-dimensional embedding data. Since $\mathbf{P}^g(z)$ contains different topology of the same net in different corners, subtle discrepancy exists. The function of GF is to filter and amplify the differences, then distill it into low-dimensional feature ($\mathbf{P}^h(z)$), enhancing the correlation between $\mathbf{P}^h(z)$ and $\mathbf{Y}(z)$. For each graph $G \in \mathbf{P}^g(z)$, with node attribute $a(v) \in \mathbb{R}^{d_0}$, the first filter layer employs graph kernel tricks based on node-centered subgraphs based on [5]:

$$\phi_0(v) = a(v) \tag{3}$$

$$\phi_{1,i}(v) = K(G_{v^o}, H_i^{(1)}) \tag{4}$$

$$K(\cdot,\cdot) = \sum_{p=0}^{P} \lambda_p \sum_{i,j=1}^{|\mathcal{V}\times|} \left[A^{\times p}\right]_{ij} \tag{5}$$

Here, $H_i^{(1)}$ indicates subgraph hidden representation in the first layer. ϕ_l is node embedding (also termed as node feature) in layer l. $K(\cdot,\cdot)$ represents P-step random walk graph kernel. This layer updates the node's embedding based on node $\mathcal{V}^o$ centered subgraph G_{v^o}. The node embeddings $\phi_0(v)$ are transformed into $\phi_1(v)$, aggregating local information from neighbors to the central node $\mathcal{V}^o$. Utilizing a random walk kernel makes it computationally

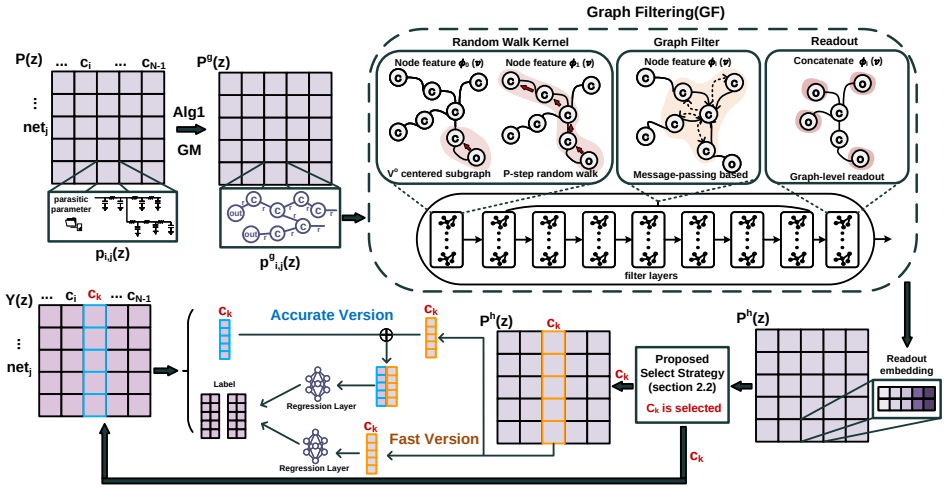


Figure 2: The proposed cross-corner interconnect timing prediction strategy (accurate version and fast version).

Algorithm 1: Graph Mapping

Input: Matrix $\mathbf{P}(z)$;
Output: Matrix $\mathbf{P}^g(z)$;
for $p_{i,j}$ *in* $P(z)$ **do**
  Assign capacitance as node set $\mathcal{V}_{i,j}$ from $\mathbf{p}_{i,j}$;
  Assign output nodes as o-node set $\mathcal{V}^o_{i,j}$;
  Assign resistance as edge set $\mathcal{E}_{i,j}$ from $\mathbf{p}_{i,j}$;
  $G(\mathcal{V}_{i,j}, \mathcal{E}_{i,j})$ with adjacency matrix $\mathcal{A}_{i,j} \in \mathbb{R}^{|\mathcal{V}_{i,j}|\times|\mathcal{V}_{i,j}|}$;
  Capacitance value, Elmore delay $\sum_k R_{ks}C_k \rightarrow$ node attribute $a(v)$;
  while $|\mathcal{V}_{i,j}| < |\mathcal{E}_{i,j}| + 1$ **do**
    $G(\mathcal{V}_{i,j}, \mathcal{E}_{i,j}) \xrightarrow{breaking\ loop\ based\ on[3]} \hat{G}(\hat{\mathcal{V}}_{i,j}, \hat{\mathcal{E}}_{i,j})$
  end
  $\mathbf{p}^g_{i,j} = \hat{G}(\hat{\mathcal{V}}_{i,j}, \hat{\mathcal{E}}_{i,j})$;
end
return Matrix $\mathbf{P}^g(z)$.

