LiMo: A Framework Leveraging Machine Learning for Multi-Input Switching Timing Models of Complex Logic Gates

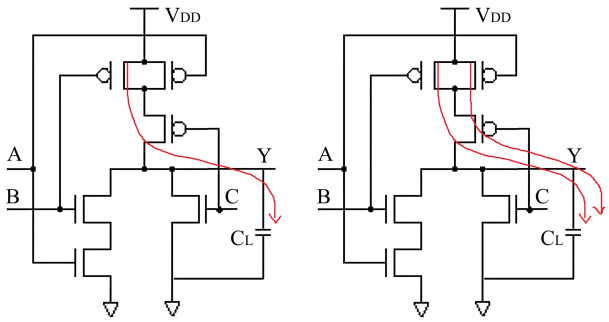
Pooja Beniwal, Sneh Saurabh, N Vignesh Chowdary, Suriya Skariah,
Ajoy Mandal and Ramakrishnan Venkatraman

***Abstract*—Traditional standard cell libraries employ a single-input switching (SIS) assumption for characterization. However, in real circuits, multiple inputs can switch simultaneously, resulting in a significant speed-up. This discrepancy can lead to circuit failures due to inaccurate early analysis, despite static timing analysis (STA) tools adopting conservative strategies. Consequently, multi-input switching (MIS) models have gained importance for timing libraries. For complex logic gates, however, the number of possible transitions that must be considered for characterization increases significantly, posing a substantial challenge in modeling MIS effects. We propose a novel approach to address this challenge by combining the power of satisfiability (SAT) solvers and machine learning (ML) techniques. First, we perform a logical analysis on the Boolean function of a given logic gate to identify input patterns that can lead to MIS-induced speed-up. This task is formulated as a SAT problem, and a SAT solver is utilized to extract all MIS-relevant transitions, thereby significantly reducing the characterization and modeling effort. Next, we leverage ML techniques to accurately capture the complex dependencies of MIS-induced speed-up under various circuit and environmental conditions. To streamline this process, we introduce an automated tool framework named LiMo (Library Model), which integrates a SAT solver, SPICE simulator, dataset optimization strategies, ML training/testing infrastructure, and multiprocessing capabilities to create MIS-aware timing libraries. The results on benchmark circuits reveal that ignoring MIS effects can result in relative root mean square errors (RRMSE) exceeding 40% in timing attributes. In contrast, LiMo-generated libraries achieve RRMSE errors below 7% compared to SPICE simulations, while delivering results $\sim 10,000\times$ faster, underscoring their suitability for incorporating MIS effects in STA for industrial designs.**

***Index Terms*—Single-input switching, multiple-input switching, technology libraries, machine learning, and timing verification.**

I. Introduction

Static Timing Analysis (STA) is an essential task in the VLSI design flow that ensures the timing correctness of a design. It decomposes a given design into various timing paths and verifies them conservatively. A timing path consists of timing arcs with delay attributes between the input–output pins of a cell, and the accuracy of STA largely depends on the timing models of these arcs. The most popular approach to model these timing arcs is using the liberty format look-up tables (LUTs) that offer a good trade-off between computational efficiency and accuracy [1]. During library characterization, delays corresponding to various combinations of input slews and load capacitances are simulated and stored in the library. The STA engine retrieves delay values from these tables and determines the delay under specific instantiation and operating conditions [2].



(a) (b)

Fig. 1: SIS and MIS (a) SIS (A= $1 \rightarrow 0$, B=1, C=0) (b) MIS (A= $1 \rightarrow 0$, B= $1 \rightarrow 0$, C=0)

However, a limitation of the traditional timing model is the Single-Input Switching (SIS) assumption that considers only one input of a gate transitioning at a time while others remain at non-controlling values. This simplification, though computationally efficient, does not account for realistic scenarios where multiple inputs of a gate can switch simultaneously, referred to as Multiple-Input Switching (MIS). Neglecting MIS effects can result in inaccuracies in STA, potentially leading to design failures. For instance, it can cause hold time violations due to overestimated early arrival times [3], [4].

An example demonstrating the impact of MIS in an AND-OR-Invert (AOI) cell is shown in Fig. 1. Under SIS, when input A transitions from $1 \rightarrow 0$ with $B = 1$ and $C = 0$, the output load capacitance C_L is charged through a single PMOS transistor, as depicted in Fig.1(a). However, under MIS, when both inputs A and B transition simultaneously from $1 \rightarrow 0$ with $C = 0$, the output load capacitance is charged through two parallel PMOS transistors, thereby reducing the delay compared to the SIS, as shown in Fig.1(b).

In practice, to account for potential speed-up caused by MIS, commercial timing analysis tools [5] often apply MIS derates. These derates adjust the nominal cell delays during early-mode (hold) analysis by applying a global or cell-specific scaling factor to the SIS-based delays. However, this approach is typically conservative and based on clock relationships and timing proximity window, rather than accurately modeling the actual impact of MIS based on temporal distance. For instance, as shown in this paper, there are scenarios where

two inputs switch simultaneously, yet no delay variation due to MIS is observed in complex gates. Therefore, applying these derates indiscriminately is unnecessary and can lead to overly pessimistic delay estimations. This underscores the need for more accurate modeling of MIS effects in timing analysis.

In the past, several methods have been proposed to model the impact of MIS on delay calculations. An analytical model is presented in [6], which leverages SIS delays from standard-cell libraries along with fitting constants derived from minimal MIS characterization. In [7], a polynomial approximation-based approach is proposed, requiring extensive Simulation Program with Integrated Circuit Emphasis (SPICE) simulations across multiple input combinations for coefficient extraction. Statistical timing analysis approaches [8], [9] utilize multiple SPICE simulations to extract coefficients, enabling analysis of the sensitivity of required arrival time to MIS-induced delays. In [10], a piecewise linear model is proposed using three or more discrete points to represent MIS effects, thereby reducing characterization effort. However, the model's accuracy heavily depends on the precise identification of these finite points. Moreover, the linear parameters must be fitted separately for each logic gate, input transition combination, and load condition. A hybrid delay modeling technique was introduced in [11], in which transistors are represented as ideal switches within a simplified RC network, and the resulting ordinary differential equations (ODEs) are solved analytically to yield explicit delay expressions for a two-input Complementary Metal-Oxide-Semiconductor (CMOS) NOR gate. Building on this foundation, the work in [12] shows that the model in [11] remains continuous even under MIS effects; and replaces the prior complicated approximation ODE solution with the found explicit ODE solutions to derive more accurate MIS delay formula, complete with a direct parametrization procedure. Although these modeling techniques provide valuable insights into MIS behavior, they often involve complex computations, limiting their practical integration into STA tools.

In recent times, machine learning (ML) has shown promise in overcoming various limitations of traditional timing models. Deep learning techniques have been applied to STA for accelerating delay predictions, accounting for variations and ageing effects, improving accuracy and runtime of characterization, and design technology co-optimization (DTCO) [13], [14], [15], [16], [17], [18]. Recent works have also introduced ML-based approaches for modeling MIS effects in timing model characterization [19], [20]. In [19], an Artificial Neural Network-based model was used to capture the impact of MIS on timing attributes, demonstrating higher prediction accuracy than the traditional timing models. Similarly, a Graph Neural Network-based model was proposed in [20], where gate topologies were represented as graphs, enabling the model to learn in a unified manner. However, a key challenge with these models is that as the number of inputs increases, the effort required for ML-based library characterization also grows significantly. The large number of possible input combinations leads to large MIS datasets or an exponential increase in the number of ML models, making the approach computationally expensive [10]. Capturing all input patterns exhaustively would necessitate an impractical number of simulations, resulting in excessive computational, data, and storage overhead. Using such models in STA can be cumbersome, limiting the practicality of incorporating these effects in realistic designs with multi-input and complex gates.

While previous studies have demonstrated that MIS-induced delay variations occur when two signals switch simultaneously, this is not always the case for complex gates. In some scenarios, even when two inputs switch simultaneously, no MIS-induced delay variation is observed for complex gates (as detailed in Section III). Filtering out such input patterns is crucial to reducing the size of the MIS dataset, thereby mitigating computational, data, and storage overhead, making MIS-aware characterization more practical for STA workflows. In this direction, a graph-based pattern reduction algorithm was proposed in [21] to identify the worst-case input patterns for complex gates. While this method significantly reduces characterization efforts, it focuses only on worst-case scenarios, which can lead to overly pessimistic timing estimates.

