



# **Machine Learning-Driven Timing Analysis and Physical Design for VLSI**

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF

**DOCTOR OF PHILOSOPHY**

BY

**POOJA BENIWAL**

**(PHD20111)**

UNDER THE GUIDANCE OF

**DR. SNEH SAUARBH**

PROFESSOR, IIIT-DELHI

DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING  
INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI  
(IIIT-DELHI)

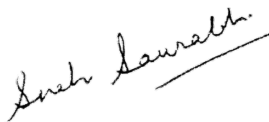
NEW DELHI- 110020

**August, 2026**

# CERTIFICATE

This is to certify that the thesis titled **Machine Learning-Driven Timing Analysis and Physical Design for VLSI**, submitted by **Pooja Beniwal**, to the Indraprastha Institute of Information Technology Delhi (IIIT-DELHI), for the award of the degree of **Doctor of Philosophy**, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.



**Dr. Sneh Saurabh**

Thesis Supervisor

Professor

Department of Electronics and Communications Engineering

Indraprastha Institute of Information Technology Delhi-110020, India

Place: New Delhi

Date: August, 2026

## Declaration

This is to certify that the thesis titled **Machine Learning-Driven Timing Analysis and Physical Design for VLSI**, submitted by **Pooja Beniwal**, to the Indraprastha Institute of Information Technology Delhi (IIIT-DELHI), for the award of the degree of **Doctor of Philosophy**, is an original research work carried out by me. This research work has been carried out under the supervision of **Dr. Sneh Saurabh**. The study pertaining to this thesis has not been submitted in part or in full, to any other University or Institution for the award of any other degree.

**Pooja Beniwal**   
PhD Candidate (PhD20111)  
Department of Electronics and Communications Engineering  
Indraprastha Institute of Information Technology Delhi-110020, India

Place: New Delhi  
Date: August, 2026

## **Declaration Regarding Use of AI in Thesis**

I acknowledge that I am fully responsible for the entire content of my thesis, including any sections assisted by online tools, including Artificial Intelligence-based tools. I accept full accountability for any violations of ethical standards in publications arising from the use of such tools.

**Pooja Beniwal**   
PhD Candidate (PhD20111)  
Department of Electronics and Communications Engineering  
Indraprastha Institute of Information Technology Delhi-110020, India

Place: New Delhi  
Date: August, 2026



## ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Sneha Saurabh, for his invaluable guidance, constant encouragement, and unwavering support throughout my research journey. His insightful feedback, patience, and rigorous approach to research have been instrumental in shaping this work. His high standards, attention to detail, and commitment to excellence have profoundly influenced both my academic and personal growth. It has been a privilege to work under his mentorship and benefit from his vast knowledge and experience.

I would also like to sincerely thank my collaborators-Suriya Skariah, Ajoy Mandal, and Ramakrishnan Venkatraman-for their valuable technical discussions and thoughtful insights. Their contributions have significantly enhanced the quality and depth of this research. I am equally grateful to my thesis committee members, Dr. Sujay Deb and Dr. Anuj Grover, for their time, constructive feedback, and encouraging guidance, which helped refine this work.

I extend my appreciation to the members of the Nanoscale Devices and Circuits research group, especially Dr. Shelly Garg, Dr. Abhinav Gupta, Dr. Amina Haroon, and Ms. Jasmine Kaur, for their support, detailed reviews, and engaging brainstorming sessions. I also acknowledge the newer members of the group-Mr. Yashwardhan Tyagi, Mr. Rithu Sagar, Mr. Nitin Dwivedi, and Ms. Garima-whose curiosity and thoughtful discussions brought fresh perspectives and renewed motivation during this journey.

I feel especially fortunate for the presence of wonderful friends and well-wishers-Ms. Sneha Aggarwal, Dr. Ritu Tomer, Dr. Alok Nikhil, Mr. Nitin Malik, Mrs. Prashasti, Mrs. Neelam Singh, Ms. Sonam, Mr. Deepank Grover, and Mr. Tarun Sharma-who have been a constant source of strength, encouragement, and positivity. Their companionship, understanding, and unwavering belief in me provided balance during challenging times and made this journey far more meaningful. The friendships I have built along the way are truly invaluable, and I will cherish them for a lifetime.

I gratefully acknowledge IIT Delhi for providing an excellent academic environ-

ment and research infrastructure. I also thank the academic staff and IT support team for their timely assistance and continuous support. I sincerely acknowledge the Semiconductor Research Corporation (SRC) and the University Grants Commission (UGC) for their financial support, which enabled this research.

Beyond all professional achievements, I am profoundly indebted to my family, whose love and support have been the cornerstone of my journey. I would like to especially thank my brother, Mr. Ravi Beniwal, and my sister, Mrs. Manju Beniwal Singh, for their constant encouragement, strength, and belief in me during every phase of this journey. Above all, I owe my deepest gratitude to my parents, Mr. Suresh Beniwal and Mrs. Amrita Beniwal, whose sacrifices, unconditional love, and unwavering faith in my abilities have made everything possible. This milestone stands as a reflection of their support and the values they have instilled in me. I could not have achieved this without them.

Finally, I express my heartfelt gratitude to Almighty God for His endless blessings, strength, guidance, and grace throughout this journey. His constant presence gave me the courage, perseverance, and hope to overcome every challenge and successfully complete this work.

I am thankful to everyone who, directly or indirectly, contributed to this journey and made it a memorable and enriching experience.

# ABSTRACT

Electronic design automation (EDA) comprises software tools for designing modern integrated circuits that includes synthesis, placement, routing, timing analysis, and verification. As designs scale to billions of transistors, challenges arise from increased complexity, tighter timing margins, higher power–performance–area (PPA) demands, and long runtimes. Traditional EDA methods in VLSI design face several challenges due to their reliance on simplified analytical models and tuned heuristics. These approaches often fail to capture complex, non-linear relationships among design parameters, resulting in limited accuracy. They are also computationally expensive and time-consuming because they depend on iterative optimization and repeated computation. As design complexity increases, traditional methods struggle to scale efficiently and require significant manual effort and expert intervention. Additionally, their ability to explore the design space is limited, often leading to suboptimal solutions, and they are less adaptable to variations and changing design conditions. Machine learning (ML) helps overcome these limitations by providing a data-driven approach that can model complex and high-dimensional relationships more efficiently. ML enables faster predictions compared to iterative methods, reducing computational cost and design time. It improves scalability for large and complex designs while minimizing the need for manual intervention. Furthermore, ML supports more efficient exploration of the design space and adapts better to variations, resulting in improved optimization and overall design quality.

In the first part of this research, standard-cell modeling is made more accurate and robust by incorporating Multi-Input Switching (MIS) effects, which are ignored under the Single-Input Switching (SIS) assumption of traditional standard-cell library characterization. MIS can cause significant delay variations and glitches at the output, resulting in inaccurate timing analysis, hold-time violations, signal integrity issues, increased power consumption, and incorrect sequential behavior. A novel satisfiability(SAT)-guided, ML-driven MIS characterization framework is proposed for multi-input combinatorial standard cells. First, a logical analysis of the Boolean function of a given logic

gate is performed to identify input patterns that can lead to MIS-induced speed-up or glitches. MIS-relevant patterns are extracted by formulating a SAT problem such that the satisfiable instances are the relevant input combinations. For each identified pattern, data is collected through SPICE simulations to capture the impact of MIS on timing attributes and glitches. This data is then used to develop an augmented technology library that accurately captures MIS effects by leveraging ML techniques. Subsequently, MIS-aware STA is performed using traditional STA tool coupled with the augmented technology library. Further, to streamline the entire process of pattern extraction, data generation, and augmented library creation, an automated tool framework named LiMo (Library Model), is developed in this work. This framework integrates a SAT solver, a SPICE simulator, dataset optimization strategies, ML training/testing infrastructure, and multiprocessing capabilities to create MIS-aware augmented timing libraries. The proposed framework is validated through experimental results on ISCAS benchmark circuits. In delay calculation of multi-input combinational cells, ignoring MIS leads to Relative Root Mean Square Error (RRMSE) exceeding 40% compared to golden SPICE results, whereas LiMo-generated libraries achieve RRMSE below 7% compared to golden SPICE. For predicting the attributes of the MIS-induced glitches, employing the augmented libraries model deliver RRMSE less than 6%. Moreover, computation using augmented libraries is  $10,000 \times$  faster than the SPICE simulation. Hence, the proposed methodology can be employed in MIS-aware timing analysis for industrial designs.