feasible to traverse all paths within the receptive field, enabling the extraction of valuable information from neighboring nodes for the target one, thus enhancing the model's capacity to interpret and leverage the structural data in the graph. Graph filters based on message-passing techniques are utilized in the subsequent layers, where the node embedding map is defined as: $\phi_l(v) = combine\Big(\phi_{l-1}(v), aggregate\Big(\phi_{l-1}(u)\Big)\Big)$, where $u \in Neighborhood(v)$. Graph-level readout layer is then deployed to generate the embedding for the entire graph, which concatenates node embedding from all layers:

$$p^h_{i,j} = concat\Big(\sum_{v\in\mathcal{V}^o} \phi_l(v)\Big| l = 0, 1, \ldots, 7\Big) \tag{6}$$

Assuming corner c_k is selected as an observed corner using our proposed selection strategy (Section 2.2), Multilayer Perceptron (MLP) regression layers predict the timing for the remaining corners. In the fast version, the regression layer outputs all columns of the matrix $\mathbf{Y}(z)$ except c_k, with only the column c_k from matrix $\mathbf{P}^h(z)$ as input. Conversely, in the accurate version, the regression layer also predicts the remaining columns of $\mathbf{Y}(z)$ but uses both the column c_k from $\mathbf{P}^h(z)$ and the entire column from $\mathbf{Y}(z)$ concatenated as its input.

2.2 Learning-based Known RC Corner Selection

Previous approaches, such as [10], utilize a greedy-based search policy to traverse all corners and select known corners, which are then fixed for all designs. However, this fixed selection may not be optimal for different designs or different technology nodes, necessitating re-selection of known corners. To address this, we propose a CNN classifier-based corner selection strategy as shown in Fig.3. Given a trained cross corner timing predictor model, this classifier will output the best known or observed corner (c_k in Fig. 2) by signing off only a few nets as input feature. Here, we first define the Mean Absolute Error (MAE), a metric used to evaluate the prediction error between the proposed model and the actual timing values. It is calculated as follows:

$$MAE = \frac{\sum_{i=1}^{n} |predicted_i - actual_i|}{n} \tag{7}$$

where n is the total number of predictions. MAE($c0, c1$) represents the MAE for the predicted timing value of an unknown corner $c1$ using a selected known corner $c0$. This metric can thus reflect the correlation between corner $c1$ and $c0$ within a trained timing prediction model. However, accurately calculating MAE for a new design requires sign-off on all timing for the entire test dataset, which is impractical for fast inference when the goal is merely to assess the correlation between corners and select the best one. To address this, we propose replacing the accurate MAE with a more easily calculated metric, termed "blurred" MAE. Subsequently, a learning-based classifier is trained to select the best candidate corner based on this blurred MAE as input feature.

The process for employing the "blurred" MAE to predict the known corner is outlined here. In the case of a new design, a small subset of S nets, usually ranging from 100 to 500 as observed in our experiments, is randomly selected to run through the actual sign-off process. The "blurred" MAE is subsequently calculated between the trained predictor and the golden timing values of these sampled nets by same Equation 7. Since S represents only a fraction of the larger dataset, the blurred MAE varies with different samples, providing insight into the underlying correlation between corners rather than directly assessing prediction accuracy. After

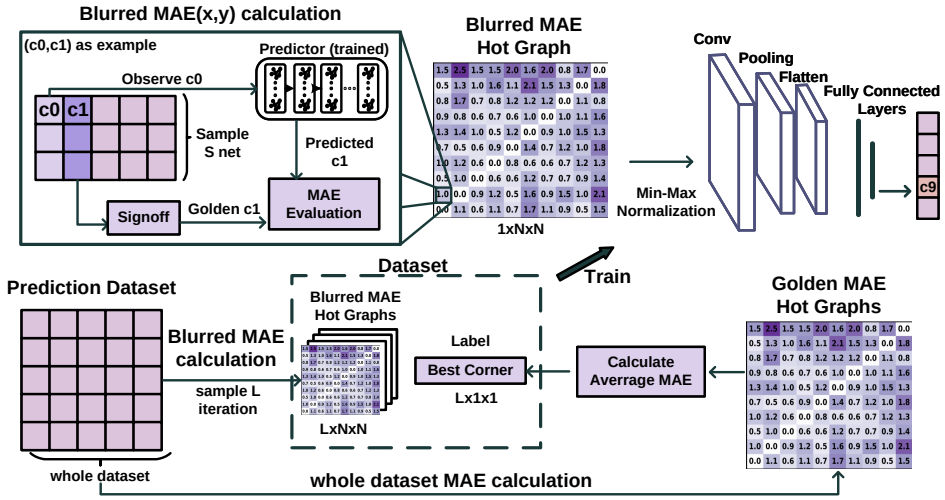


Figure 3: Learning-based known corner prediction strategy that employs a CNN classifier.