The key challenges of previous ML-based approaches to tackle MIS effects can be summarized as high computational, data, and storage overhead. Additionally, focusing on worst-case scenarios leads to overly pessimistic timing estimates. To address these, we propose a hybrid tool framework, LiMo (Library Model), that leverages the combined power of a satisfiability (SAT) solver and ML to efficiently create and use the MIS-aware timing models for complex logic gates in timing analysis. Using the Boolean function for the given logic gate, it extracts the input combinations that can produce MIS-induced delay variations by formulating it as a SAT problem and employing a SAT solver to obtain the MIS-relevant transitions. This approach reduces the number of simulations required for characterization, optimizes the size of the MIS dataset, eases timing analysis, and enhances the scalability of MIS-aware STA, particularly for complex logic gates. Moreover, LiMo efficiently leverages the power of ML in capturing complex dependencies among various design attributes impacting MIS effects by automating dataset generation, data handling, model training, and validation. It also reduces the characterization runtime significantly by decomposing the problem appropriately and utilizing multi-processing capabilities. Additionally, with a Tool Command Language (TCL)-based interface, debugging capabilities, and features such as handling Process-Voltage-Temperature (PVT) corners, LiMo proposes and demonstrates a practical solution for realistic industrial designs. The key contributions of this paper are as follows:

- Proposed a SAT-based MIS pattern extraction technique that identifies input transitions leading to MIS-induced delay variations by analyzing the standard cell's Boolean function, significantly reducing computational efforts of characterization.
- Proposed a ML-based MIS modeling approach that effectively captures timing variations caused by MIS under diverse conditions, while reducing data requirements, computational complexity, and storage overhead.
- Introduced an integrated tool framework, LiMo, which combines the SAT-based MIS pattern extraction tech-

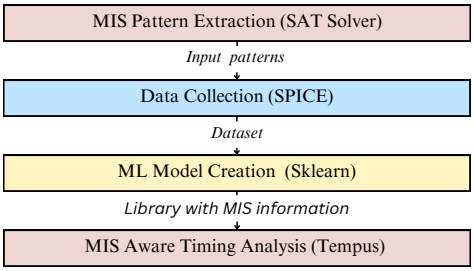


Fig. 2: Proposed Methodology

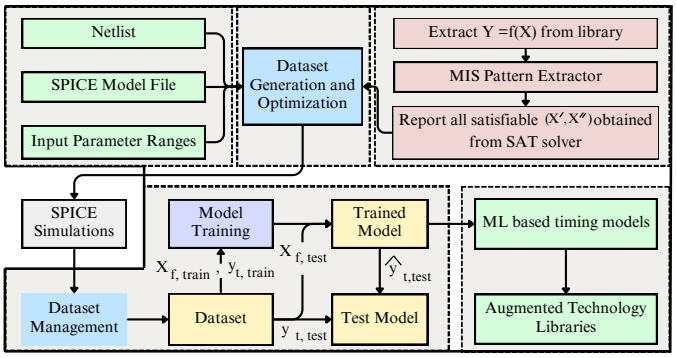


Fig. 3: LiMo's Block Diagram

nique, SPICE simulation, dataset optimization, multi-processing, ML-driven MIS modeling, and PVT corner handling to automate the generation of MIS-aware timing libraries.

- Evaluated and demonstrated improvements in accuracy obtained using MIS-aware timing libraries produced by Limo.

The remainder of this paper is structured as follows: Section II gives a top-level view of the proposed methodology, Section III examines the simultaneous switching timing behavior of complex CMOS logic gates, Section IV explains SAT solver-based MIS pattern extraction, Section V discusses ML-based model development, Section VI describes augmenting the traditional library with MIS information, Section VII presents results obtained using the proposed methodology, and Section VIII concludes the paper.

II. Proposed Methodology and Framework

The top-level view highlighting the major tasks in the proposed methodology is shown in Fig. 2. Given a complex gate, MIS-relevant patterns are extracted by formulating it as a SAT problem and taking the help of a highly efficient SAT solver [22]. For each identified pattern, data is collected using SPICE simulations to capture the impact of MIS on the timing attributes [23]. This data is then used to experiment and create an augmented technology library by leveraging ML techniques using the scikit-learn libraries [24]. Subsequently, MIS-aware STA is performed using Tempus loosely coupled with the generated libraries [5].

The proposed methodology is implemented in the form of LiMo, which is shown in Fig. 3. LiMo is an automated framework with a TCL-based interface, enabling seamless integration of the SAT solver, SPICE simulator, dataset optimization, ML training/testing infrastructure, and multi-processing

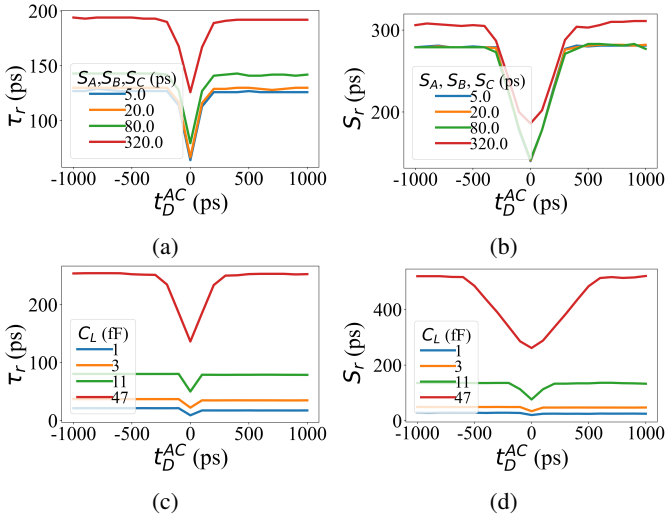


Fig. 4: Dependence of τ_r, S_r on t_D^{AC} in an AOI gate $Y = \neg(A.B + C)$ for varying (a, b) input slews (c, d) output capacitive load.

capabilities. It determines the MIS-relevant input transitions for each gate using the MIS pattern extractor, as explained in Section IV. Then, it efficiently generates training data using SPICE simulations, taking inputs from the designer about the PVT, and customization requirements, and applying data set optimization as explained in Section V. Since this is the most time-intensive stage of LiMo, multiple processing capabilities and load-balancing techniques are employed to optimize performance. The subsequent phase focuses on dataset management, where data analysis and visualization tools aid designers in making informed decisions during feature selection and outlier detection. The dataset is then partitioned into training and testing sets, followed by training and testing the model to evaluate its accuracy and generalization capabilities. Finally, LiMo generates trained ML models and augmented traditional libraries that enable STA tools to account for MIS-induced delay variations.

III. Simultaneous Switching Timing Behavior

A. Factors impacting behavior

The simultaneous switching behavior of multi-input gates is strongly influenced by the *temporal distance* or *skew*, which quantifies the time difference between the arrival of signals at the input pins. For a given gate, the temporal distance between two input pins i and j, denoted as t_D^{ij}, can be mathematically expressed as: $t_D^{ij} = (AT_{\mathrm{j}} - AT_{\mathrm{i}})$, where AT_{i} and AT_{j} represent the arrival times of the signal at inputs i and j, respectively. In this paper, for simplicity, we denote temporal distance as t_D when not referring to a specific pair of input pins.

The relationships between t_D and timing attributes, such as rise/fall delays (τ_r/τ_f) and rise/fall output slews (S_r/S_f) are crucial in modelling MIS effects. Reducing $|t_D|$ can result in shorter delays and output slews, resulting in speed-up [19]. As illustrated in Fig. 4(a) 4(d), the speed-up is observed in a 3-input AOI21 gate when temporal distance approaches zero. The reduction in $|t_D^{AC}|$ leads to significant decreases in timing

attributes: τ_r and S_r. The delay and output slew exhibit a characteristic V-shaped delay profile resulting from the MIS effect.

While simultaneous transitions across all inputs are theoretically possible, such events are statistically less likely [10]. Consequently, this paper analyzes the pairwise temporal distances between input signals for gates with more than two inputs. For example, in a three-input gate with input pins A, B, and C, the relevant pairwise temporal distances considered are between $A-B$ (t_D^{AB}), $B-C$ (t_D^{BC}), and $A-C$ (t_D^{AC}). These pairwise simultaneous arrivals of signals are more likely to occur than the simultaneous arrival of all three input signals. By focusing on these pairwise signal arrivals, which have a significant impact on delay variations, we can effectively model MIS-induced delay variations using a simplified approach that emphasizes the timing overlaps most likely to trigger MIS effects. It is worth pointing out that in the unlikely event of three or more inputs switching near-simultaneously, the proposed framework would need to be extended to handle those cases.

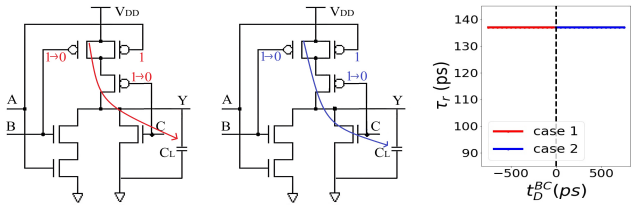
Besides t_D, other parameters, such as input slew (denoted as S_i for the input pin i) and output load capacitance (denoted as C_L) are essential in modeling MIS effects. Higher values of input slews S_A, S_B, S_C and increased load capacitance C_L amplify the impact of MIS, as demonstrated in Fig. 4(a) through 4(d).

B. Delay Profile in Complex Gates

Previous studies have demonstrated that MIS-induced delay variations occur when two signals switch simultaneously [19], [20], [10], [21]. However, in complex gates, the simultaneous switching of two signals results in three distinct types of delay profiles as t_D varies. We illustrate this phenomenon using a three-input AOI gate, where $Y = \neg(A.B + C)$.

1) SIS-like behavior: Assume that initially $A = 1$, $B = 1$, and $C = 1$, which forces the output $Y = 0$. If C is held to 1, the output will be forced to 0 regardless of the values at A and B, indicating that the output is not sensitive to A and B, unless C toggles to 0. Similarly, if $A = 1$ and $B = 1$ are fixed, the output will be forced to 0 regardless of the value of C, indicating the output is not sensitive to changes in C until A or B toggles.

Next, consider the transition: $B = 1 \to 0$ and $C = 1 \to 0$, as shown in Fig. 5(a). The output is expected to transition to 1. However, this transition is expected to happen when both inputs toggle: a single input toggle is insufficient to change the output. Hence, the delay is determined by the input that arrives later. In Fig. 5(a), for $t_D^{BC} \ll 0$ (red curve in Fig. 5(c)), C arrives early and can be treated as settled to 0 by the time B arrives. Similarly, in Fig. 5(b), for $t_D^{BC} \gg 0$ (blue curve in Fig. 5(c)), B arrives early and can be treated as settled to 0 by the time C arrives. In both cases, the output capacitor is charged through the same path (two PMOS transistors in series). Moreover, between these extremes of temporal distance, the charging path remains the same. Consequently, the observed delay does not show variation with t_D^{BC}. In effect, despite simultaneous switching, the behavior is SIS-like.