Building on advances in combinational standard-cell characterization, limitations of conventional timing models of sequential elements are addressed in this work. In traditional approaches, setup time (ST), hold time (HT), and clock-to-Q (C2Q) delays are modeled independently using two-dimensional lookup tables. Such a deterministic and decoupled representation fails to capture their interdependence, thereby introducing pessimism that results in artificial violations and unnecessary design iterations. To address these limitations, a novel statistical, ML-driven characterization framework is proposed, in which fixed timing constraints are replaced with a probabilistic formulation. Instead of explicitly modeling ST and HT, an ML-based latch prediction model is proposed to estimate the probability of correct data latching under multidimensional operating conditions, including setup/hold skews and process variations. Through this approach, complex parameter dependencies are learned, and a safe operating region is

defined within an extended timing space. In addition, a second ML model is developed for the prediction of C2Q delay within this extended timing region, ensuring accurate delay estimation even beyond conventionally defined timing constraints. Unlike traditional characterization methods, which assume fixed timing arcs, continuous and context-aware delay modeling is enabled under near-critical conditions. A hierarchical violation-waiver framework is proposed to ensure that violations are safely waived. Along with latching probability, it is verified that the extension of the timing space does not introduce issues for flip-flops that were already operating safely under traditional timing constraint. The proposed framework is validated using TAU benchmark circuits in 45 nm technology and is verified against Monte Carlo SPICE simulations. Experimental results demonstrate 100% precision in filtering marginal violations, with no false positives, while maintaining C2Q delay prediction errors below 2% when compared to golden SPICE values within the extended timing region.

In the final phase of this research, the focus shifts toward extending the application of machine learning techniques to address inaccuracies in congestion estimation during placement. Conventional early global routing (eGR) techniques, widely used for congestion-aware placement, rely on simplified models that fail to capture fine-grained variations in routing demand caused by cell dimensions, pin density, and local layout structures. This results in a mismatch with global routing (GR), leading to suboptimal placement decisions and degraded routing closure. To address this, an ML-driven congestion prediction and alleviation framework is proposed, leveraging the ability of ML to learn complex, high-dimensional relationships between placement features and routing demand. Instead of invoking a routing engine, the model predicts congestion directly from design features—including newly proposed features that capture local structural and pin-access characteristics, thereby enabling fast and accurate estimation. A placement-modification method is also introduced to alleviate predicted congestion hot spots, and its effectiveness is demonstrated through loose integration with an industrial place-and-route (PnR) tool, thereby enabling more robust and congestion-aware physical design flows. The proposed technique is evaluated on ISPD 2014/2015 benchmark designs and validated against golden global routing congestion obtained from a commercial PnR tool. An RMSE reduction of 68% in congestion prediction is achieved by the ML model compared to early global routing estimates, and an 8-30% reduction in routing congestion is enabled through improved optimization.

# TABLE OF CONTENTS

|                                                                                              |            |
|----------------------------------------------------------------------------------------------|------------|
| <b>ACKNOWLEDGEMENTS</b>                                                                      | <b>i</b>   |
| <b>ABSTRACT</b>                                                                              | <b>iii</b> |
| <b>LIST OF TABLES</b>                                                                        | <b>xi</b>  |
| <b>LIST OF FIGURES</b>                                                                       | <b>xiv</b> |
| <b>ABBREVIATIONS</b>                                                                         | <b>xv</b>  |
| <b>NOTATIONS</b>                                                                             | <b>xvi</b> |
| <b>1 Introduction</b>                                                                        | <b>1</b>   |
| 1.1 Motivation . . . . .                                                                     | 2          |
| 1.1.1 Motivation for the Selected Problems . . . . .                                         | 4          |
| 1.2 Research Objectives . . . . .                                                            | 5          |
| 1.3 Major Contributions . . . . .                                                            | 6          |
| 1.3.1 SAT-guided Framework to Capture MIS Effects . . . . .                                  | 6          |
| 1.3.2 Flip-Flop Timing Model for Resolving Marginal Timing Viola-<br>tions . . . . .         | 7          |
| 1.3.3 Prediction and Alleviation of Congestion during Placement .                            | 7          |
| 1.3.4 Overall Impact . . . . .                                                               | 8          |
| 1.4 List Of Publications . . . . .                                                           | 8          |
| 1.5 Organization . . . . .                                                                   | 9          |
| <b>2 Modeling Effects of Multi-input Switching (MIS) on Delay of Complex<br/>Logic Gates</b> | <b>11</b>  |
| 2.1 Background and Related Work . . . . .                                                    | 11         |
| 2.2 Proposed Work and Framework . . . . .                                                    | 14         |
| 2.3 Simultaneous Switching Timing Behavior . . . . .                                         | 16         |
| 2.3.1 Factors impacting behavior . . . . .                                                   | 16         |
| 2.3.2 Delay Profile in Complex Gates . . . . .                                               | 17         |
| 2.4 MIS Pattern Extraction . . . . .                                                         | 22         |
| 2.4.1 Notations and Problem Definition . . . . .                                             | 22         |
| 2.4.2 Mathematical Formulation . . . . .                                                     | 23         |
| 2.4.3 Algorithm . . . . .                                                                    | 25         |
| 2.4.4 Illustration . . . . .                                                                 | 25         |

|          |                                                                                           |           |
|----------|-------------------------------------------------------------------------------------------|-----------|
| 2.5      | Model Development . . . . .                                                               | 26        |
| 2.5.1    | Dataset generation and optimization . . . . .                                             | 27        |
| 2.5.2    | Model training and testing . . . . .                                                      | 28        |
| 2.6      | Augmenting Library with MIS annotation . . . . .                                          | 29        |
| 2.7      | Results . . . . .                                                                         | 31        |
| 2.7.1    | MIS Model Creation . . . . .                                                              | 31        |
| 2.7.2    | Circuit-Level Model Evaluation . . . . .                                                  | 35        |
| 2.7.3    | Impact of MIS Modeling on STA Accuracy . . . . .                                          | 39        |
| 2.7.4    | Evaluation of Runtime Performance . . . . .                                               | 40        |
| 2.8      | Conclusion . . . . .                                                                      | 41        |
| <b>3</b> | <b>Modeling Glitches due to Multi-input Switching (MIS) in Complex Logic Gates</b>        | <b>44</b> |
| 3.1      | Background and Related Work . . . . .                                                     | 44        |
| 3.2      | Proposed Work and Framework . . . . .                                                     | 46        |
| 3.3      | Design Attributes Impacting Glitch Generation . . . . .                                   | 48        |
| 3.3.1    | Transition Time ( $t_t$ ) . . . . .                                                       | 48        |
| 3.3.2    | Temporal Distance ( $t_D$ ) . . . . .                                                     | 49        |
| 3.3.3    | Capacitive Load ( $C_L$ ) . . . . .                                                       | 51        |
| 3.4      | Identification of Glitch-Inducing MIS Input Patterns in Multi-Input Logic Gates . . . . . | 52        |
| 3.4.1    | Illustration . . . . .                                                                    | 53        |
| 3.5      | Modeling Glitches using ML . . . . .                                                      | 55        |
| 3.5.1    | Dataset Generation and Optimization . . . . .                                             | 55        |
| 3.5.2    | Model Training . . . . .                                                                  | 58        |
| 3.5.3    | Deriving the Glitch Width ( $G_W$ ) . . . . .                                             | 59        |
| 3.5.4    | Integration of MIS-Aware Glitch Modeling into STA Flow . . . . .                          | 60        |
| 3.6      | Results . . . . .                                                                         | 62        |
| 3.6.1    | Gate-Level Model Evaluation . . . . .                                                     | 63        |
| 3.6.2    | Circuit-Level Model Evaluation . . . . .                                                  | 68        |
| 3.6.3    | Runtime Evaluation . . . . .                                                              | 74        |
| 3.6.4    | Glitch Attenuation . . . . .                                                              | 75        |
| 3.6.5    | Delay and Noise Impact of MIS-induced glitches . . . . .                                  | 77        |
| 3.7      | Conclusion . . . . .                                                                      | 81        |
| <b>4</b> | <b>Timing Model for Latching Probability in a Flip-flop and its Application</b>           | <b>83</b> |
| 4.1      | Background and Related Work . . . . .                                                     | 83        |