Table 2: Benchmark Statistics

Process	Benchmark	#Cells	#Nets	#FFs
28nm	B19	13863	15394	1980
	PCI_BRIDGE	8518	9269	3337
	DES_PERT	1506	1634	192
	AES-128	5363	5432	935
	SALSA	11292	12749	1674
	OPENGFX	8220	8534	1584
sub-10nm	Industrial Design (confidential)	Over 1 million nets		

computing the "blurred" $MAE(c_i, c_j)_{0 \le i < N}^{0 \le j < N}$, a $1 \times N \times N$ "blurred" MAE heatmap is generated, which acts as a single-channel input feature, where N is the number of corners. A CNN model, consisting of convolutional layers, pooling layers, and fully-connected layers, is then used to classify the best corner based on this input feature heatmap. It's important to note that the predictor model used can be any type, whether a fast, accurate version or based on previous work, making this selection strategy broadly applicable.

For training on the dataset, the best predictive corner k for each design is determined from the actual MAE of entire dataset (not the sampled nets) and recorded as classification labels. This procedure is repeated for each design L times, resulting in $L \times N \times N$ labeled graphs. By framing the selection as a classification problem, we provide an adaptive approach for new designs and process nodes to choose the best "known corner". The selection results from this strategy will be discussed in Section 3.3.

2.3 Integration with ECO Flow

Unlike prior work that often overlooked the integration and compatibility of such frameworks within the established design flow, we have taken steps to address this concern. In our proposed work, we have seamlessly integrated the proposed framework with a conventional ECO flow that utilizes existing commercial implementation tools, as depicted in Fig. 1. The interface has been carefully designed to ensure smooth integration, enabling the cross-corner interconnect timing information to be fed into state-of-the-art signoff tools. Subsequently, the gate timing information is collected, and comprehensive timing reports are generated for all corners. Based on these reports, either user-defined or automatically generated ECO scripts are utilized and executed through implementation tools to support incremental place and route adjustments. Typically, the ECO process will be iterated tens of times before the final timing closure, depending on the complexity of the design. This integration ensures that our framework aligns seamlessly with existing design workflows and processes.