(a) (b) (c)

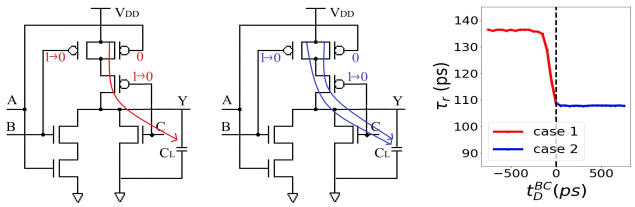
Fig. 5: SIS-like behavior: (a) charging current when B arrives late (b) charging current when C arrives late (c) delay profile.

The behavior described above arises when the output is not sensitive to either input individually, but can toggle when both inputs change together through a series path in the CMOS transistor stack. Since the path remains the same for both $t_D < 0$ and $t_D > 0$, no V-shaped delay profile is observed. This type of transition is already modeled in the traditional library as SIS arcs (E.g., $A = 1$, $B = 1 \to 0$, $C = 0$). Therefore, we do not consider it for modeling MIS effects in this paper.

2) State-Dependent delay behavior (SD): Assume that initially $A = 0$, $B = 1$, and $C = 1$, which forces the output $Y = 0$. If $A = 0$ and $B = 1$ are fixed, the output is not forced to 0 or 1, but determined by the value of C, indicating the output is sensitive to changes in C. However, if C is held to 1, the output will be forced to 0 regardless of the values at A and B, indicating that the output is not sensitive to A and B.

For a given set of input values, the input that solely determines the output value is referred to as the controlling input (C in this case), while the inputs that do not influence the output are termed non-controlling inputs (A and B in this case).

Next, consider the transition: $B = 1 \to 0$ and $C = 1 \to 0$, as shown in Fig. 6(a). The output is expected to transition to 1. This transition is expected to happen when C makes the transition: the transition at B is not critical because the output is not sensitive to B. Hence, the delay is primarily determined by the arrival time of the signal at the controlling input, i.e., C. However, the values at the non-controlling inputs decide the paths through which the output load capacitor is charged. In Fig. 6(a), for $t_D^{BC} \ll 0$ (red curve in Fig. 6(c)), B arrives later and can be treated as 1 (having old value) when C arrives. Hence, in this case, the output capacitor is charged through one path, as shown in Fig. 6(a), and we observe a greater



(a) (b) (c)

Fig. 6: State-dependent behavior: (a) charging current when B arrives late (b) charging current when C arrives late (c) delay profile.

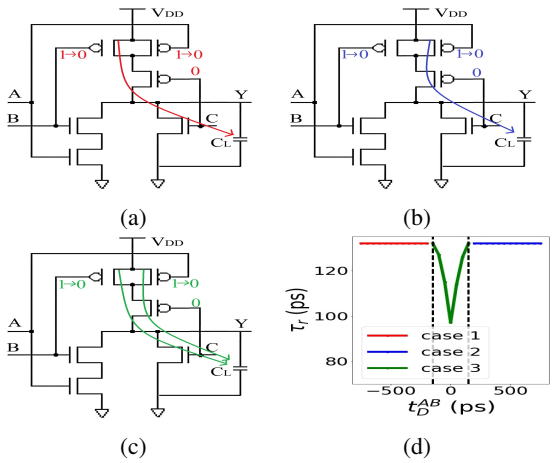


Fig. 7: MIS-induced speed-up: (a) charging current when A arrives late (b) charging current when A and B arrives together (c) charging current when B arrives late (d) delay profile.

delay value. However, in Fig. 6(b), for $t_D^{BC} \gg 0$ (blue curve in Fig. 6(c)), B arrives early and can be treated as settled to 0 by the time C arrives. Hence, in this case, the output capacitor is charged through two paths, as shown in Fig. 6(b), and we observe a smaller delay value. In between these extremes of temporal distance, we observe an intermediate delay when the two charging paths are enabled partially during the output transition. Hence, we observe a step-like delay profile.

The behavior described above occurs when the output is sensitive to one input while being insensitive to the other input. However, toggling non-controlling inputs can enable or disable specific paths in the CMOS transistor implementation, resulting in distinct delay values. This type of delay behavior is traditionally modeled in libraries as state-dependent conditional SIS arcs (E.g., $A = 0$, $B = 1$, $C = 1 \rightarrow 0$ and $A = 0$, $B = 0$, $C = 1 \rightarrow 0$). In Liberty format, these arcs are modeled separately using distinct "when" conditions. For simultaneous switching, the delay remains bounded by these two state-dependent delay values. As a result, STA tools select the more pessimistic value, and timing analysis will not yield false positives. Therefore, we do not consider this type of transition for modeling MIS effects in this paper.

3) MIS-induced speed-up: Assume that initially $A = 1$, $B = 1$, and $C = 0$, which forces the output $Y = 0$. If $B = 1$ and $C = 0$ are fixed, the output is not yet forced to 0 or 1, but determined by the value of A, indicating the output is sensitive to changes in A. Similarly, if $A = 1$ and $C = 0$ are fixed, the output is not yet forced to 0 or 1, but determined by the value of B, indicating the output is sensitive to changes in B. Thus, the output is sensitive to both A and B for the above input combination.

Next, consider the transition: $A = 1 \rightarrow 0$ and $B = 1 \rightarrow 0$, as shown in Fig. 7(a). The output is expected to transition to 1. This transition is expected to happen when either A or B toggles because the output is sensitive to both. In Fig. 7(a), for $t_D^{AB} \ll 0$ (red curve in 7(d)), B arrives early and A can be treated as 1 (having old value). Hence, in this case, the output capacitor is charged through one path, as shown in Fig. 7(a). Similarly, for $t_D^{AB} \gg 0$ (blue curve in in 7(d)), A arrives early and B can be treated as 1 (having old value). Hence, in this case, the output capacitor is charged through one path, as shown in Fig. 7(b). However, when $|t_D^{AB}|$ approaches zero, the output capacitor gets charged through two paths, as shown in Fig. 7(c). Hence, when both transitions occur simultaneously, the delay decreases significantly, and we observe the characteristics MIS V-shaped delay profile (green curve in Fig. 7(d)).

The behavior described above occurs when the output is sensitive to both the inputs that are switching simultaneously. In such cases, the simultaneous toggling of both inputs activates multiple paths, causing the output capacitance to charge faster, thereby reducing delay. In traditional libraries, these transitions are modeled as separate SIS arcs. However, the speed-up caused by simultaneous switching results in delays significantly lower than those characterized by individual arcs. Consequently, even though STA tools select pessimistic values, this mismatch can lead to false positives and potential circuit failures. Therefore, in this paper, we focus on identifying and modeling this type of transition in MIS-augmented libraries.

The above observations on the pattern-dependent delay impact of simultaneous switching suggest that logic analysis can be effectively employed to model MIS effects. In the next section, we present the MIS pattern extraction framework, which leverages logic techniques to identify transitions expected to produce the MIS-induced speed-up.

IV. MIS Pattern Extraction

A. Notations and Problem Definition

We represent the logical functionality of a complex logic gate as:

$$Y = f(X)$$

where Y is the output and $X = \{x_1, x_2, ..., x_N\}$ are the N inputs of the logic gate. We represent a transition at the input as $X' \rightarrow X''$ where the input changes from $X' = \{x_1', x_2', ..., x_N'\}$ to $X'' = \{x_1'', x_2'', ..., x_N''\}$. The MIS pattern extraction algorithm needs to find all $X' \rightarrow X''$ transitions that satisfy the following properties:

1) Only a pair of inputs, x_i and x_j, toggle their values. This restriction arises because this work focuses exclusively on pairwise MIS, although, in theory, more than two inputs could switch simultaneously.
2) The output Y should toggle its value. An input transition is interesting from the timing perspective only when the output transitions. Transitions that cause only intermediate glitches at the output without leading to a final output change are ignored.
3) The output must be sensitive to both toggling inputs, x_i and x_j. As explained in the previous section, when transitions at both inputs directly influence the output, the characteristic MIS-induced V-shaped delay profile is observed, with each transition dominating one arm of the V.

Next, we encode the above properties mathematically, which will enable the SAT solver to be employed and automatically extract the MIS-relevant patterns.

B. Mathematical Formulation

For a given transition $X' \rightarrow X''$, only a pair of inputs, x_i and x_j, toggle their values if the following function evaluates to 1:

$$M(X', X'', x_i, x_j) = (x_i' \oplus x_i'').(x_j' \oplus x_j''). \prod_{k \neq i,j} [\neg(x_k' \oplus x_k'')] \tag{1}$$

The first two terms represent the XOR of the pre- and post-switching values for x_i and x_j, ensuring they toggle their values. The product $\prod_{k \neq i,j}$ is the logical AND over all inputs other than x_i and x_j. By taking the XNOR of their pre- and post-switching values, we ensure that all other inputs do not toggle their values.

For a given transition $X' \rightarrow X''$, the output toggles if the following evaluates to 1:

$$T(X', X'') = f(X') \oplus f(X'') \tag{2}$$

The XOR of pre- and post-switching values at the output ensures that the input transition causes the output to change, and is interesting for delay calculation.

To describe the logical property of MIS patterns, we use the notion of Boolean difference. The Boolean difference of Y with respect to a variable x_i is the XOR of the negative and positive cofactors of Y with respect to x_i and can be expressed as:

$$\frac{\partial Y}{\partial x_i} = f(x_1, x_2, ..., x_{i-1}, 0, x_{i+1}, ..., x_N) \oplus f(x_1, x_2, ..., x_{i-1}, 1, x_{i+1}, ..., x_N) \tag{3}$$

Evidently, $\frac{\partial Y}{\partial x_i}$ is independent of x_i. For a given assignment of variables (other than x_i), the Boolean difference $\frac{\partial Y}{\partial x_i}$ evaluates to 1 if Y depends on x_i, else it evaluates to 0. In other words, if $\frac{\partial Y}{\partial x_i}$ evaluates to 1, when we toggle the value of x_i the output Y is expected to toggle. Hence, Boolean difference can capture the sensitivity of a function with respect to a given variable.