|          |                                                                  |            |
|----------|------------------------------------------------------------------|------------|
| 4.2      | Proposed Work and Framework . . . . .                            | 85         |
| 4.3      | Factors Impacting Timing in a Flip-flop . . . . .                | 87         |
| 4.4      | ML-driven Flip-flop Timing Models . . . . .                      | 89         |
| 4.4.1    | LatchPredict . . . . .                                           | 90         |
| 4.4.2    | C2QPredict . . . . .                                             | 96         |
| 4.5      | Comparison with Prior Work . . . . .                             | 99         |
| 4.5.1    | Case-Level Comparison . . . . .                                  | 100        |
| 4.5.2    | Complete Test Set Evaluation . . . . .                           | 103        |
| 4.6      | Timing Violation Waiver . . . . .                                | 104        |
| 4.6.1    | Tier 0: Identifying marginal violations . . . . .                | 104        |
| 4.6.2    | Tier 1: Checking latching probability . . . . .                  | 105        |
| 4.6.3    | Tier 2: Checking fanout . . . . .                                | 105        |
| 4.6.4    | Integration of Violation Waiver into STA Flow . . . . .          | 106        |
| 4.7      | Results . . . . .                                                | 107        |
| 4.7.1    | Waiving Violation . . . . .                                      | 107        |
| 4.7.2    | Detecting capture problem . . . . .                              | 109        |
| 4.7.3    | Detecting violation in the fanout . . . . .                      | 110        |
| 4.7.4    | Results on benchmark circuits . . . . .                          | 111        |
| 4.7.5    | Evaluation of Runtime Performance . . . . .                      | 118        |
| 4.7.6    | Using 7nm Libraries (ASAP 7nm PDK) . . . . .                     | 119        |
| 4.8      | Conclusions . . . . .                                            | 121        |
| <b>5</b> | <b>Prediction and Alleviation of Congestion during Placement</b> | <b>123</b> |
| 5.1      | Background and Related Work . . . . .                            | 123        |
| 5.2      | Proposed Work and Framework . . . . .                            | 124        |
| 5.3      | Model Development . . . . .                                      | 125        |
| 5.3.1    | Feature Selection . . . . .                                      | 128        |
| 5.3.2    | Impact of neighbourhood . . . . .                                | 130        |
| 5.3.3    | Model Selection and Fine Tuning . . . . .                        | 131        |
| 5.4      | Results . . . . .                                                | 132        |
| 5.4.1    | Congestion Prediction . . . . .                                  | 132        |
| 5.4.2    | Congestion Alleviation . . . . .                                 | 134        |
| 5.5      | Conclusion . . . . .                                             | 137        |
| <b>6</b> | <b>Conclusion and Future Work</b>                                | <b>138</b> |
| 6.1      | Conclusion . . . . .                                             | 138        |



|                                       |            |
|---------------------------------------|------------|
| 6.2 Future Work . . . . .             | 139        |
| <b>BIBLIOGRAPHY</b>                   | <b>142</b> |
| <b>LIST OF PAPERS BASED ON THESIS</b> | <b>153</b> |
| <b>LIST OF OTHER PAPERS</b>           | <b>154</b> |
| <b>BRIEF BIODATA OF THE AUTHOR</b>    | <b>155</b> |

# LIST OF TABLES

|      |                                                                                                                                                                |     |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 2.1  | Results of MIS Pattern Extraction (Proc. = tt, Volt. = 0.7 V, $T = 25^{\circ}\text{C}$ , $C_L = 25$ fF, $S_A = S_B = S_C = 160$ ps.) . . . . .                 | 26  |
| 2.2  | Performance Analysis of Machine Learning Models: Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), and Artificial Neural Network (ANN) . . . . . | 32  |
| 2.3  | Hyperparameter Tuning for ANN (NAND-2i/p) . . . . .                                                                                                            | 33  |
| 2.4  | Impact of Dataset Optimization . . . . .                                                                                                                       | 34  |
| 2.5  | Evaluation on 3-Input Gate (for region $R_2$ ) . . . . .                                                                                                       | 37  |
| 2.6  | Evaluation on Complex Gate (for region $R_2$ ) . . . . .                                                                                                       | 39  |
| 2.7  | Evaluation of Runtime Performance . . . . .                                                                                                                    | 40  |
| 3.1  | Extraction of Glitch-Inducing MIS Input Patterns in Multi-Input Gates . . . . .                                                                                | 54  |
| 3.2  | Performance of Regression Models on Various Logic Gates . . . . .                                                                                              | 64  |
| 3.3  | Hyperparameter Tuning Results for ANN (NAND-2i/p) . . . . .                                                                                                    | 66  |
| 3.4  | Performance Comparison for Glitch-Inducing MIS Input Patterns . . . . .                                                                                        | 67  |
| 3.5  | Evaluation of Glitches in ISCAS Benchmark Paths . . . . .                                                                                                      | 72  |
| 3.6  | Delay Impact of MIS Induced Glitches . . . . .                                                                                                                 | 79  |
| 4.1  | Performance Comparison of Single Models . . . . .                                                                                                              | 93  |
| 4.2  | Hyperparameter Tuning . . . . .                                                                                                                                | 93  |
| 4.3  | Confusion Matrix and Performance Metrics ( $\theta_{SPICE} = 50\%$ and $\theta_{Model} = 50\%$ ) . . . . .                                                     | 94  |
| 4.4  | Model Performance Comparison . . . . .                                                                                                                         | 98  |
| 4.5  | Hyperparameter Tuning Results (Stacked) . . . . .                                                                                                              | 98  |
| 4.6  | Input and Measured/Predicted $\tau_{C2Q}$ and $P_L$ Values . . . . .                                                                                           | 101 |
| 4.7  | Comparison with Previous and Proposed Models . . . . .                                                                                                         | 101 |
| 4.8  | Waiving Violation ( $\lambda = -10$ ps, $\theta_{Model} = 99.7\% (3\sigma)$ ) . . . . .                                                                        | 108 |
| 4.9  | Detecting capture problem ( $\lambda = -10$ ps, $\theta_{Model} = 99.7\% (3\sigma)$ ) . . . . .                                                                | 109 |
| 4.10 | Detecting violation in the fanout ( $\lambda = -10$ ps, $\theta_{Model} = 99.7\% (3\sigma)$ ) . . . . .                                                        | 110 |
| 4.11 | Effect of $\lambda$ at $\theta_{Model}=50\%$ for leon2_iccad and leon3mp_iccad . . . . .                                                                       | 112 |
| 4.12 | Effect of $\lambda$ at $\theta_{Model}=95.4\%$ for leon2_iccad and leon3mp_iccad . . . . .                                                                     | 113 |
| 4.13 | Effect of $\lambda$ at $\theta_{Model}=99.7\%$ for leon2_iccad and leon3mp_iccad . . . . .                                                                     | 113 |
| 4.14 | Comparison of Previous [19, 66] and Proposed Methodology ( $\lambda=-10\text{ps}$ ) . . . . .                                                                  | 116 |
| 4.15 | Evaluation of Runtime Performance . . . . .                                                                                                                    | 117 |
| 5.1  | Attributes of the ISPD14 and ISPD15 Designs used in this work . . . . .                                                                                        | 126 |

|     |                                                                          |     |
|-----|--------------------------------------------------------------------------|-----|
| 5.2 | Summary of Design Features that Impact Global Routing Congestion         | 126 |
| 5.3 | Results of VIF analysis . . . . .                                        | 129 |
| 5.4 | Comparison of ML Models and Attributes of ANN . . . . .                  | 131 |
| 5.5 | Results of Predicting Congestion ( $N_w \equiv$ Number of Windows) . . . | 134 |

# LIST OF FIGURES

|      |                                                                                                                                                                                               |    |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1  | SIS and MIS (a) SIS ( $A=1, B=1 \rightarrow 0, C=0$ ) . . . . .                                                                                                                               | 12 |
| 2.2  | SAT-Guided Framework for MIS Delay Modeling . . . . .                                                                                                                                         | 15 |
| 2.3  | Dependence of $\tau_r$ , $S_r$ on $t_D^{AC}$ in an AOI gate $Y = \neg(A.B + C)$ for varying (a, b) input slews (c, d) output capacitive load. . . . .                                         | 17 |
| 2.4  | SIS-like behavior: (a) charging current when $B$ arrives late (b) charging current when $C$ arrives late (c) delay profile. . . . .                                                           | 18 |
| 2.5  | State-dependent behavior: (a) charging current when $B$ arrives late (b) charging current when $C$ arrives late (c) delay profile. . . . .                                                    | 19 |
| 2.6  | 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. . . . . | 21 |
| 2.7  | Framework for MIS pattern extraction . . . . .                                                                                                                                                | 25 |
| 2.8  | (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. . . . .                                      | 28 |
| 2.9  | Model Evaluation Framework . . . . .                                                                                                                                                          | 35 |
| 2.10 | (a) ISCAS modified benchmark circuit C17 (added 3 input NAND gates) [28] (b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models. . . . .                       | 36 |
| 2.11 | (a) ISCAS modified benchmark circuit C17 (added 3 input AOI21 gates) [28] (b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models. . . . .                      | 38 |
| 2.12 | Relevant portion of the circuit s1196 for hold violation . . . . .                                                                                                                            | 39 |
| 3.1  | Glitch at 2 input NAND gate . . . . .                                                                                                                                                         | 45 |
| 3.2  | Proposed Methodology: SAT-Guided Framework for MIS Glitch Modeling . . . . .                                                                                                                  | 47 |
| 3.3  | Correlation of $G_H$ and $G_A$ with $t_t$ . . . . .                                                                                                                                           | 49 |
| 3.4  | Correlation of $G_H$ and $G_A$ with $t_D$ . . . . .                                                                                                                                           | 50 |
| 3.5  | Correlation of $G_H$ and $G_A$ with $C_L$ . . . . .                                                                                                                                           | 51 |
| 3.6  | Overall flow of modeling glitches using ML . . . . .                                                                                                                                          | 55 |
| 3.7  | $t_D$ vs Shape of Glitch . . . . .                                                                                                                                                            | 59 |
| 3.8  | Overall flow of integration of MIS-Aware glitch modeling into STA . . . . .                                                                                                                   | 61 |
| 3.9  | ISCAS benchmark circuit C17 and Flow for Testing. . . . .                                                                                                                                     | 69 |
| 3.10 | Evaluation of Glitches in C17 ISCAS Benchmark . . . . .                                                                                                                                       | 70 |
| 3.11 | ISCAS Benchmark Paths . . . . .                                                                                                                                                               | 71 |
| 3.12 | Comparison of glitch characteristics obtained using SPICE and proposed model in S1196 ISCAS benchmark path. . . . .                                                                           | 74 |