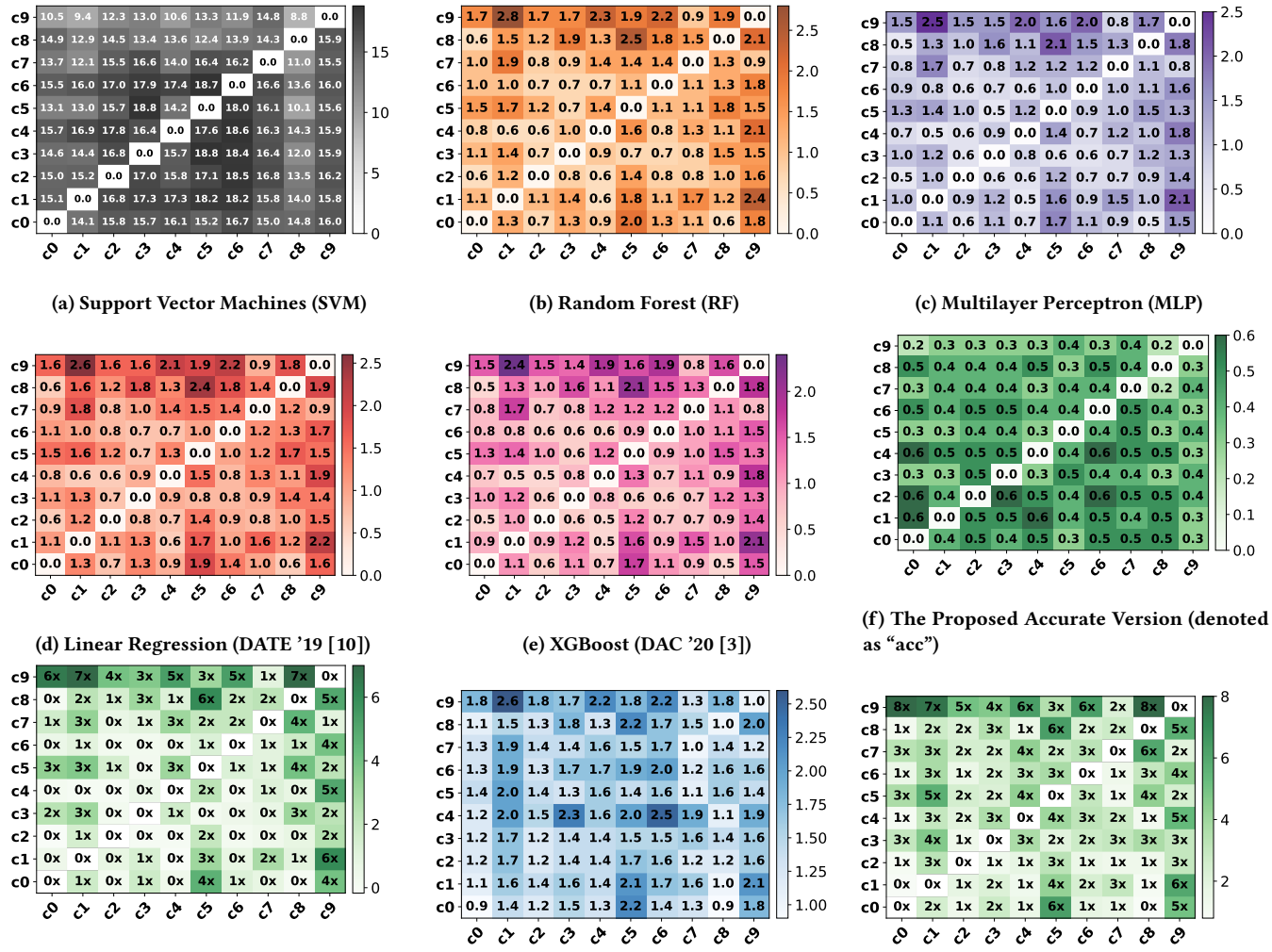
3 EVALUATION RESULTS

The proposed framework is evaluated using two industry-standard technology nodes: the cutting-edge sub-10nm and the more established 28nm. Two datasets are developed for this purpose. The sub-10nm data is obtained from industrial in-house extraction and STA tools based on an internal design, which, unfortunately, cannot be disclosed due to NDA rules. This design has over a million nets. The 28nm dataset is created using commercial tools and open-source designs. The 28nm data is generated from a commercial implementation and signoff tools suites based on widely used open-source designs. To ensure fair comparisons with prior work, such as [19, 20], the evaluation includes the same benchmark designs that are available in the public domain. A summary of the benchmarks is provided in Table 2.

We implement the proposed framework with PyTorch [15] and the Deep Graph Library (DGL) [16], a widely-used graph learning toolkit. The prediction model is trained and tested on a Linux machine with a 3.7GHz AMD 5900X CPU and 128GB RAM. The training and testing datasets are split for each design with the ratio $8:2$. We evaluate using MAE against signoff tools for prediction accuracy. For both technology nodes, the number of RC corners is 10, denoted as $[c_0, \ldots, c_9]$. Runtime has also been recorded for each process.

3.1 Cross-Corner Interconnect Timing Prediction Accuracy and Runtime Analysis

The baselines for comparing cross-corner prediction accuracy have been carefully chosen. The work by [10] represents the state-of-the-art in cross-corner signoff interconnect delay prediction and employs a linear regression (LR) model. While [3] is one of the



(a) Support Vector Machines (SVM)

(b) Random Forest (RF)

(c) Multilayer Perceptron (MLP)

(d) Linear Regression (DATE '19 [10])

(e) XGBoost (DAC '20 [3])

(f) The Proposed Accurate Version (denoted as "acc")

(g) Accurate Version over Previous Best (denoted as "pb") across (a) - (e)

(h) The Proposed Fast Version (denoted as "fa")

(i) Accurate Version over Fast Version

Figure 4: MAE (in ps) heatmap for cross-corner interconnect delay prediction results using various regression and learning-based approaches in the sub-10nm technology node, with one known corner. The X-axis represents the corner used as the sole known corner, and the Y-axis denotes the predicted MAE for the remaining unobserved corners for (a) through (h). Y-axis represents the accuracy differences between accurate and fast versions for (i).

Table 3: A summary of the predicted MAE (ps) (mean/min/max) of the unobserved corners when using c_i as the observed corner in the sub-10nm technology node.

Benchmark		c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
Industrial Design	**pb**	1.5/0.8/2.4	1.2/0.5/2.1	1.0/0.7/1.7	0.8/0.6/1.5	1.0/0.5/1.5	0.8/0.5/1.8	0.8/0.6/1.3	0.8/0.5/1.4	1.1/0.5/2.1	0.9/0.5/1.7
	fa	1.8/1.3/2.6	1.5/1.1/2.2	1.4/1.2/1.9	1.6/1.3/2.0	1.5/1.3/2.0	1.8/1.2/2.5	1.5/1.4/1.7	1.4/1.4/1.7	1.6/1.1/2.1	1.4/1.2/2.2
	acc	**0.3/0.2/0.4**	**0.4/0.3/0.5**	**0.3/0.2/0.4**	**0.4/0.3/0.5**	**0.3/0.3/0.5**	**0.4/0.3/0.6**	**0.3/0.3/0.5**	**0.5/0.4/0.6**	**0.4/0.3/0.6**	**0.4/0.3/0.5**

latest learning-based models for signoff interconnect timing prediction in a single corner and utilizes the powerful XGBoost model [2]. In addition to reproducing these two previous works, we also implement the framework using other regression or ML models, including Support Vector Machine (SVM), Random Forest (RF), and Multi-layer Perceptron (MLP) models, offering a broad comparison to gauge our framework's efficacy in cross-corner interconnect timing predictions. In the following sections, the predicted accuracy for each technology node will be presented.

3.1.1 Interconnect Delay Prediction Accuracy in sub-10nm. In Fig. 4, the MAE of the predicted interconnect delay is visualized in a heatmap for all aforementioned methods when using only one corner as the known corner in the sub-10nm node. The X-coordinate (c_x) represents the selected observed corner (c_0 to c_9), and the Y-coordinates (c_y) denote the predicted corners (c_0 to c_9). Fig. 4a through 4f depict the results when using signoff data from the known corner, while Fig. 4h illustrates the fast version when predicting results based on the SPEF of the known corner directly.