For a given input value X', the output will be sensitive to the change in x_i and x_j, if the following expression evaluates to 1:

$$S(X', x_i, x_j) = \frac{\partial Y}{\partial x_i}.\frac{\partial Y}{\partial x_j}\bigg|_{X=X'} \tag{4}$$

Here, we take the logical AND of two terms in the RHS. The first term ensures that the output is affected when x_i toggles, while the second term ensures the impact of x_j on the output. Together, these terms guarantee that the output is influenced by the transitions of both inputs. The extent of impact due to simultaneous transitions (relative to a single transition) will depend on the transistor-level implementation of Y, input slews, and output load, and we cannot estimate this based on logical analysis. However, the above condition filters out transitions that are not expected to show MIS-induced delay variations and simplifies the characterization task significantly.

The input transitions that are of interest for MIS modelling should satisfy all three conditions mentioned in the previous sub-section. Hence, combining Eq. 1, 2, and 4, for an MIS transition $X' \rightarrow X''$ (with x_i and x_j toggling), the following function should evaluate to 1:

$$M(X', X'', x_i, x_j).T(X', X'').S(X', x_i, x_j) \tag{5}$$

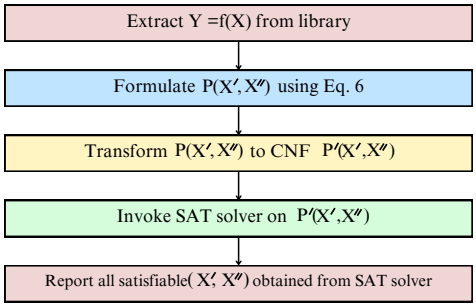


Fig. 8: Framework for MIS pattern extraction

Here, each term M, T, and S is Boolean (evaluates to either 0 or 1). The MIS pattern extraction algorithm is expected to extract all patterns that satisfy the above condition. Hence, all (X', X'') pairs that satisfy the following condition are the MIS-specific transitions:

$$P(X', X'') = \sum_{(i,j)} M(X', X'', x_i, x_j).T(X', X'').S(X', x_i, x_j) \tag{6}$$

Here, the summation $\sum_{(i,j)}$ is a logical OR over all unordered input pairs (i, j).

C. Algorithm

Fig. 8 shows the framework for MIS pattern extraction. The Boolean function $Y = f(X)$ is extracted for a given logic gate from the traditional library. A logic expression $P(X', X'')$ is constructed using $f(X)$ and Eq. 6. Note that if $f(X)$ contains N Boolean variables, $P(X', X'')$ will contain $2 \times N$ Boolean variables: N variables for the initial state X' and N variables for the final state X''.

Subsequently, $P(X', X'')$ is transformed to its Conjunctive Normal Form (CNF), $P'(X', X'')$, using Tseitin transformation, and given to a SAT solver in a suitable format [25]. In this work, we have used MiniSat, though any other efficient SAT solver can be employed to extract the MIS pattern [22]. We allow the SAT solver to identify and report all possible satisfying assignments for the given SAT problem. The results obtained by the SAT solver are subsequently decoded to extract all MIS patterns (X', X''). Since $f(X)$ are typically simple functions in the libraries and contain only a few variables, the problem presented is quite easy for state-of-the-art SAT solvers. Hence, the runtime needed in MIS pattern extraction is negligible compared to other tasks in creating MIS-augmented libraries.

D. Illustration

We present the results obtained using the above algorithm and validate it using SPICE simulations for an AOI gate represented as $Y = \neg(A.B + C)$ and an OAI gate represented as $Y = \neg((A + B).C)$. Each gate has 30 possible input transitions (denoted as $X' = A'B'C' \rightarrow X'' = A''B''C''$) that can toggle the output Y. Among these transitions, 14 involve simultaneous switching of two inputs: 7 leading to rise transitions and 7 to fall transitions. For each case, we vary t_D (between $A - B$, $B - C$, and $A - C$) in the range -750 ps

TABLE I: Results of MIS Pattern Extraction (Proc. = tt, Volt. = 0.7 V, $T = 25°C$, C_L = 25 fF, $S_A = S_B = S_C$ = 160 ps.)

(a) AOI: Rising

X'	X''	Delay Profile	MIS effect?	Extracted?
110	000		Yes	Yes
011	000		No (SD)	No
001	010		No (SD)	No
111	100		No	No
101	000		No (SD)	No
111	010		No	No
001	100		No (SD)	No

(b) AOI: Falling

X'	X''	Delay Profile	MIS effect?	Extracted?
000	110		No	No
000	011		No	No
010	001		No	No
100	111		Yes	Yes
000	101		No	No
010	111		Yes	Yes
100	001		No	No

(c) OAI: Rising

X'	X''	Delay Profile	MIS effect?	Extracted?
111	001		No	No
011	000		Yes	Yes
111	100		No	No
101	110		No	No
101	000		Yes	Yes
111	010		No	No
011	110		No	No

(d) OAI: Falling

X'	X''	Delay Profile	MIS effect?	Extracted?
001	111		Yes	Yes
000	011		No	No
100	111		No (SD)	No
110	101		No (SD)	No
000	101		No	No
010	111		No (SD)	No
110	011		No (SD)	No

to +750 ps and observe the rise delay (τ_r) or fall delay (τ_f) profiles using SPICE simulations.

The results are summarized in Tab. I (a-d) separately for each gate and output transition type. Each row corresponds to a transition represented by $X' \rightarrow X''$. The third column presents the delay profile for varying t_D obtained using SPICE simulation. The V-type profile indicates delay variations due to the MIS effect and is marked in the fourth column ("SD" implies that variations are due to state-dependent delay changes). The last column indicates whether the MIS pattern extractor identified the corresponding transition $X' \rightarrow X''$ using the algorithm described above. Notably, the results produced by the MIS pattern extractor align with those obtained from SPICE simulations (as shown in the last two columns), validating the effectiveness of the proposed technique for deriving MIS-specific patterns.

V. Model Development

For each pattern identified by the MIS pattern extractor, we develop a delay model quantifying MIS-induced delay variations by leveraging ML, as described in the following paragraphs.

A. Dataset generation and optimization

The dataset is generated using SPICE simulations for the given two pins i and j for a combinational gate. The input slews (S_i and S_j), load capacitance (C_L), and temporal distance (t_D^{ij})) are treated as features and τ_r/S_r and τ_f/S_f are treated as labels. For the features S_i, S_j, and C_L, the starting value (x_{start}) and ending value (x_{end}) are taken from the existing standard cell library. Intermediate values are generated based on a geometric progression (GP), given by the following equation: $x_n = a \cdot r^{n-1}$, where x_n represents the n-th value in the progression, $a = x_{\text{start}}$ is the starting value, and r is a

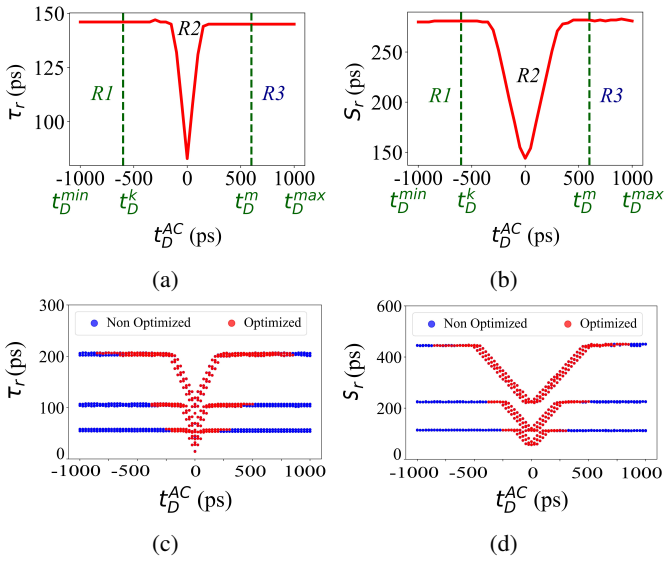


Fig. 9: (a, b) Definition of regions R1, R2, and R3 for delay and slew data collection (c, d) Effectiveness of optimization strategy for delay and slew.

user-defined ratio. The progression continues until x_n exceeds x_{end}. This method generates a desired number of intermediate values between x_{start} and x_{end}, with the control parameter r determining the spread of values. For t_D an arithmetic progression is used to uniformly distribute values between a user-defined range (minimum t_D^{min} and maximum t_D^{max}):

$$t_D^{(i)} = t_D^{min} + (i-1) \cdot \Delta t_D \tag{7}$$

where $t_D^{(i)}$ denotes the i-th temporal distance value and Δt_D is the step size given as follows:

$$\Delta t_D = \frac{t_D^{max} - t_D^{min}}{M} \tag{8}$$

The total number of divisions M is appropriately selected to ensure that the variations in t_D adequately cover the MIS effects. For each combination S_i, S_j, C_L, and t_D^{ij} simulations are performed and resulting τ_r, S_r, τ_f, and S_f are stored in an output file for further processing and analysis.

To reduce characterization efforts, we also employ a data optimizer algorithm as follows. From Fig. 9(a),(b) we can infer that the delay profile can be divided into three distinct regions R1, R2, and R3: regions R1 and R3 have negligible variation in delay, while region R2 (around $t_D^{AC} = 0$) has significant delay variations. The algorithm aims to identify the boundaries between these regions in the range $[t_D^{min}, t_D^{max}]$, and focuses on collecting data for region R2 while avoiding simulations in regions R1 and R3. The optimization algorithm explores t_D in the range $[t_D^{min}, 0]$ starting at $t_D = 0$ with a step size of $-\Delta t_D$. However, if for three successive values of t_D, no variation in delay and output slew is noticed, we mark the corresponding value of t_D as t_D^k, the boundary between regions R1 and R2. Similarly, we determine t_D^m, the boundary between regions R2 and R3, by varying t_D in the range $[0, t_D^{max}]$. For $t_D \in [t_D^{min}, t_D^k]$ and $t_D \in [t_D^m, t_D^{max}]$, delay and output slews do not exhibit significant MIS-induced variations, and therefore no simulations are performed, significantly reducing the characterization and model training workload while still ensuring thorough coverage of the delay behavior due to MIS. Fig. 9(c),(d) illustrate the effectiveness of the proposed strategy for delay and output slew.