|      |                                                                                                                                                                                                                  |     |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 3.13 | Load Enhancement ( $t_{N13}$ : 20ps, $t_{N236}$ : 54ps, $t_{N367}$ : 18ps, $AT_{N13}$ - $AT_{N236}$ : 100ps, $AT_{N236}$ - $AT_{N367}$ : -100ps)                                                                 | 76  |
| 3.14 | Arrival Time Adjustment                                                                                                                                                                                          | 76  |
| 3.15 | Arrival Time Adjustment ( $t_{N13}$ :20ps, $t_{N236}$ :54ps, $t_{N367}$ :18ps, $C_L$ (Y1):5fF, $C_L$ (Y2):5fF)                                                                                                   | 77  |
| 3.16 | Example circuit ( $C_L$ :5fF, $C_C$ :5fF, $C_G$ :5fF)                                                                                                                                                            | 78  |
| 3.17 | Delay impact due to MIS-induced glitches for different input switching cases.                                                                                                                                    | 79  |
| 3.18 | Noise Impact of MIS induced glitches (Aggressor $G_H^{At} = 0.635V$ & Aggressor $G_H^{Pred} = 0.648V$ )                                                                                                          | 80  |
| 4.1  | Proposed Methodology: Top-level view of the modeling and resolution framework.                                                                                                                                   | 86  |
| 4.2  | Definition of circuit parameters in a flip-flop.                                                                                                                                                                 | 88  |
| 4.3  | (a) Setup skew ( $\tau_s$ ) vs. probability of latching ( $P_L$ ), (b) latching threshold for the setup skew $L_{setup}$ vs. hold skew ( $\tau_h$ ), (c) C2Q delay ( $\tau_{C2Q}$ ) vs. setup skew ( $\tau_s$ ). | 89  |
| 4.4  | Model creation framework                                                                                                                                                                                         | 90  |
| 4.5  | LatchPredict Classification Flow                                                                                                                                                                                 | 94  |
| 4.6  | Precision vs. Threshold Plot                                                                                                                                                                                     | 95  |
| 4.7  | Comparison of true positive (TP) and false positive (FP) counts across all 800 test points for the proposed probabilistic LatchPredict and previous ANN-based timing models [19, 66]                             | 102 |
| 4.8  | Timing violation waiver framework                                                                                                                                                                                | 104 |
| 4.9  | A portion of the circuit from leon2_iccad.                                                                                                                                                                       | 108 |
| 4.10 | A portion of the circuit from leon2_iccad.                                                                                                                                                                       | 109 |
| 4.11 | A portion of the circuit from leon2_iccad.                                                                                                                                                                       | 110 |
| 4.12 | (a) Slack Histogram leon3mp_iccad, (b) Slack Histogram leon2_iccad.                                                                                                                                              | 111 |
| 4.13 | Correlation plots for C2Q Delay (a) leon3mp_iccad, (b) leon2_iccad.                                                                                                                                              | 115 |
| 4.14 | (a) Setup skew ( $\tau_s$ ) vs. probability of latching ( $P_L$ ), (b) latching threshold for the setup skew $L_{setup}$ vs. hold skew ( $\tau_h$ ), (c) C2Q delay ( $\tau_{C2Q}$ ) vs. setup skew ( $\tau_s$ ). | 120 |
| 4.15 | Precision vs. Threshold Plot (ASAP 7nm PDK)                                                                                                                                                                      | 121 |
| 5.1  | Proposed Methodology: Top level view of Congestion Prediction and Alleviation                                                                                                                                    | 126 |
| 5.2  | Model development framework                                                                                                                                                                                      | 127 |
| 5.3  | Correlation between features and target variable                                                                                                                                                                 | 128 |
| 5.4  | $RMSE$ obtained for different feature sets                                                                                                                                                                       | 130 |
| 5.5  | Impact of neighborhood on $RMSE$                                                                                                                                                                                 | 130 |

|     |                                                                        |     |
|-----|------------------------------------------------------------------------|-----|
| 5.6 | Framework for benchmarking prediction accuracy . . . . .               | 132 |
| 5.7 | Congestion maps for <i>mgc_pci_bridge32_1</i> . . . . .                | 133 |
| 5.8 | Congestion maps for <i>mgc_ft_a</i> . . . . .                          | 133 |
| 5.9 | Results obtained using three congestion alleviation techniques . . . . | 135 |

## ABBREVIATIONS

|                 |                                                     |
|-----------------|-----------------------------------------------------|
| <b>EDA</b>      | Electronic Design Automation                        |
| <b>ML</b>       | Machine Learning                                    |
| <b>PPA</b>      | Power, Performance, and Area                        |
| <b>PVT</b>      | Process, Voltage, and Temperature                   |
| <b>SPICE</b>    | Simulation Program with Integrated Circuit Emphasis |
| <b>CMOS</b>     | Complementary Metal-Oxide-Semiconductor             |
| <b>TCL</b>      | Tool Command Language                               |
| <b>STA</b>      | Static Timing Analysis                              |
| <b>SIS</b>      | Single Input Switching                              |
| <b>MIS</b>      | Multi Input Switching                               |
| <b>SAT</b>      | Boolean Satisfiability                              |
| <b>RMSE</b>     | Root Mean Square Error                              |
| <b>SVR</b>      | Support Vector Regression                           |
| <b>DT</b>       | Decision Tree                                       |
| <b>RF</b>       | Random Forest                                       |
| <b>XGB</b>      | Extreme Gradient Boosting                           |
| <b>LightGBM</b> | Light Gradient Boosting Machine                     |
| <b>ANN</b>      | Artificial Neural Network                           |
| <b>SI</b>       | Signal Integrity                                    |
| <b>ST</b>       | Setup Time                                          |
| <b>HT</b>       | Hold Time                                           |
| <b>MC</b>       | Monte Carlo                                         |
| <b>PnR</b>      | Place and Route                                     |
| <b>eGR</b>      | Early Global Routing                                |
| <b>GR</b>       | Global Routing                                      |
| <b>VIF</b>      | Variance Inflation Factor                           |

## NOTATIONS

|                 |                                                      |
|-----------------|------------------------------------------------------|
| $t_D^{ij}$      | Temporal distance between two input pins $i$ and $j$ |
| $AT_i$          | Arrival times of the signals at inputs $i$           |
| $S_r/S_f$       | Rise/Fall output slews                               |
| $\tau_r/\tau_f$ | Rise/Fall delays                                     |
| $C_L$           | Load Capacitance                                     |
| $V_{DD}$        | Supply voltage                                       |
| $X_f$           | Feature matrix                                       |
| $y_t$           | Target labels                                        |
| $t_c$           | Clock Slew                                           |
| $t_d$           | Data Slew                                            |
| $\tau_s$        | Setup skew                                           |
| $\tau_h$        | Hold skew                                            |
| $\tau_{C2Q}$    | clock-to-Q delay                                     |



# CHAPTER 1

## Introduction

Electronic Design Automation (EDA) comprises tools that automate the complex processes involved in designing modern integrated circuits, including logic synthesis, placement, routing, timing analysis, and verification. As semiconductor technology advances and chips scale to billions of transistors, EDA tools face growing challenges. These include exploding design sizes, increasingly tight timing margins, stringent power-performance-area (PPA) constraints, and long computation cycles that hinder design productivity and time-to-market. Addressing these challenges with conventional heuristic-driven algorithms alone has become increasingly difficult, motivating the exploration of new paradigms for design automation. Machine learning (ML) has emerged as a promising and powerful approach to enhance EDA methodologies. By leveraging large volumes of design, simulation, and manufacturing data generated throughout the IC design flow, ML techniques can learn complex, high-dimensional relationships that are difficult to capture using conventional rules or analytical models. Unlike conventional methods that repeatedly solve each design problem independently, ML-driven approaches accumulate knowledge from prior designs and reuse it to guide optimization, enabling faster inference and more informed decision-making. This data-driven capability is particularly attractive for EDA problems, many of which are NP-complete or NP-hard and require navigating extremely large design spaces under tight runtime constraints.