Table 4: The number of required corners for the previous best methods to attain a comparable MAE level when compared against the proposed accurate version of the prediction strategy, evaluated in the sub-10nm process node.

	Proposed	Previous Best							
Known Corner	#1	#1	#2	#3	#4	#5	#6	#7	# Minimum Required Corner
c_0	**0.30**	1.62	0.83	0.5	0.38	0.40	0.39	**0.29**	7
c_1	**0.41**	1.36	0.62	0.47	0.50	0.50	0.53	**0.27**	7
c_2	**0.36**	1.06	0.64	0.52	0.51	0.53	0.38	**0.28**	7
c_3	**0.43**	0.88	0.60	0.60	0.61	**0.36**			5
c_4	**0.37**	1.12	0.62	0.63	**0.34**				4
c_5	**0.49**	0.92	0.74	0.50	**0.38**				4
c_6	**0.38**	0.89	0.63	0.41	**0.38**				4
c_7	**0.50**	0.83	0.77	0.71	0.76	**0.36**			5
c_8	**0.48**	1.18	0.93	0.76	**0.36**				4
c_9	**0.43**	1.02	0.77	**0.40**					3
Average									5

Additionally, the proposed accurate version is compared against the previous best results across all previous models, as well as against the results of the fast version. These comparisons are visually represented in Fig. 4g and Fig. 4i, respectively. The notation 0× indicates unchanged accuracy, and n× signifies that the accurate version is n times more accurate than the previous best or fast version result under the same condition. This comparison demonstrates that the proposed accurate model outperforms all previous approaches, achieving a maximum of 7× accuracy improvement. For the proposed fast version, although it is less accurate compared to the accurate version, its MAE remains close to the previous best values in most cases, indicating its ability to swiftly predict interconnect delay with acceptable accuracy levels. The cross-corner interconnect delay prediction results in the sub-10nm node are also summarized in Table 3, presenting the mean, minimum, and maximum MAE values when using c_i as the observed corner. The accurate version (acc) notably outperforms the other two strategies, while the fast version (fa) achieves an acceptable level of accuracy when compared against the previous best (pb) version.

Previous methods, as shown in [10], generally require observing multiple corners. Our experiments aimed to identify the fewest corners needed by these methods to match the MAE of our proposed single-corner prediction approach, using c_x as the observed corner. The findings, detailed in Table 4, indicate that an average of 5 corners is necessary for prior best methods to achieve comparable MAE levels to our strategy, which uses just one corner. This suggests a substantial decrease in both runtime and computational resources needed for data collection, while enhancing accuracy.

3.1.2 Interconnect Delay Prediction Accuracy in 28nm. Utilizing a methodology similar to the sub-10nm node evaluation, we assess the proposed framework in the 28nm node and present the results in Table 5. The table includes the average MAE values across all unobserved corners, along with the runtime for each benchmark design. The results exhibit a similar trend compared to the sub-10nm node evaluation shown in Table 3. The accurate version outperforms previous best strategies by orders of magnitude with only a marginal runtime increase of approximately 10s across all designs. The proposed fast version achieves over a 2× speedup compared to the accurate version with an accuracy degradation of less than 0.5 ps on average.

Another state of the art work [19] proposes a graph-learning based model to predict signoff interconnect delay under a single corner. While this prior work is relevant, it is challenging to reproduce due to limited information provided in the paper. Despite these challenges, we made our best effort to compare against it, assuming that the total runtime in their case is the product of the number of corners and the single-corner time. Additionally, it should be noted that the technology nodes are different. In cases where processes can run in parallel, this runtime metric can also be converted into the total required amount of computing resources. Table 6 summarizes the comparison results, demonstrating that the proposed framework outperforms [19] in both accuracy and runtime across all benchmark circuits.

3.1.3 Cross-Corner Interconnect Slew and Load Capacitance Prediction. Referring to Fig. 1, accurately predicting cell delay requires obtaining interconnect slew and net capacitance, using library lookup tables. We employ learning-based cross-corner approaches, similar to those used for estimating interconnect delays in this paper, to predict interconnect slew and net capacitance values. Table 7 compares the accuracy of our method with previous techniques for a 28nm process. The results demonstrate a substantial improvement in our method, reducing the MAE in slew predictions from 3ps to 0.2ps and in net capacitance estimates from 0.1fF to 0.03fF across all RC corners.