B. Model training and testing

The dataset $D = \{X, y\}$, generated by a data generator and optimizer, undergoes preprocessing before being utilized for model training and testing. Here, $X \in \mathbb{R}^{n \times m}$ represents the feature matrix, where n is the number of samples and m is the number of features, while $y \in \mathbb{R}^n$ represents the corresponding target values. The dataset D undergoes min-max scaling to map all features and targets to the range $[0, 1]$. This scaling ensures that all features are comparable, preventing any feature with a larger magnitude from disproportionately influencing the model. After scaling, the dataset is divided into a training set $D_{\text{train}} = \{X_{\text{f,train}}, y_{\text{t,train}}\}$ and a testing set $D_{\text{test}} = \{X_{\text{f,test}}, y_{\text{t,test}}\}$.

Various ML models, varying in complexity and data representation requirements, are developed using different learning approaches. More complex models generally require more computational resources for training, but they can achieve higher accuracy if overfitting is carefully controlled. Additionally, optimizing hyperparameters is crucial to enhance model performance. Methods like grid search are employed to identify the best set of hyperparameters for the chosen model.

To evaluate model performance efficiently and reduce the risk of overfitting, K-fold cross-validation is used. The training dataset D_{train} is divided into k equally sized disjoint folds, denoted as $\{D_1, D_2, \ldots, D_k\}$. In each fold, i, the model is trained on the other $(k-1)$ folds and validated on the i-th fold. This process is repeated k times, with each fold serving as the validation set exactly once [24].

The model's performance is assessed using the Normalized Root Mean Squared Error (NRMSE). First, the Root Mean Squared Error (RMSE) is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2} \tag{9}$$

where N is the number of samples in the test set, y_t is the true target value, and $\hat{y}_t$ is the predicted value. The NRMSE is then computed as:

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{\text{max}} - y_{\text{min}}} \tag{10}$$

where y_{max} and y_{min} are the maximum and minimum values of y_t, respectively.

VI. Augmenting Library with MIS annotation

The ML model created by LiMo needs to be integrated with the STA tools. We propose augmenting the existing SIS-based libraries in Liberty format [1] with the MIS information. In Liberty format, the delay and the output slew information are annotated on timing arcs separately for each transition. A timing arc consists of an input pin i and the output pin y. We propose to model the MIS effect by adding another input pin j to the timing arc if simultaneous switching of i and j produces MIS-induced delay variations (as inferred by the MIS pattern extractor). Additionally, we associate the corresponding MIS-specific ML model with this input pin j. An STA tool can extract this information from the augmented technology library and utilize it to compute MIS-aware timing attributes (τ_r, S_r, τ_f, and S_f) using circuit conditions such as the input slews S_i, S_j, temporal distance t_D^{ij}, and output load C_L.

As an illustration, consider the three-input AOI gate shown in Tab. I. For the rising output, the MIS effect is observed for the following transition: $\{A = 1, B = 1, C = 0\} \rightarrow \{A = 0, B = 0, C = 0\}$. The SIS-based library contains an arc $A \rightarrow Y$ assuming $B = 1, C = 0$, as shown below.

```
pin(Y) {
  timing() {
    related_pin : "A";
    cell_rise(delay_template) {
        ...
    }
  }
}
```

We augment this arc in the library with the MIS-related information, as shown below.

```
pin(Y) {
  timing() {
    related_pin : "A";
    cell_rise(delay_template) {
        ...
        mis_info("AB_11_00") {
           mis_pin : "B";
           mis_model : "reference_to_mis_model";
        }
    }
  }
}
```

TABLE II: Performance Analysis of ML Models

Gate	DT			RF			GB			ANN		
	Train	**Val**	**Test**	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**
NAND-2i/p	0.0360	0.0398	0.0450	0.0360	0.0416	0.0482	0.0345	0.0378	0.0425	0.0305	0.0319	0.0320
AND-2i/p	0.0281	0.0280	0.0310	0.0201	0.0237	0.0218	0.0242	0.0289	0.0257	0.0238	0.0250	0.0244
NOR-2i/p	0.0263	0.0294	0.0266	0.0260	0.0295	0.0300	0.0212	0.0221	0.0223	0.0225	0.0228	0.0225
OR-2i/p	0.0321	0.0350	0.0368	0.0304	0.0335	0.0342	0.0290	0.0311	0.0323	0.0281	0.0295	0.0302
NAND-3i/p	0.0301	0.0343	0.0339	0.0295	0.0327	0.0351	0.0313	0.0330	0.0339	0.0332	0.0337	0.0328
AND-3i/p	0.0315	0.0340	0.0367	0.0297	0.0321	0.0340	0.0289	0.0304	0.0311	0.0300	0.0314	0.0311
NOR-3i/p	0.0330	0.0345	0.0372	0.0312	0.0342	0.0361	0.0328	0.0339	0.0363	0.0342	0.0356	0.0336
OR-3i/p	0.0342	0.0360	0.0385	0.0320	0.0350	0.0374	0.0335	0.0348	0.0371	0.0324	0.0339	0.0345
AOI21-3i/p	0.0308	0.0370	0.0317	0.0277	0.0300	0.0325	0.0242	0.0255	0.0272	0.0287	0.0290	0.0288
OAI-3i/p	0.0318	0.0342	0.0371	0.0292	0.0324	0.0355	0.0277	0.0293	0.0315	0.0286	0.0298	0.0298
Average			**0.0354**			**0.0344**			**0.0319**			**0.0299**

The simultaneously switching pin ("mis_pin") and corresponding ML model ("mis_model") are defined as named attributes in the group "mis_info" on the associated SIS arc $A \rightarrow Y$. An STA tool can utilize this information to appropriately adjust the corresponding SIS-based timing attributes, thereby accounting for MIS-induced delay variations. If a timing arc does not have MIS-specific augmentation, it signifies that MIS-induced delay variations are not applicable to that arc, and the existing delay model is adequate. Thus, the proposed approach utilizes MIS-specific models only when required, maintaining the efficiency of a traditional SIS-based model for non-MIS scenarios. It is worth pointing out that the "mis_info" group and the above named attributes are not supported by standard Liberty and EDA tool support would require parsing and comprehending them.

VII. Results

We performed experiments on standard cells implemented using the ASAP 7nm process design kit (PDK) [26]. We used industry-standard tools Tempus and Virtuoso to gather these results [5], [23]. First, we present the results of MIS model creation, followed by the results of applying the model for timing analysis.

A. MIS Model Creation

We conducted experiments with various types of ML models, explored their hyperparameters, and applied optimization strategies to reduce computational effort.

1) Comparative Analysis of ML models: Tab. II presents the performance analysis of several ML models— Decision Trees (DT), Random Forests (RF), Gradient Boosting (GB), and Artificial Neural Networks (ANN)—applied to different logic gates. The performance is evaluated across three datasets: training, validation, and test sets. For each model, the best configuration was selected after hyperparameter tuning, ensuring that the models did not suffer from either underfitting or overfitting. This tuning process involved monitoring both training and validation errors, ensuring they remained comparable, which suggests that the models generalize well to unseen data. As shown in the table, there is no significant gap between training, validation, and test errors, indicating that the models are well-optimized.

The ANN model outperforms other models with an average testing error of 0.0299 across all datasets, achieving consistently lower errors. While other models show similar errors across various gates, the ANN is preferred for its superior average error and reliable performance across all gates. One of the key advantages of the ANN over tree-based models is its high adaptability and scalability to handle large datasets, offering greater flexibility for diverse standard cell scenarios.

TABLE III: Hyperparameter Tuning for ANN (NAND-2i/p)

Hidden Layers	**Test NRMSE**	**Hidden Layers**	**Test NRMSE**
(10, 10)	0.0919	(10, 10, 10)	0.0857
(10, 10, 10, 10)	0.0786	(10, 10, 10, 10, 10)	0.0919
(50, 50)	0.0685	(50, 50, 50)	0.0570
(50, 50, 50, 50)	0.587	(50, 50, 50, 50, 50)	0.0590
(100, 100)	0.0447	(100, 100, 100)	0.0364
(100, 100, 100, 100)	0.0438	(100, 100, 100, 100, 100)	0.0469
(150, 150)	0.0388	**(150, 150, 150)**	**0.0320**
(150, 150, 150, 150)	0.0355	(150, 150, 150, 150, 150)	0.0359
(250, 250)	0.0395	(250, 250, 250)	0.405
(250, 250, 250, 250)	0.0339	(250, 250, 250, 250, 250)	0.0355
(500, 500)	0.0347	(500, 500, 500)	0.0389
(1000, 1000)	0.0449	(1000, 1000, 1000)	0.0466
(100, 50, 10)	0.0550	(250, 150, 100, 50)	0.0516
(500, 250, 150, 100)	0.0363	(1000, 500, 100)	0.0359

2) Hyperparameter Tuning for ANN: Tab. III presents the NRMSE values obtained from hyperparameter tuning of the ANN model for a two-input NAND gate for different hidden layer configurations. Hyperparameter tuning is crucial in optimizing model performance by selecting the best configurations to minimize prediction errors. In this study, grid search was used to assess different hyperparameter combinations, such as the number of neurons, activation functions, optimizers, learning rates, regularization, and batch sizes. The ReLU activation function and Adam optimizer were selected, with a learning rate of 0.01 and an L2 regularization parameter of 0.01 applied. From the Tab. III, the optimal architecture was identified as having three hidden layers, each with 150 neurons, which yielded the lowest test NRMSE of 0.0320. Interestingly, increasing the complexity of the architecture—such as adding more neurons (1000, 1000) or (1000, 1000, 1000), or more hidden layers (150, 150, 150, 150) or (150, 150, 150, 150, 150)—did not result in improved NRMSE. This suggests that simpler models are sufficient to capture the complex