In recent years, ML for EDA has become a rapidly growing research area, with extensive studies demonstrating the benefits of learning-based techniques across nearly all stages of the chip design flow. Applications include design space reduction and exploration, logic and physical synthesis, placement and routing optimization, timing and power modeling, testing and verification, and manufacturing-aware analysis. In many cases, ML-enhanced EDA tools have shown significant improvements over conventional methods in terms of runtime efficiency, modeling accuracy, and robustness to design and process variations. Rather than replacing existing algorithms, ML is increasingly used to complement and guide conventional EDA engines, enabling smarter

optimization and more informed design decisions.

Within this broader landscape, this dissertation focuses on the application of ML to critical stages of the EDA flow where accuracy, reducing pessimism, and runtime efficiency are dominant concerns. In particular, ML is applied to standard-cell characterization, static timing analysis (STA), and physical design to improve modeling accuracy, reduce unnecessary pessimism, lower overall design effort, and enable faster and more scalable design flows. By integrating data-driven learning techniques with established EDA frameworks, this work aims to demonstrate the practical potential of ML to enhance core EDA tasks and address key challenges in modern integrated circuit design. The following sections describe the specific motivations behind this work and the objectives pursued in this dissertation.

## 1.1 Motivation

The timing safety, functional correctness, speed, and reliability of modern digital integrated circuits are fundamentally dependent on the accuracy of STA, a cornerstone of EDA flows. At the heart of STA lie standard-cell timing libraries, which abstract transistor-level behavior into compact tabular models suitable for fast evaluation across billion-transistor designs. Therefore, the accuracy of STA is strongly dependent on the depth of abstraction and assumptions made in creating these standard-cell libraries[3].

Conventional standard-cell libraries are characterized under the Single-Input Switching (SIS) assumption, where only one input transition is considered while all other inputs are held at non-controlling values. While this abstraction simplifies characterization and keeps library sizes manageable, it fails to reflect real circuit operation. In practical designs, multiple inputs frequently switch simultaneously, producing Multi-Input Switching (MIS) effects. MIS can significantly alter effective gate delays [3]. Although STA designers attempt to compensate the errors by adding some conservative margins, neglecting MIS effects introduces systematic inaccuracies. These inaccuracies can manifest as misleading timing analysis results, expensive and unexpected hold time violations in silicon, or latent failures that emerge late in the design cycle-when fixes are most expensive [4], [5].

Beyond delay variation, MIS also induces glitches at the output of combinational

cells that are largely ignored in conventional timing libraries. These glitches cause signal integrity (SI) issues, increase dynamic power consumption, and propagate to downstream logic, potentially triggering timing or functional failures. As technology scales and clock frequencies rise, transition times shorten and input interactions become more temporally aligned, making MIS-induced glitches more frequent and severe [34, 35]. Accurately capturing glitch behavior requires modeling temporal correlations among multiple inputs under varying process, voltage, and temperature conditions.

The challenge is exacerbated for complex logic gates, where the number of possible MIS transition combinations grows exponentially with the number of input pins. Capturing all input exhaustively patterns would necessitate an impractical number of simulations, resulting in excessive computational overhead. As a result, a persistent gap exists between simplified library abstractions and actual circuit behavior. Bridging this gap requires intelligent methods that selectively identify critical MIS patterns and model their impact without resorting to brute-force simulation.

Besides inaccuracies in modeling combinations cells, conservative abstractions of sequential elements remain a major obstacle to timing closure. Traditional sequential timing models define safe operation using independently characterized setup time (ST), hold time (HT), and clock-to-Q (C2Q) delay tables. While these abstractions ensure correctness, they ignore intrinsic interdependencies among these parameters. In reality, setup and hold time constraints are not independent but are interrelated, and ignoring their mutual dependence can introduce unnecessary pessimism into the timing analysis, making timing closure more difficult [50], [51]. During signoff, this pessimism frequently results in marginal timing violations that often lead to unnecessary buffering, resizing, or pipeline restructuring. Many such violations are modeling artifacts rather than genuine functional risks, yet they incur area, power, and schedule penalties. Reducing this pessimism without compromising signoff safety demands a fundamentally different abstraction of sequential timing behavior.

In physical design implementation, routing congestion remains one of the most persistent challenges. To tackle this challenge, congestion-aware placement is traditionally performed that rely on early global routing (eGR) to estimate routing demand and guide cell placement decisions. However, eGR operates under coarse-grained assumptions about routing resources and fails to capture fine-scale variations caused by cell geom-

etry, pin density, macro placement, and local layout constraints. Consequently, early congestion estimates often diverge significantly from final routing outcomes. Improving eGR accuracy typically requires invoking more detailed routing engines, incurring substantial runtime overhead and limiting scalability [3, 75]. This mismatch motivates the development of fast and accurate congestion prediction techniques that can operate directly at the placement stage, improving design quality and turnaround time without invoking expensive routing computations.

### **1.1.1 Motivation for the Selected Problems**

Among the numerous EDA problems that could benefit from ML, this thesis focuses on three critical stages of the digital implementation flow: standard-cell characterization, static timing analysis, and physical design. These problems were selected because they represent fundamental bottlenecks in modern VLSI design, where modeling accuracy, computational efficiency, and scalability directly influence design closure and overall implementation quality.

Standard-cell characterization provides the timing and power models used by static timing analysis (STA), making it the foundation of reliable timing verification. Any inaccuracies introduced during characterization propagate throughout the design flow, potentially leading to timing pessimism or design failures. Accurate timing analysis is, in turn, essential for achieving timing closure and ensuring that designs meet their performance constraints. During physical design, placement quality directly affects routing congestion, which subsequently impacts timing, routing convergence, power, and overall design performance. Consequently, improvements at these stages have a cumulative impact on the complete chip implementation flow.

The rapid advancement of ML presents a transformative opportunity to address these challenges. ML models are capable of learning complex nonlinear relationships from simulation and design data, enabling accurate prediction of circuit behaviour that is difficult to capture using conventional analytical models alone. Once trained, these models provide fast inference, making them well suited for integration into iterative EDA workflows. Motivated by these capabilities, this thesis investigates the systematic integration of ML into standard-cell characterization, timing analysis, and physical design. Rather than replacing established EDA algorithms, the proposed methodologies

augment conventional design flows by combining ML with Boolean satisfiability (SAT), SPICE simulation, statistical timing analysis, and industrial place-and-route tools to improve modeling accuracy, reduce computational overhead, and enhance the scalability of modern VLSI design automation.

## 1.2 Research Objectives

The goal of this dissertation is to develop scalable, ML-driven methodologies that enhance timing and physical design accuracy while preserving compatibility with industrial EDA infrastructures and signoff requirements.

The first objective is to develop an MIS-aware combinational cell characterization framework that bridges the gap between SIS-based abstractions and real circuit behavior. This involves systematic extraction of transition combinations that can induce delay variations or glitches. By combining Boolean satisfiability-based logical analysis with targeted Simulation Program with Integrated Circuit Emphasis (SPICE) simulations, a high-quality dataset is constructed to train ML models capable of predicting MIS-induced delay and glitch behavior. The objective is not only to improve modeling accuracy but also to achieve orders-of-magnitude faster inference compared to direct SPICE simulation. An automated tool framework is designed to streamline the entire process of characterization, model generation, and creating MIS-aware libraries.

The second objective is to redefine sequential timing abstraction by replacing independently defined setup and hold constraints with a probabilistic and interdependency-aware framework. Instead of explicitly modeling ST and HT boundaries, the aim is to predict the probability of correct data latching across an extended timing space that naturally captures setup–hold coupling and the impact of process-induced variations. A complementary objective is to accurately predict C2Q delay within this extended space. The framework aims to safely filter marginal violations that are artifacts of pessimistic modeling while guaranteeing zero false positives, thereby improving timing closure efficiency without compromising robustness.

The third objective is to apply ML techniques to solve difficult challenges in physical design implementation, especially in predicting and alleviating congestion during placement and without invoking a routing engine. This includes systematic extraction

and evaluation of structural and layout features, development of a predictive model that approximates golden global routing congestion with high fidelity, and integration of a placement-modification strategy to mitigate predicted hotspots. The expected benefits include enhanced placement robustness, reduced routing iterations, and improved overall design turnaround time.

## 1.3 Major Contributions

The major contributions of this dissertation are presented below. These contributions address limitations of existing timing analysis and physical design flows and demonstrate how ML techniques can be systematically integrated to improve modeling accuracy, reduce pessimism, and enhance computational efficiency.

### 1.3.1 SAT-guided Framework to Capture MIS Effects

A novel SAT-guided and ML-based framework for capturing MIS effects has been developed. The major contribution is in the structured integration of Boolean satisfiability (SAT) engine with data-driven ML modeling to enable scalable and accurate standard-cell characterization beyond the conventional SIS abstraction. An integrated tool framework, called LiMo (Library Model), has been developed for MIS-aware timing characterization of complex combinational logic gates in this work. The identification of MIS-relevant input transitions was formulated as a SAT problem, allowing the exponential transition space to be pruned to a small, delay/glitch-critical subset. The ML models were then trained to capture MIS-induced variations in delay and glitch characteristics across operating conditions. LiMo integrates SAT solving, simulation, dataset optimization, multiprocessing, and ML modeling into a unified framework, producing MIS-aware timing libraries that substantially improve STA accuracy and enable orders of-magnitude faster characterization.