3.2 ECO Runtime Improvement

In traditional ECO flows, the STA tools and parasitic extraction tools need to be invoked for each corner, contributing to the complexity and duration of each ECO iteration. In the proposed framework, runtime has been significantly reduced and can be segmented into three primary components: the extraction time and signoff time for a single observed corner (for the accurate version), along with the cross-corner prediction time. Table 8 demonstrates that in an ECO iteration, the proposed framework achieves over 10× runtime improvement when compared against a traditional ECO flow that relies solely on commercial tools. This showcases promise for the application of this framework in the signoff of designs. For fast version, it is able to speed up the process by over 2× further with scarification of accuracy. It's important to note that while the fast version lacks perfect accuracy, it remains highly beneficial in the early design stages for assessing coverage, which is typically optimized for just one or a few corners. This approach offers designers a rapid way to evaluate scenarios beyond the chosen implementation corners, helping to avoid unforeseen issues or potential redesigns later in the design process.

3.3 Known Corner Selection Accuracy

To evaluate the known corner selection accuracy that is discussed in Section 2.2, we generate a dataset with 600 labeled "blurred" MAE heatmap graphs, varying the sample size S from 100 to 500. For example, sample size=100 means 100 nets are required to sample for signoff to generate blured MAE heatmap as input feature. The experiment result shown in Table 9 is conducted on the 28nm node, and the training and testing dataset are split at a 7:3 ratio. The "mixed" label indicates a dataset mixed from the previous three sample groups. The results, as shown in Table 9, indicate that even with a relatively small sample size of 200, the test accuracy surpasses 70%. The classification accuracy remains almost unchanged when the sample size reaches 500. Sample sizes from 100 to 500

Table 5: A summary of the predicted MAE (ps) (mean) of the unobserved corners when using c_i as the observed corner in the 28nm node. The runtime (s) is also listed.

Benchmark		c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	Run Time (s) [1]
B19	**pb**	0.122	0.133	0.162	0.169	0.105	0.110	0.151	0.162	0.111	0.116	161
	fa	0.464	0.483	0.529	0.540	0.457	0.466	0.552	0.548	0.501	0.519	**53**
	acc	**0.005**	**0.005**	**0.005**	**0.006**	**0.005**	**0.005**	**0.006**	**0.006**	**0.005**	**0.006**	172
PCI_BRIDGE	**pb**	0.240	0.268	0.258	0.269	0.185	0.197	0.254	0.261	0.192	0.190	115
	fa	0.801	0.860	0.857	0.860	0.752	0.811	0.869	0.865	0.810	0.827	**50**
	acc	**0.011**	**0.011**	**0.011**	**0.011**	**0.011**	**0.011**	**0.012**	**0.012**	**0.011**	**0.011**	126
DES_PERT	**pb**	0.088	0.084	0.107	0.094	0.078	0.078	0.108	0.088	0.063	0.067	59
	fa	0.450	0.459	0.625	0.600	0.425	0.508	0.572	0.600	0.499	0.517	**42**
	acc	**0.011**	**0.012**	**0.012**	**0.013**	**0.011**	**0.012**	**0.013**	**0.013**	**0.013**	**0.012**	69
AES-128	**pb**	0.240	0.268	0.258	0.269	0.185	0.197	0.254	0.261	0.192	0.190	96
	fa	0.802	0.854	0.891	0.920	0.818	0.839	0.930	0.927	0.854	0.876	**47**
	acc	**0.009**	**0.010**	**0.010**	**0.010**	**0.009**	**0.009**	**0.010**	**0.010**	**0.010**	**0.010**	107
SALSA	**pb**	0.341	0.391	0.407	0.442	0.287	0.289	0.388	0.419	0.273	0.281	150
	fa	1.436	1.475	1.502	1.560	1.397	1.455	1.569	1.610	1.467	1.518	**54**
	acc	**0.012**	**0.012**	**0.012**	**0.012**	**0.012**	**0.012**	**0.012**	**0.013**	**0.012**	**0.012**	161
OPENGFX	**pb**	0.147	0.162	0.205	0.220	0.135	0.135	0.218	0.223	0.135	0.138	123
	fa	0.620	0.646	0.682	0.706	0.609	0.647	0.692	0.723	0.650	0.671	**49**
	acc	**0.007**	**0.007**	**0.007**	**0.008**	**0.007**	**0.007**	**0.008**	**0.008**	**0.007**	**0.007**	134

[1] This runtime encompasses the total time required for predicting interconnect timing for all unobserved corners. Notably, it doesn't include preprocessing time, which is deemed negligible and therefore excluded for all conditions.

Table 6: Comparison against a state-of-the-art work (DATE '23 [19]) targeting single-corner interconnect delay prediction with graph-learning based model.

			B19	PCI_BRIDGE	DES_PERT	AES-128	SALSA	OPENGFX
MAE (ps)	16nm	**[19]**	-	-	0.43	1.14	-	2.54
	28nm	**acc**	**0.005**	**0.011**	**0.011**	**0.009**	**0.012**	**0.007**
		fa	0.457	0.752	0.450	0.802	1.397	0.620
Runtime (s)	16nm	**[19]**	-	-	274	567	-	976
	28nm	**acc**	172	126	69	107	161	134
		fa	**53**	**50**	**42**	**47**	**54**	**49**

Table 7: MAE from the cross-corner slew (ps)/net load cap (fF) prediction in 28nm.