TABLE IV: Impact of Dataset Optimization

Without Optimization				
Gates	# Total Data	# Train Data	# Test Data	NRMSE
NAND-2i/p	6464	5171	1293	0.0320
AND-2i/p	6464	5171	1293	0.0244
NOR-2i/p	6464	5171	1293	0.0225
OR-2i/p	6464	5171	1293	0.0302
NAND-3i/p	19392	15513	3879	0.0328
AND-3i/p	19392	15513	3879	0.0311
NOR-3i/p	19392	15513	3879	0.0336
OR-3i/p	19392	15513	3879	0.0345
AOI21-3i/p	19392	15513	3879	0.0288
OAI-3i/p	19392	15513	3879	0.0298
Average	**1,42,208**	**1,13,762**	**28,446**	**0.0299**
With Optimization				
Gates	# Total Data	# Train Data	# Test Data	NRMSE
NAND-2i/p	2536	2536	1293	0.0344
AND-2i/p	1655	1655	1293	0.0247
NOR-2i/p	2745	2745	1293	0.0260
OR-2i/p	1314	1314	1293	0.0326
NAND-3i/p	6682	6682	3879	0.0390
AND-3i/p	3632	3632	3879	0.0294
NOR-3i/p	6606	6606	3879	0.0321
OR-3i/p	3941	3941	3879	0.0368
AOI21-3i/p	7426	7426	3879	0.0320
OAI-3i/p	6837	6837	3879	0.0317
Average	**43,374**	**43,374**	**28,446**	**0.0318**

relationships in the data. Hence, balancing model complexity and predictive accuracy is crucial for MIS model creation. Similar hyperparameter tuning was performed for other gates, yielding similar insights into the optimal configurations.

3) Optimized Gate Level Evaluation: Tab. IV compares the performance of models with and without dataset optimization across various logic gates. The $NRMSE$ obtained using the optimized training dataset shows a comparable error to that of the model trained with a large, unoptimized dataset. However, the optimized dataset contains significantly fewer data points, substantially reducing the characterization effort, which is the most time-consuming task in creating library models. This reduction also decreases the model training runtime. Therefore, the dataset optimization strategy is crucial in modeling MIS effects efficiently. It is important to point out that the test dataset remains consistent across both models for a fair comparison.

B. Circuit-Level Model Evaluation

To assess the efficacy of the proposed methodology at the circuit level, we use the evaluation framework illustrated in Fig. 10. First, we performed STA to extract key timing attributes (S_i (max), S_j (max), AT_i(max), AT_j(max), and C_L) from Tempus [5] which were then utilized in SPICE simulations and the trained ML model. The STA tool provides SIS-based timing attributes T_{STA} (rise delay τ_r^{SIS}, rise output slew S_r^{SIS}, fall delay τ_f^{SIS}, and fall output slew S_f^{SIS}). The SPICE simulations provide actual values for these timing attributes, denoted as A_{SPICE} (τ_r^{At}, S_r^{At}, τ_f^{At}, and S_f^{At}). The trained ML model generates predicted values for these attributes, denoted as P_{ML} (τ_r^{Pred}, S_r^{Pred}, τ_f^{Pred}, and S_f^{Pred}).

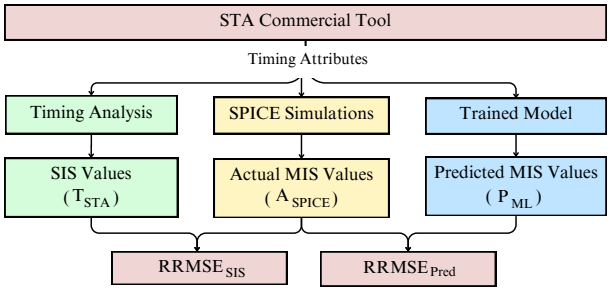


Fig. 10: Model Evaluation Framework

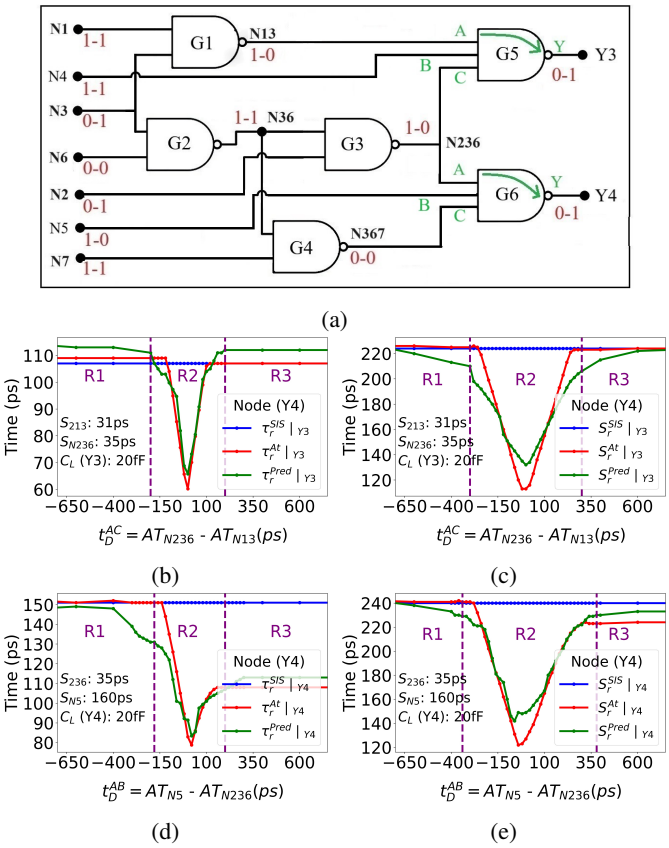


Fig. 11: (a) ISCAS modified benchmark circuit C17 (added 3 input NAND gates) [27] (b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models.

To quantitatively evaluate the performance of the proposed methodology, we used the Relative Root Mean Square Error (RRMSE) which is defined as follows:

$$RRMSE_{SIS} = \left(\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(A_{\text{SPICE},i} - T_{\text{STA},i}\right)^2}}{\frac{1}{N}\sum_{i=1}^{N} A_{\text{SPICE},i}} \right) \quad (11)$$

$$RRMSE_{Pred} = \left(\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(A_{\text{SPICE},i} - P_{\text{ML},i}\right)^2}}{\frac{1}{N}\sum_{i=1}^{N} A_{\text{SPICE},i}} \right) \quad (12)$$

where N represents the total number of samples.

TABLE V: Evaluation on 3-Input Gate (for region $R2$)

Metric	$Y3$		$Y4$		Average
	τ_r	S_r	τ_r	S_r	
$RRMSE_{SIS}$	18.41%	59.01%	39.70%	59.01%	**44.03%**
$RRMSE_{Pred}$	4.03%	6.72%	7.24%	6.72%	**6.17%**

1) Evaluation on 3-Input NAND Gate: We evaluate the accuracy of the proposed ML model on the circuit shown in Fig. 11(a). We consider MIS effects in the three-input NAND gates G5 and G6, driving nets Y3 and Y4, respectively.

For a three-input NAND gate $Y = \neg(A.B.C)$, the MIS pattern extractor produces the following MIS transitions: $111 \rightarrow 001$, $111 \rightarrow 100$, and $111 \rightarrow 010$. LiMo generates ML models to capture MIS effects for these transitions and augments the SIS-based library with this information. We present results obtained using these ML models in the following paragraphs.

We adjust the arrival time of the signals at the primary inputs such that, for gate G5, the pins A (net N13) and C (net N236) switch simultaneously, while the pin B (net N4) is held constant. For gate G6, the pins A (net N236) and B (net N5) switch simultaneously, while the pin C (net N367) is held constant.

Fig. 11(b) and 11(c) compare the results obtained using the SIS model, SPICE and the proposed ML model for the rise delay and output slew, respectively, for the gate G5 (port Y3). Similarly, Fig. 11(d) and 11(e) compare them for the gate G6 (port Y4). We observe that, in general, the ML model is able to match the SPICE simulation in the region R2, in which the MIS effect is dominant. To compare quantitatively, the RRMSE is shown in Tab. V for SIS and the proposed model. We observe that, on average, if we ignore MIS effects, we incur an error of 44.03 % in the timing attributes, while the proposed model can predict them with an error of 6.17 %.

TABLE VI: Evaluation on Complex Gate (for region $R2$)

Metric	$Y5$		$Y6$		Average
	τ_f	S_f	τ_r	S_r	
$RRMSE_{SIS}$	43.54%	85.62%	17.73%	24.30%	**43.29%**
$RRMSE_{Pred}$	3.88%	4.36%	3.79%	4.80%	**4.20%**

2) Evaluation on Complex Gate: We evaluate the accuracy of the proposed ML model on the circuit shown in Fig. 12(a). We consider MIS effects in the three-input AOI gates G5 and G6, driving nets Y5 and Y6, respectively.

For an AOI gate, as discussed earlier, LiMo generates ML models to capture MIS effects for transitions extracted from the MIS pattern extractor and augments the SIS-based library with this information. We present results predicted using these ML models in the following paragraphs.

For gate G5, the pins A (net N13) and C (net N236) switch simultaneously, while the pin B (net N4) is held constant. Hence, the model corresponding to $010 \rightarrow 111$ is invoked for computing timing attributes. For gate G6, the pins A (net N236) and B (net N5) switch simultaneously, while the pin C (net N367) is held constant. Hence, the model corresponding to $110 \rightarrow 000$ is invoked for computing timing attributes.

Fig. 12(b) and 12(c) compare the results obtained using the SIS model, SPICE and the proposed ML model for the rise delay and output slew, respectively, for the gate G5 (port Y5).