The proposed framework is validated through experimental results on ISCAS benchmark circuits. For delay modeling, ignoring MIS leads to RRMSE exceeding 40%, whereas LiMo-generated libraries achieve RRMSE below 7% compared to SPICE, with approximately 10,000× faster evaluation, enabling practical industrial MIS-aware STA. For glitch modeling, strong agreement with SPICE is observed, achieve RRMSE below

6%, while providing nearly 10,000 $\times$  faster inference.

### **1.3.2 Flip-Flop Timing Model for Resolving Marginal Timing Violations**

An ML-based framework has been proposed for flip-flops to define a safe operating region, thereby effectively extending the traditional timing space. The novelty is in predicting the probability of correct data latching by a flip-flop (LatchPredict) rather than explicitly modeling ST and HT constraints. The relevant circuit conditions, including setup and hold skews, are incorporated, and process-induced variations are taken into account in the proposed model. This allows dependencies among parameters neglected in conventional models to be implicitly captured in the proposed model. In addition, a second ML-based model (C2QPredict) has been developed to accurately predict the clock-to-Q (C2Q) delay within the extended timing space. The application of these models to resolve marginal timing violations through waivers, instead of design modifications, has been demonstrated. A hierarchical violation-waiver framework has been introduced to ensure that violations are safely waived. Furthermore, it has been verified that, in conjunction with latching probability evaluation, the extension of the timing space does not introduce any adverse effects for flip-flops that were already operating safely under traditional timing constraints.

The framework has been validated on TAU benchmark circuits implemented in 45nm technology and verified against Monte Carlo SPICE simulations. Experimental results demonstrate 100% precision in filtering marginal violations, with no false positives observed. Within the extended timing region, the error in predicting C2Q delay is less than 2% compared to golden SPICE values. The proposed model reduces the pessimism of the traditional flip-flop timing model, enabling safe and more efficient timing closure.

### **1.3.3 Prediction and Alleviation of Congestion during Placement**

An ML-driven congestion prediction and alleviation framework has been developed to address limitations of early global routing (eGR) in congestion-aware placement. Instead of invoking computationally expensive routing engines, fast and accurate con-

gestion estimation is achieved directly at the placement stage. A comprehensive set of structural and layout design features have been systematically extracted and evaluated, including local routing resource variations caused by cell geometry, pin density, and macro placement. A placement-modification method is also introduced to alleviate predicted congestion hot spots, and its effectiveness is demonstrated through loose integration with an industrial place-and-route (PnR) tool, thereby enabling more robust and congestion-aware physical design flows.

The proposed approach has been evaluated on ISPD 2014 and 2015 benchmark designs and validated against golden global routing congestion obtained from a commercial PnR tool. An RMSE reduction of 68% in congestion prediction has been achieved compared to early global routing estimates. Furthermore, routing congestion reductions ranging from 8% to 30% have been obtained through improved placement optimization.

### 1.3.4 Overall Impact

Collectively, this dissertation advances the state of the art by systematically embedding ML into critical stages of timing analysis and physical design. Modeling pessimism has been reduced, physical effects ignored by traditional flows have been considered, predictive accuracy has been significantly improved, and computational efficiency has been enhanced. The developed methodologies remain compatible with industrial design flows and signoff requirements, thereby contributing toward the realization of intelligent, data-driven EDA frameworks capable of addressing the growing complexity of next-generation integrated circuits.

## 1.4 List Of Publications

1. **P. Beniwal**, S. Saurabh, N. V. Chowdary, S. Skariah, A. Mandal, and R. Venkatraman, "LiMo: A Framework Leveraging Machine Learning for Multi-Input Switching Timing Models of Complex Logic Gates," *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 72, no. 12, pp. 7979–7992, Dec. 2025. doi: [10.1109/TCSI.2025.3579073](https://doi.org/10.1109/TCSI.2025.3579073).
2. **P. Beniwal**, A. Mandal, R. Venkatraman, S. Skariah, and S. Saurabh, "Modeling



Glitches due to Multiple Input Switching using Machine Learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Mar. 2026. doi: [10.1145/3802920](https://doi.org/10.1145/3802920).

3. **P. Beniwal**, S. Saurabh, A. Mandal, S. Skariah, and R. Venkatraman, "Machine Learning-Driven Flip-Flop Timing Model and Its Application in Resolving Marginal Timing Violations," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Dec. 2025. doi: [10.1145/3785294](https://doi.org/10.1145/3785294).
4. **P. Beniwal** and S. Saurabh, "Artificial Neural Network-based Prediction and Alleviation of Congestion during Placement," *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, Kolkata, India, 2024, pp. 300–305. [10.1109/VLSID60093.2024.00056](https://doi.org/10.1109/VLSID60093.2024.00056).

## 1.5 Organization

This dissertation is organized as follows:

- **Chapter 1** introduces the background and motivation behind this research, clearly defining the key objectives and contributions. It establishes the technical context and provides a roadmap for the subsequent chapters..
- **Chapter 2** presents the ML-driven SAT-guided framework for MIS induced delay variation for complex logic gates. This chapter details the methodology for identifying MIS-relevant input patterns impacting delays using SAT solvers, the development of ML models, and their integration into STA for improved accuracy.
- **Chapter 3** explains the ML-driven, SAT-guided framework for modeling MIS induced glitches for complex logic gates. It describes the workflow for efficient dataset generation, glitch attribute prediction, and how these models contribute to accurate timing and power analysis.
- **Chapter 4** introduces ML-Driven flip-flop timing models. It proposes the probabilistic modeling of correct data latching taking into account all circuit conditions and process-induced variations, and the application of these models to safely resolve marginal timing violations in STA.

- **Chapter 5** presents an ML-driven framework for predicting and alleviating routing congestion during placement. This chapter explains feature extraction from the layout, model development, and its application in guiding congestion-aware placement, reducing design iterations and improving overall design quality.
- **Chapter 6** concludes the dissertation by summarizing the key findings, highlighting the impact of the proposed ML methodologies across timing, glitch, and placement domains, and outlining potential directions for future research in EDA tool enhancement.

## CHAPTER 2

# Modeling Effects of Multi-input Switching (MIS) on Delay of Complex Logic Gates

This chapter presents an ML-driven, SAT-guided framework to model delay variations caused by MIS in complex logic gates. It details a systematic methodology for identifying MIS-relevant input patterns that significantly impact delay using SAT solvers, followed by the development of MIS-aware ML models. These models are further integrated into STA to improve accuracy in delay estimation.

The organization of this chapter is as follows. Section 2.1 gives a background by discussing previous works related to this chapter. Section 2.2 provides a top-level overview of the proposed framework. The simultaneous switching timing behavior of complex logic gates is discussed in Section 2.3. Section 2.4 describes the extraction of MIS-relevant input patterns using a SAT solver. The development of ML-driven models is presented in Section 2.5. Section 2.6 details the augmentation of conventional timing libraries with MIS-aware information. The results obtained using the proposed methodology are discussed in Section 2.7, and finally, the chapter is concluded in Section 2.8.

The work presented in this chapter has been published in [1].

## 2.1 Background and Related Work

Static Timing Analysis (STA) is an essential task in the VLSI design flow, as the timing correctness of a design is ensured by it. A given design is decomposed into various timing paths, which are then verified conservatively. A timing path is composed of timing arcs with delay attributes between the input and output pins of a cell, and the accuracy of STA is largely dependent on the timing models of these arcs. The most popular approach for modeling these timing arcs is the Liberty format look-up tables (LUTs), which provide a good trade-off between computational efficiency and accuracy [30]. During library characterization, delays corresponding to various combinations of input slews and load capacitances are computed using simulation and stored in the

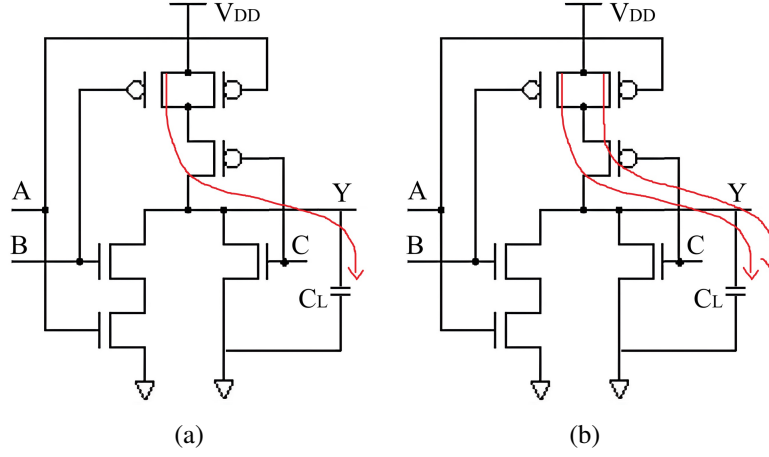


Figure 2.1: SIS and MIS (a) SIS ( $A=1, B=1 \rightarrow 0, C=0$ )  
(b) MIS ( $A=1 \rightarrow 0, B=1 \rightarrow 0, C=0$ )

library. An STA tool retrieves delay values from these tables under specific instantiation and operating conditions[3].