Benchmark		c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	Avg.
B19	**pb**	2.64/ 0.06	2.85/ 0.07	2.59/ 0.07	2.73/ 0.06	2.88/ 0.07	2.88/ 0.06	2.32/ 0.06	2.75/ 0.07	2.42/ 0.06	2.67/ 0.06	2.67/ 0.07
	acc	**0.19/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.19/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.19/ 0.02**	**0.18/ 0.02**
PCI_BRIDGE	**pb**	3.03/ 0.08	3.34/ 0.10	3.10/ 0.10	3.09/ 0.08	3.91/ 0.10	3.35/ 0.08	2.70/ 0.08	3.28/ 0.10	2.90/ 0.09	3.22/ 0.09	3.19/ 0.09
	acc	**0.18/ 0.02**	**0.19/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.19/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**	**0.18/ 0.02**
DES_PERT	**pb**	1.34/ 0.05	1.21/ 0.05	1.28/ 0.05	1.21/ 0.05	1.53/ 0.05	1.21/ 0.05	1.21/ 0.05	1.13/ 0.05	1.20/ 0.04	1.07/ 0.05	1.24/ 0.05
	acc	**0.19/ 0.02**	**0.20/ 0.02**	**0.20/ 0.03**	**0.21/ 0.03**	**0.20/ 0.02**	**0.19/ 0.02**	**0.20/ 0.03**	**0.21/ 0.03**	**0.20/ 0.02**	**0.20/ 0.02**	**0.20/ 0.02**
AES-128	**pb**	4.12/ 0.06	4.91/ 0.07	3.92/ 0.07	4.32/ 0.07	4.65/ 0.07	4.70/ 0.07	3.47/ 0.06	4.08/ 0.07	3.87/ 0.06	4.43/ 0.06	4.25/ 0.07
	acc	**0.22/ 0.02**	**0.22/ 0.02**	**0.22/ 0.02**	**0.23/ 0.02**	**0.22/ 0.02**	**0.22/ 0.02**	**0.22/ 0.02**	**0.22/ 0.02**	**0.23/ 0.02**	**0.22/ 0.02**	**0.22/ 0.02**
SALSA	**pb**	3.72/ 0.21	3.60/ 0.23	3.75/ 0.23	3.53/ 0.22	4.36/ 0.22	3.79/ 0.21	3.79/ 0.20	3.63/ 0.22	3.54/ 0.20	3.53/ 0.21	3.73/ 0.22
	acc	**0.17/ 0.06**	**0.16/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**	**0.17/ 0.06**
OPENGFX	**pb**	3.43/ 0.10	3.61/ 0.12	3.37/ 0.12	3.11/ 0.10	3.94/ 0.11	3.75/ 0.10	3.13/ 0.10	3.14/ 0.11	3.25/ 0.10	3.37/ 0.10	3.41/ 0.11
	acc	**0.23/ 0.03**	**0.22/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**	**0.24/ 0.03**	**0.23/ 0.03**	**0.23/ 0.03**

Table 8: Runtime (s) for two ECO iterations (in 28nm).

	Iteration #1			Iteration #2		
Benchmark	**fa**	**acc**	**Traditional ECO**	**fa**	**acc**	**Traditional ECO**
B19	53	169	1574	53	172	1608
PCI_BRIDGE	70	147	1354	50	126	1154
DES_PERT	43	79	697	42	69	588
AES-128	42	112	1009	47	107	962
SALSA	53	127	1582	54	161	1502
OPENGFX	50	100	1285	49	134	1228
Average	**51.83**	**122.33**	**1250.17**	**49.17**	**128.17**	**1173.67**

nets represent a very small subset compared to the design scales in our benchmark circuits listed Table 2, indicating minimal signoff

Table 9: Know corner selection accuracy with respect to the sample size (in 28nm).

Sample Size	Training Accuracy	Test Accuracy
100	0.663	0.611
200	0.805	0.744
500	0.771	0.767
Mixed	0.814	0.8

cost is required for our classifier to predict the best known corner. When similar tests are conducted on sub-10nm processes, the test accuracy also reaches a high level, exceeding 90%. These outcomes affirm the efficiency of our proposed CNN-based method in accurately selecting an "optimal" corner for specific designs with limited samples, demonstrating its adaptability across various designs and technology nodes.

4 CONCLUSIONS AND FUTURE WORK

In this paper, a learning-based cross-corner timing signoff framework is proposed. It features an "one-for-all" approach requiring only one RC corner to predict interconnect timing for the remaining unobserved corners. Additionally, we propose a learning-based strategy for selecting known corners, enhancing the framework's adaptability across different designs and technology nodes. Through comprehensive evaluations on two mainstream technology nodes and comparisons with various state-of-the-art methodologies, our framework demonstrates significant accuracy improvements with an acceptable runtime penalty. We also introduce a fast version for swift timing estimation. Integrated into the ECO flow, the proposed framework accelerates each iteration substantially by over 10×. We plan to open-source this framework to support community collaboration.