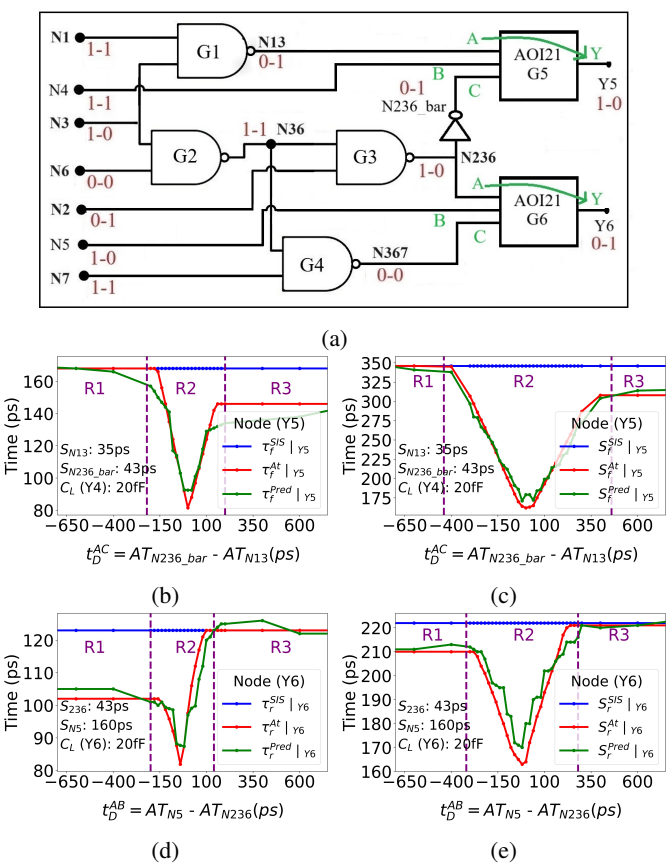


Fig. 12: (a) ISCAS modified benchmark circuit C17 (added 3 input AOI21 gates) [27] (b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models.

Similarly, Fig. 12(d) and 12(e) compare them for the gate G6 (port Y6). We observe that, in general, the ML model is able to match the SPICE simulation in the region R2, in which the MIS effect is dominant. To compare quantitatively, the RRMSE is shown in Tab. VI for SIS and the proposed model. We observe that, on average, if we ignore MIS effects, we incur an error of 43.29% in the timing attributes, while the proposed model can predict them with an error of 4.20%.

The above results demonstrate that the proposed model effectively captures MIS-induced delay variations across a wide range of gates, transitions, temporal distances, slews, loads, and instantiating conditions. Additionally, these results highlight the significant speed-up in complex logic gates, which renders traditional SIS delay models highly unrealistic for MIS scenarios. This discrepancy can result in circuit failures due to incorrect early analysis, despite STA tools adopting conservative strategies. In contrast, the proposed model accurately captures the MIS effects, making it suitable for timing analysis in industrial designs.

C. Impact of MIS Modeling on STA Accuracy

Neglecting MIS impact during STA can lead to inaccurate delay estimation, potentially causing critical violations, such as hold time failures, to be undetected. This oversight may result in post-silicon functional errors, underscoring the importance

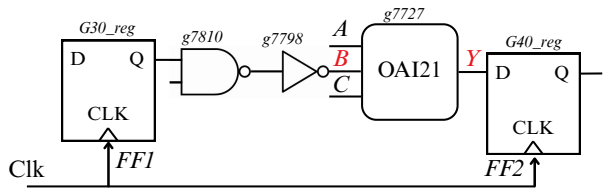


Fig. 13: Relevant portion of the circuit s1196 for hold violation

of incorporating MIS-aware delay models into STA flows. To demonstrate the practical significance of MIS modeling, we consider an example from the ISCAS'89 benchmark circuit s1196 [27] shown in Fig. 13, mapped to the ASAP 7nm PDK [26]. STA is performed using delay values extracted under the SIS model. Then, we observe OAI21 cell: input B has a slew of 7 ps, input C has a slew of 120 ps, and the output Y drives a capacitive load of 15 fF. The SIS delay for the B-to-Y arc under these conditions is 73 ps. The arrival time difference between inputs B and C is 44 ps, denoted as t_D^{BC}. Under the SIS assumption, STA computes the slack for the corresponding hold check as +2 ps, indicating no violation. However, when MIS effects are considered, the simultaneous switching activity causes a reduction in the effective delay of the OAI21 gate. SPICE simulations confirm this behavior, reporting the actual delay for the B-to-Y arc under MIS conditions as 49 ps, resulting in a revised slack of –22 ps, indicating a hold violation that was previously undetected. To address this, the proposed ML-based delay model is used, as in the library B-to-Y arc was augmented with mis_info to predict the MIS-aware delay. The model estimates the B-to-Y arc delay as 50 ps, closely matching the SPICE-derived value, and the corresponding slack is computed as –21 ps. This demonstrates that the ML-augmented library captures the MIS-induced delay variation with high accuracy and enables STA tools to identify violations that would otherwise be overlooked when using SIS-based models.

D. Evaluation of Runtime Performance

We evaluated the runtime performance of various computational tasks within the proposed framework on a Linux platform equipped with a 1.8 GHz processor, four cores, and 4 GB of RAM and summarized the results in Table VII.

While creating the library, the SPICE simulation required approximately 12 hours to generate a dataset of 43,374 data points for 10 cells shown in Table IV. The ANN model training took around 10 minutes with 43,374 data points, and the architecture consisted of three layers, each with 150 neurons. While data collection is the most time-consuming process, its runtime has been significantly reduced using various techniques discussed in the previous sections. Importantly, the data collection is a one-time process for a given library (similar to traditional library characterization), and it enables reusing the model across multiple designs, reducing the overall cost of developing MIS-aware libraries.

During inference, the ANN model predicted MIS-aware delay for 10,000 queries in approximately 0.51 seconds. In contrast, querying the same number of MIS-aware delays with SPICE simulations took around 2.8 hours. The results

TABLE VII: Evaluation of Runtime Performance

Library Creation		**MIS Delay Computation**	
Data Collection	**Model Training**	**ANN**	**SPICE**
12 hr	10 min	0.51 sec	2.8 hr

demonstrate that the proposed model is $\sim 10,000\times$ faster than SPICE simulations in inference while maintaining reasonable accuracy, making it suitable for incorporating MIS effects in STA for industrial designs.

The MIS pattern identification is based on a systematic Boolean function formulation, which is designed to accommodate logic cells with varying numbers of inputs. This makes the proposed framework inherently scalable, as it can efficiently handle both simple and complex logic cells. In the evaluation of the scalability, we considered an industry-standard cell library, which includes a total of 565 cells. Of these, 214 are combinational cells with multiple inputs, and there are 50 unique Boolean functions for these cells spanning multiple drive strengths. The MIS pattern extractor operates on Boolean functions. Therefore, only unique logic functions are required to be processed.

The MIS pattern extractor produced 506 MIS-relevant patterns in total for these 50 multi-input Boolean functions, with an overall SAT solver runtime of $\sim$ 2.64 sec. The SAT solver identifies MIS-relevant patterns efficiently, even when the complexity of the cell grows. For example, for the most complex cell in the library:

$$\overline{(C_0 \cdot C_1) + (A_0 \cdot A_1) + (B_0 \cdot B_1)}$$

, the 63 MIS-relevant patterns were extracted, and the SAT solver runtime was 0.26 sec. This demonstrates that the proposed framework is efficient enough to process the entire library.

Based on the 506 MIS-relevant patterns identified by the extractor for the 50 multi-input combinational cells, ML models are developed only for these 506 cases. In contrast, considering all possible MIS combinations would result in 5,144 cases, requiring 5,144 ML models. Thus, the proposed approach achieves nearly a $\sim 10\times$ reduction in ML model development effort. This marks a significant improvement over traditional ML-based techniques that typically require modeling every possible combination of MIS transitions per cell.

VIII. Conclusion

In this paper, we addressed the challenge of developing MIS-aware library models for complex logic gates. The proposed approach significantly reduces computational overhead by quick logical analysis to identify transitions likely to exhibit MIS-induced speed-up, eliminating the reliance on extensive SPICE simulations for most cases. Intelligent dataset generation avoids SIS regions while focusing exclusively on dip-effects to ensure accuracy. ML is leveraged to capture multi-dimensional dependencies efficiently, and multiprocessing with load balancing enhances computational efficiency. The results demonstrate that the proposed solution is well-suited for integrating MIS-aware models into modern STA flows. In this work, the STA tool and the library models

were loosely coupled for timing analysis in a circuit, with the STA output being separately read and processed by the ML models to compute delays. Future work will focus on tightly integrating the STA tool with the library models and conducting experiments on larger industrial designs to further validate the efficacy of LiMo across the entire design flow. A tight integration of the proposed methodology into commercial STA tools such as PrimeTime and Tempus would require comprehending the augmented MIS libraries and extracting the associated ML models. These tools can then incorporate the ML models into their native delay computation engines to handle MIS scenarios, adjusting SIS delays based on the MIS delay predictions. However, the most effective integration strategy would ultimately depend on the tool's internal architecture and implementation details.

Interestingly, this methodology shows promise beyond combinational logic cells. We can treat clock and data interactions in flip-flops as a form of MIS. When data transitions occur close to the active clock edge, the clock-to-Q delay becomes sensitive to the temporal distance between the clock and the data signals, causing it to increase from the nominal value or, in extreme cases, failure to latch data. In the future, creating flip-flop models, including that of multi-bit flip-flops and latches, by employing mechanisms as proposed in this work, can be explored.