However, a limitation of the conventional timing model is the SIS assumption, under which only one input of a gate is considered to transition at a time while others are maintained at non-controlling values. Although this simplification is computationally efficient, realistic scenarios in which multiple inputs of a gate switch simultaneously, referred to as MIS, are not accounted for. MIS effects being neglected can result in inaccuracies in STA, potentially leading to design failures. For instance, hold time violations can be caused by overestimated early arrival times [4], [5].

An example illustrating the impact of MIS in an AND-OR-Invert (AOI) cell is presented in Fig. 2.1. Under SIS, when input  $A = 1$ ,  $B$  is switched from  $1 \rightarrow 0$  and  $C = 0$ , the output load capacitance  $C_L$  is charged through a single PMOS transistor, as depicted in Fig. 2.1a. However, under MIS, when both inputs  $A$  and  $B$  are switched simultaneously from  $1 \rightarrow 0$  with  $C = 0$ , the output load capacitance is charged through two parallel PMOS transistors, thereby resulting in a reduced delay compared to SIS, as shown in Fig. 2.1b.

In practice, to account for potential speed-up caused by MIS, MIS derates are often applied by commercial timing analysis tools [6]. Nominal cell delays are adjusted during timing analysis by applying a global or cell-specific scaling factor to the SIS-based delays. However, this approach is typically considered conservative and is based on clock relationships and timing proximity windows, rather than on the actual impact of MIS as determined by temporal distances. For instance, as shown in this chapter,

scenarios exist in which two inputs switch simultaneously, yet no delay variation due to MIS is observed in complex gates. Therefore, indiscriminate application of these derates is unnecessary and can result in overly pessimistic delay estimations.

In the past, several methods have been proposed for modeling the impact of MIS on delay calculations. An analytical model was presented in [7], in which SIS delays from standard-cell libraries were leveraged along with fitting constants derived from minimal MIS characterization. In [8], a polynomial approximation-based approach was proposed, requiring extensive SPICE simulations across multiple input combinations for coefficient extraction. Statistical timing analysis approaches [9, 10] make use of multiple SPICE simulations to extract coefficients, enabling the sensitivity of required arrival time to MIS-induced delays to be analyzed. In [11], a piecewise linear model was proposed using three or more discrete points to represent MIS effects, thereby reducing characterization effort. However, the accuracy of the model is heavily dependent on the precise identification of these finite points, and 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 [12], in which transistors were represented as ideal switches within a simplified RC network, and the resulting ordinary differential equations (ODEs) were solved analytically to yield explicit delay expressions for a two-input CMOS NOR gate. Building on this foundation, it was shown in [13] that the model in [12] remains continuous even under MIS effects, and the prior complicated approximation ODE solution was replaced with the derived explicit ODE solutions to obtain a more accurate MIS delay formula, complete with a direct parametrization procedure. Although valuable insights into MIS behavior are provided by these modeling techniques, complex computations are often involved, which limits their practical integration into STA tools.

In recent times, promise has been shown by ML in overcoming various limitations of conventional timing models. Deep learning techniques have been applied to STA for accelerating delay predictions, accounting for variations and ageing effects, and improving the accuracy and runtime of characterization, as well as design technology co-optimization (DTCO) [14]-[19]. ML-driven approaches have also been introduced in recent works for the modeling of MIS effects in timing model characterization [20, 21]. In [20], an Artificial Neural Network (ANN)-based model was used to capture the impact of MIS on timing attributes, and higher prediction accuracy was demonstrated com-

pared to conventional timing models. Similarly, in [21], a Graph Neural Network-based model was proposed, in which gate topologies were represented as graphs, enabling unified learning by the model. However, a key challenge with these models is posed by the fact that as the number of inputs increases, the effort required for ML-driven library characterization is also significantly increased. A 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 [11]. Exhaustive capturing of all input patterns would necessitate an impractical number of simulations, resulting in excessive computational overhead. The use of such models in STA can be cumbersome, thereby limiting the practicality of incorporating these effects in realistic designs with multi-input and complex gates.

While it has been demonstrated in previous studies that MIS-induced delay variations occur when two signals switch simultaneously, it is not always observed for complex gates. In some scenarios, even when two inputs are switched simultaneously, no MIS-induced delay variation is detected for complex gates (as detailed in Section 2.3). The filtering of such input patterns is considered crucial for reducing the size of the MIS dataset, thereby mitigating computational, data, and storage overhead and making MIS-aware characterization more practical for STA workflows. In this context, a graph-based pattern reduction algorithm was proposed in [22] to identify the worst-case input patterns for complex gates. Although significant reductions in characterization efforts are achieved by this method, only worst-case scenarios are focused on, which can result in overly pessimistic timing estimates.

## 2.2 Proposed Work and Framework

To address the challenges of previous works in modeling MIS effects in complex logic gates, a hybrid tool framework called LiMo is proposed, in which the combined power of a SAT solver and ML is leveraged to efficiently create and utilize MIS-aware timing models. Using the Boolean function of a given logic gate, input combinations capable of producing MIS-induced delay variations are extracted by formulating it as a SAT problem and employing a SAT solver to obtain the MIS-relevant transitions. By this approach, the number of simulations required for characterization is reduced, the size of the MIS dataset optimized, timing analysis can utilize MIS-specific information more

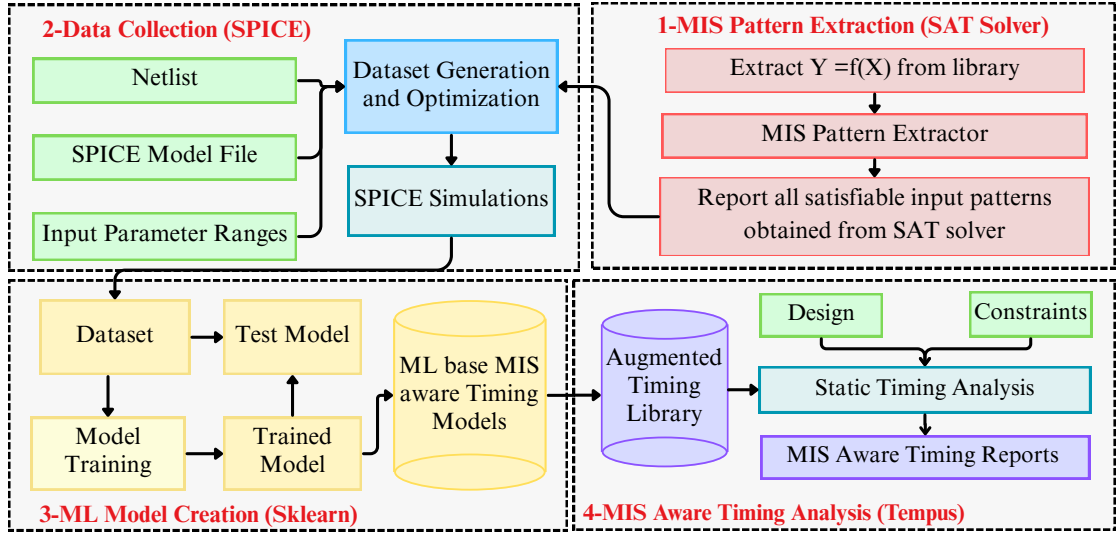


Figure 2.2: SAT-Guided Framework for MIS Delay Modeling

efficiently, and the scalability of MIS-aware STA, particularly for complex logic gates, is enhanced. Moreover, the power of ML is efficiently leveraged in LiMo to capture complex dependencies among various design attributes affecting MIS effects by automating dataset generation, data handling, model training, and validation. Significant reductions in characterization runtime are also achieved by appropriately decomposing the problem and utilizing multiprocessing capabilities. Additionally, with a Tool Command Language (TCL)-based interface, debugging capabilities, and features such as the handling of Process-Voltage-Temperature (PVT) corners, a practical solution for realistic industrial designs is proposed and demonstrated by LiMo.

A top-level view highlighting the major tasks in the proposed methodology is shown in Fig. 2.2. For a given complex gate, MIS-relevant patterns are extracted by formulating it as a SAT problem and employing an efficient SAT solver [23]. For each identified pattern, data is collected using SPICE [24] simulations to capture the impact of MIS on the timing attributes. This data is subsequently used to experiment with and create an augmented technology library, in which ML techniques are leveraged through the scikit-learn libraries [25]. Thereafter, MIS-aware STA is performed using industrial tool [6], which is loosely coupled with the generated libraries.