ACKNOWLEDGEMENT

This work is supported in part by the National Science and Technology Major Project (Grant No. 2021ZD0114701), the State Key Laboratory of Integrated Chips and Systems (SKLICS) open fund, and the SJTU Explore-X fund.

REFERENCES

[1] Tuck-Boon Chan, Sorin Dobre, and Andrew B. Kahng. 2014. Improved signoff methodology with tightened BEOL corners. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, Seoul, Korea (South), 311–316.

[2] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[3] Hsien Han Cheng, Iris Hui Ru Jiang, and Oscar Ou. 2020. Fast and accurate wire timing estimation on tree and non-tree net structures. *Proceedings - Design Automation Conference* 2020-July (2020), 0–5.

[4] Luis Guerra e Silva, L. Miguel Silveira, and Joel R. Phillips. 2007. Efficient Computation of the Worst-Delay Corner. In *Proceedings of the Conference on Design, Automation and Test in Europe* (Nice, France) *(DATE '07)*. EDA Consortium, San Jose, CA, USA, 1617–1622.

[5] Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. 2022. KerGNNs: Interpretable Graph Neural Networks with Graph Kernels. *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022* 36, 1 (2022), 6614–6622. arXiv:2201.00491

[6] Khosrow Golshan. 2020. *Multi-mode Multi-corner Analysis*. Springer International Publishing, Cham, 65–76. https://doi.org/10.1007/978-3-030-49636-4_2

[7] Jiang Hu and Andrew B Kahng. 2023. The Inevitability of AI Infusion Into Design Closure and Signoff. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 1–7.

[8] Iris Hui-Ru Jiang. 2023. Lightning Talk: All Routes to Timing Closure. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, San Francisco, CA, USA, 1–2.

[9] Andrew B Kahng. 2015. New game, new goal posts: A recent history of timing closure. In *Proceedings of the 52nd Annual Design Automation Conference*. 1–6.

[10] Andrew B. Kahng, Uday Mallappa, Lawrence Saul, and Shangyuan Tong. 2019. "Unobserved Corner" Prediction: Reducing Timing Analysis Effort for Faster Design Convergence in Advanced-Node Design. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Florence, Italy, 168–173.

[11] Kwangsu Kim, Byungha Joo, Young Min Park, Taeyang Jeong, Ki Tae Kim, and Eui-Young Chung. 2021. Cross-Corner Delay Variation Model for Standard Cell Libraries. *IEEE Access* 9 (2021), 72299–72315.

[12] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2023. Is Homophily a Necessity for Graph Neural Networks? arXiv:2106.06134 [cs.LG]

[13] Sari Onaissi and Farid N. Najm. 2008. A Linear-Time Approach for Static Timing Analysis Covering All Process Corners. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 7 (2008), 1291–1304.

[14] Sari Onaissi, Feroze Taraporevala, Jinfeng Liu, and Farid Najm. 2011. A Fast Approach for Static Timing Analysis Covering All PVT Corners. In *Proceedings of the 48th Design Automation Conference* (San Diego, California) *(DAC '11)*. Association for Computing Machinery, New York, NY, USA, 777–782. https://doi.org/10.1145/2024724.2024899

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[16] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2020. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. arXiv:1909.01315 [cs.LG]

[17] Wei W. Xing, Zheng Xing, Rongqi Lu, Zhelong Wang, Ning Xu, Yuanqing Cheng, and Weisheng Zhao. 2023. TOTAL: Multi-Corners Timing Optimization Based on Transfer and Active Learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, San Francisco, CA, USA, 1–6. https://doi.org/10.1109/DAC56929.2023.10247914

[18] Tianhao Yang, Zhenyu Zhao, Ao Han, and Guoqiang Liu. 2022. Automatic Timing ECO Using Stage-based Path Delay Prediction. In *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*. IEEE, Quebec City, QC, Canada, 455–459.

[19] Yuyang Ye, Tinghuan Chen, Yifei Gao, Hao Yan, Bei Yu, and Longxing Shi. 2023. Fast and Accurate Wire Timing Estimation Based on Graph Learning. *Proceedings -Design, Automation and Test in Europe, DATE* 2023-April, Date (2023), 1–6.

[20] Yuyang Ye, Tinghuan Chen, Yifei Gao, Hao Yan, Bei Yu, and Longxing Shi. 2023. Graph-learning-driven path-based timing analysis results predictor from graph-based timing analysis. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*. IEEE, Tokyo, Japan, 547–552.

[21] Zhenyu Zhao, Shuzheng Zhang, Guoqiang Liu, Chaochao Feng, Tianhao Yang, Ao Han, and Lei Wang. 2022. Machine-learning-based multi-corner timing prediction for faster timing closure. *Electronics* 11, 10 (2022), 1571.

[22] Linyu Zhu, Yue Gu, and Xinfei Guo. 2023. RC-GNN: Fast and Accurate Signoff Wire Delay Estimation with Customized Graph Neural Networks. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, Hangzhou, China, 1–5.