A current limitation of the proposed work is that it considers only pairwise switching and does not explicitly handle three or more inputs switching simultaneously. Although such events are less probable than two-input switching, they can still occur in realistic circuits. Hence, to ensure the timing safety of a circuit for critical applications, a more detailed assessment of the likelihood and impact of these events is necessary. It is important to note that simultaneous switching of three or more inputs does not necessarily mean that the delay is influenced by all switching signals. The delay is typically governed by the arrival time of the first controlling input (for non-controlling to controlling transitions) or the last non-controlling input (for controlling to non-controlling transitions). Consequently, even in three or more input switching scenarios, the effective delay may still depend on only one or two signals (with other signals being held at state-dependent constant values), making the SIS or proposed MIS delay models potentially sufficient. In practice, while applying the proposed model for three or more input switching cases, all the pair-wise switching can be considered, and the worst-case value picked to ensure a conservative timing analysis. Nevertheless, a deeper investigation into three or more switching scenarios is needed, and future work may require extending the proposed model to handle such cases robustly.

IX. Acknowledgment

This work was partially supported by the Semiconductor Research Corporation (SRC) through task 3169.001 and by the University Grants Commission (UGC) under the Senior Research Fellowship scheme.

References

[1] Synopsys Inc., "Open Source Liberty", [Online]. Available: https://www.synopsys.com/community/interoperability-programs/tap-in.html

[2] S. Saurabh, "Introduction to VLSI Design Flow", *Cambridge University Press*, 2023.

[3] C. Lutkemeyer, "A Practical Model to Reduce Margin Pessimism for Multi-Input Switching in Static Timing Analysis of Digital CMOS Circuits," *TAU Workshop*, 2015, pp. 10–20.

[4] D. Sinha, V. Rao, C. Peddawad, M. Wood, J. Hemmett, S. Skariah, and P. Williams, "Statistical Timing Analysis considering Multiple-Input Switching", *57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2020, pp. 1-6, doi: 10.1109/DAC18072.2020.9218601.

[5] Cadence Design Systems, Inc., "Tempus Timing Signoff Solution", [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html.

[6] R. Tayade, S. Nassif, and J. Abraham, "Analytical model for the impact of multiple input switching noise on timing", *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2008, pp. 514–517, doi: 10.1109/ASPDAC.2008.4484005.

[7] Y. Jun, K. Jun, and S. Park, "An accurate and efficient delay time modeling for MOS logic circuits using polynomial approximation", *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 8, no. 9, pp. 1027–1032, Sep. 1989, doi: 10.1109/43.35557.

[8] T. Fukuoka, A. Tsuchiya, and H. Onodera, "Statistical gate delay model for multiple input switching", *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2008, pp. 286–291, doi: 10.1109/ASP-DAC.2008.4483959.

[9] J. Shin, J. Kim, N. Jang, E. Park, and Y. Choi, "A gate delay model considering temporal proximity of multiple input switching", *International SoC Design Conference (ISOCC)*, Nov. 2009, pp. 577–580, doi: 10.1109/SOCDC.2009.5423815.

[10] A. R. Subramaniam, J. Roveda, and Y. Cao, "A finite-point method for efficient gate characterization under multiple input switching", *Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 1, pp. 1–25, Dec. 2015, doi: 10.1145/2778970.

[11] A. Ferdowsi, J. Maier, D. Öhlinger, and U. Schmid, "A simple hybrid model for accurate delay modeling of a multi-input gate", *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Antwerp, Belgium, 2022, pp. 1461-1466, doi: 10.23919/DATE54114.2022.9774547

[12] A. Ferdowsi, M. Függer, T. Nowak, U. Schmid, and M. Drmota, "Faithful dynamic timing analysis of digital circuits using continuous thresholded mode-switched ODEs", *Nonlinear Analysis: Hybrid Systems*, vol. 56, 2025, doi: 10.1016/j.nahs.2024.101572

[13] W. Raslan and Y. Ismail, "Deep-learning cell-delay modeling for static timing analysis", *Ain Shams Engineering Journal*, vol. 14, no. 1, 2023, doi: 10.1016/j.asej.2022.101828.

[14] F. Klemme, Y. Chauhan, J. Henkel, and H. Amrouch, "Cell library characterization using machine learning for design technology co-optimization", *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, November 2020, pp. 1–9, doi: 10.1145/3400302.3415713.

[15] S. M. Ebrahimipour, B. Ghavami, H. Mousavi, M. Raji, Z. Fang and L. Shannon, "Aadam: A Fast, Accurate, and Versatile Aging-Aware Cell Library Delay Model using Feed-Forward Neural Network", *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1-9.

[16] P. d'Hondt, A. Ladhar, P. Girard and A. Virazel, "A Learning-Based Methodology for Accelerating Cell-Aware Model Generation", *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Grenoble, France, 2021, pp. 1580-1585, doi: 10.23919/DATE51398.2021.9474227

[17] E. Naswali, N. Kim, and P. Chandran, "Fast and Accurate Library Generation Leveraging Deep Learning for OCV Modelling", *IEEE International Symposium on Quality Electronic Design (ISQED)*, pp. 349-354, 2021. doi: 10.1109/ISQED51717.2021.9424316.

[18] M. Agarwal and S. Saurabh, "An Efficient Timing Model of Flip-Flops Based on Artificial Neural Network", *ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, Raleigh, NC, USA, 2021, pp. 1-6, doi: 10.1109/MLCAD52597.2021.9531284.

[19] O. V. S. Shashank Ram and S. Saurabh, "Modeling Multiple-Input Switching in Timing Analysis Using Machine Learning", *Transactions on Computer-Aided Design of Integrated Circuits and Systems*

(TCAD), vol. 40, no. 4, pp. 723-734, April 2021, doi: 10.1109/TCAD.2020.3009624.
[20] H. Jiang, X. Cheng and P. Cao, "Multiple-Input Switching Modeling with Graph Neural Network", *2023 International Symposium of Electronics Design Automation (ISEDA), Nanjing, China*, 2023, pp. 428-432, doi: 10.1109/ISEDA59274.2023.10218616.
[21] S. Vollala, "A Graph-Based Approach For Computation Reduction Multi-input Switching," in *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2019.
[22] N. Eén and N. Sörensson, "An extensible SAT-solver", *Theory and Applications of Satisfiability Testing*, vol. 2919, Springer, 2004, pp. 502–518. doi: 10.1007/978-3-540-24605-3_37.
[23] Cadence Design Systems,"Virtuoso ADE Suite", [Online]. Available: https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-design/virtuoso-ade-suite.html
[24] Scikit-learn, "Cross-validation: Evaluating Estimators", [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.
[25] G. S. Tseitin, "On the complexity of derivation in propositional calculus", *Automation of Reasoning*, J. H. Siekmann and G. Wrightson, Eds. Berlin, Heidelberg: Springer, 1983, pp. 466–483. doi: 10.1007/978-3-642-81955-1_28.
[26] L.T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthya, and G. Yeric,"ASAP7: A 7-nm FinFET Predictive Process Design Kit," *Microelectronics Journal*, vol. 53, pp. 105-115, July 2016.
[27] "ISCAS Circuits", [Online]. Available: http://www.pld.ttu.ee/maksim/benchmarks/

Pooja Beniwal holds a B.Tech in Electronics and Communication and an M.Tech in VLSI Design. She is a PhD candidate in Electronics and Communication Engineering at Indraprastha Institute of Information Technology, Delhi, India. Her research interests span CAD for VLSI, digital design, and the integration of machine learning techniques to enhance EDA methodologies.

Sneh Saurabh (Senior Member, IEEE) received the B.Tech. degree from the IIT Kharagpur in 2000, and the Ph.D. degree from the IIT Delhi in 2012. He is a Professor at the Department of Electronics and Communication Engineering, IIIT Delhi. Before joining IIIT Delhi in June 2016, he worked in the semiconductor industry for 16 years. He holds three U.S. patents and is the author of the book Introduction to VLSI Design Flow and Fundamentals of Tunnel Field-Effect Transistors. His current research interests include energy-efficient systems, CAD for VLSI, and probabilistic computing. He is an Editor of IETE Technical Review and an Associate Editor of IEEE Access.

N Vignesh Chowdary is currently pursuing the B.Tech. degree in Electronics and Communication Engineering at the Indraprastha Institute of Information Technology Delhi (IIIT-Delhi), New Delhi, India, and is expected to graduate in 2025. He is a PDK Development Intern with Keysight Technologies, Gurgaon, India.

Suriya Skariah joined IBM Systems' Electronic Design Automation (EDA) Team in Bangalore in 2013 after earning her Master's in VLSI Design Engineering from Cochin University of Science & Technology. She specializes in standard cell characterization and develops IBM's gate-level timing methodologies and tools. Suriya's work involves creating detailed models for accurate timing analysis, power estimation, and reliability assessment in VLSI designs. She has extensive experience applying machine learning to EDA challenges, holds multiple US patents, and has published in leading conferences. Suriya has significantly contributed to optimizing IBM's server technologies and enhancing semiconductor device performance and reliability. She is also an active mentor and advocate for women in engineering and participates in industry panels and academic collaborations. Her interests include VLSI design automation, digital design, machine learning, and quantum computing.

Ajoy Mandal received the B.Tech degree in Electronics and Electrical Communication Engineering from Indian Institute of Technology, Kharagpur, India 2001. He is Senior Member Technical Staff and EDA solutions manager with Texas Instruments, Bengaluru, India. He has over 23 years of industry experience in Timing, Extraction, and Reliability analysis. He holds 5 patents and 33 publications in various international conferences and symposia or EDA design user groups.

Ramakrishnan Venkatraman is working with On Semiconductor Technology, India, as a Platform Architect. Prior to this, he was with Texas Instruments, India as a Distinguished Member Technical Staff, focusing on architecture definition and design of microcontroller systems, and design methodology definition for robustness and efficiency. He holds a Master of Engineering degree in Microelectronics from Indian Institute of Science, Bangalore. He has ¿75 publications in conferences and EDA vendor forums and holds 6 patents.