The proposed methodology is implemented in the form of LiMo, as shown in Fig. 2.2. LiMo is designed as an automated framework with a TCL-based interface, which enables the seamless integration of the SAT solver, SPICE simulator, dataset optimization, ML training/testing infrastructure, and multi-processing capabilities. The MIS-relevant input transitions for each gate are determined by LiMo using the MIS pattern extractor.

Training data is then efficiently generated using SPICE simulations. Since this stage is the most time-intensive in 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 are used to 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 of the model to evaluate its accuracy and generalization capabilities. Finally, trained ML models and augmented libraries are generated by LiMo, enabling STA tools to account for MIS-induced delay variations.

## 2.3 Simultaneous Switching Timing Behavior

### 2.3.1 Factors impacting behavior

The simultaneous switching behavior of multi-input gates is strongly influenced by the temporal distance or skew, which is used to quantify the time difference between the arrivals of signals at the input pins. For a given gate, the temporal distance between two input pins  $i$  and  $j$ , denoted as  $t_D^{ij}$ , is mathematically expressed as  $t_D^{ij} = (AT_j - AT_i)$ , where  $AT_i$  and  $AT_j$  represent the arrival times of the signals at the inputs  $i$  and  $j$ , respectively. For simplicity, temporal distance is denoted as  $t_D$  in this chapter (when a specific pair of input pins is not being referred).

The relationships between  $t_D$  and timing attributes, such as rise/fall delays ( $\tau_r/\tau_f$ ) and rise/fall output slews ( $S_r/S_f$ ), are considered crucial for the modeling of MIS effects. Reducing  $|t_D|$  results in shorter delays and output slews, resulting in speed-up [20]. As illustrated in Fig. 2.3a -2.3d, the speed-up is observed in a 3-input AOI21 gate when temporal distance approaches zero. The reduction in  $|t_D^{AC}|$  leads to a significant decrease in timing attributes:  $\tau_r$  and  $S_r$ . A characteristic V-shaped delay profile is exhibited by the delay and output slew, which is induced by the MIS effect.

While simultaneous transitions across all inputs are theoretically possible, such events are considered to be statistically less likely [11]. Consequently, the pairwise temporal distances between input signals are analyzed in this work for gates with more than two inputs. For example, in a three-input gate with input pins  $A$ ,  $B$ , and  $C$ , the



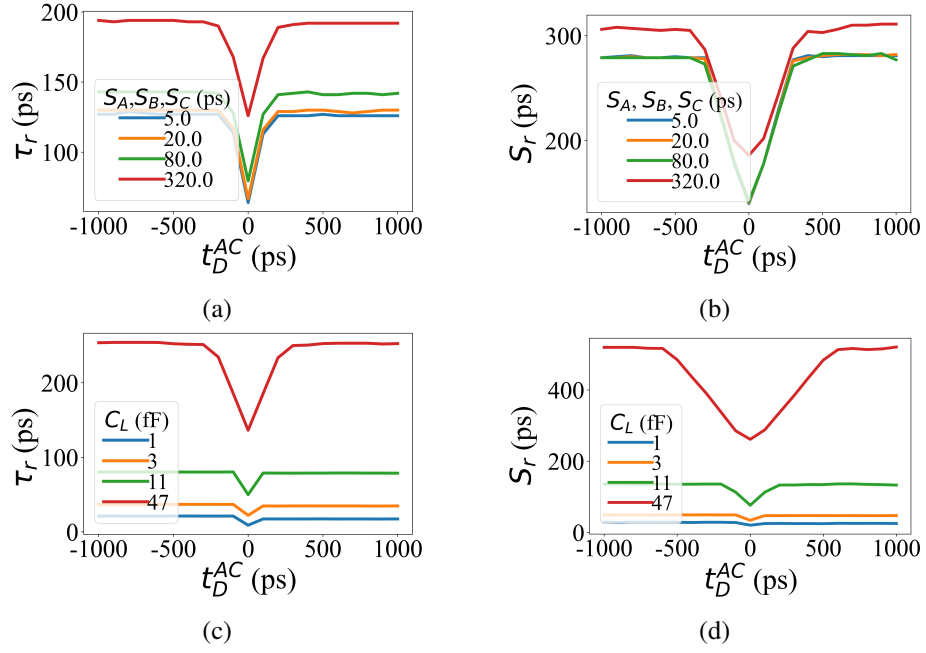


Figure 2.3: Dependence of  $\tau_r$ ,  $S_r$  on  $t_D^{AC}$  in an AOI gate  $Y = \neg(A.B + C)$  for varying (a, b) input slews (c, d) output capacitive load.

relevant pairwise temporal distances are considered to be 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 than the simultaneous arrival of all three input signals. By focusing on these pairwise signal arrivals, which have a significant impact on delay variations, MIS-induced delay variations can be effectively modeled using a simplified approach that emphasizes the timing overlaps most likely to trigger MIS effects. It should be noted that in the unlikely event of three or more inputs switching near-simultaneously, the proposed framework would need to be extended to handle such cases.

In addition to  $t_D$ , other parameters, such as input slew (denoted as  $S_i$  for input pin  $i$ ) and output load capacitance (denoted as  $C_L$ ), are considered essential for the modeling of 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 Figs. 2.3a-2.3d.

### 2.3.2 Delay Profile in Complex Gates

It has been demonstrated in previous studies that MIS-induced delay variations occur when two signals switch simultaneously [20] -[22]. However, for complex gates, three distinct types of delay profiles are exhibited as the temporal distance  $t_D$  varies under the simultaneous switching of two signals. This phenomenon is illustrated using a three-

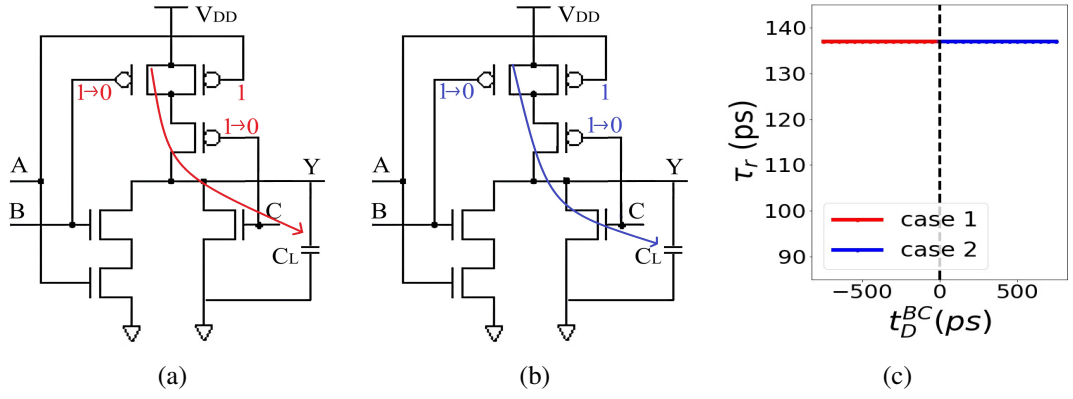


Figure 2.4: SIS-like behavior: (a) charging current when  $B$  arrives late (b) charging current when  $C$  arrives late (c) delay profile.

input AOI gate, for which  $Y = \neg(A \cdot B + C)$ .

### SIS-like behavior

Let us assume that the initial input conditions are  $A = 1$ ,  $B = 1$ , and  $C = 1$ , under which the output is  $Y = 0$ . When  $C$  is held at 1, the output remains 0 regardless of the values of  $A$  and  $B$ , indicating that the output is not sensitive to variations in  $A$  and  $B$  unless  $C$  transitions to 0. Similarly, when  $A = 1$  and  $B = 1$  are held constant, the output remains 0 regardless of the value of  $C$ , indicating that the output is not sensitive to changes in  $C$  until either  $A$  or  $B$  transitions.

Next, the transition  $B = 1 \rightarrow 0$  and  $C = 1 \rightarrow 0$  is considered, as illustrated in Fig. 2.4a. Under this condition, a transition of the output to  $Y = 1$  is expected. However, the output transition is triggered only when both inputs toggle, as a transition on a single input is insufficient to change the output state. Consequently, the delay is determined by the input that arrives later. As shown in Fig. 2.4a, for  $t_D^{BC} \ll 0$  (red curve in Fig. 2.4c), input  $C$  arrives earlier and is effectively settled to 0 by the time input  $B$  arrives. Similarly, as illustrated in Fig. 2.4a, for  $t_D^{BC} \gg 0$  (blue curve in Fig. 2.4c), input  $B$  arrives earlier and is effectively settled to 0 by the time input  $C$  arrives. In both cases, the output load capacitance is charged through the same conduction path, consisting of two PMOS transistors connected in series. Furthermore, for intermediate values of the temporal distance, the charging path remains unchanged. As a result, no variation in delay with respect to  $t_D^{BC}$  is exhibited. Despite the simultaneous switching of inputs, the resulting behavior is therefore SIS-like.

The behavior described above arises when the output is not individually sensitive to

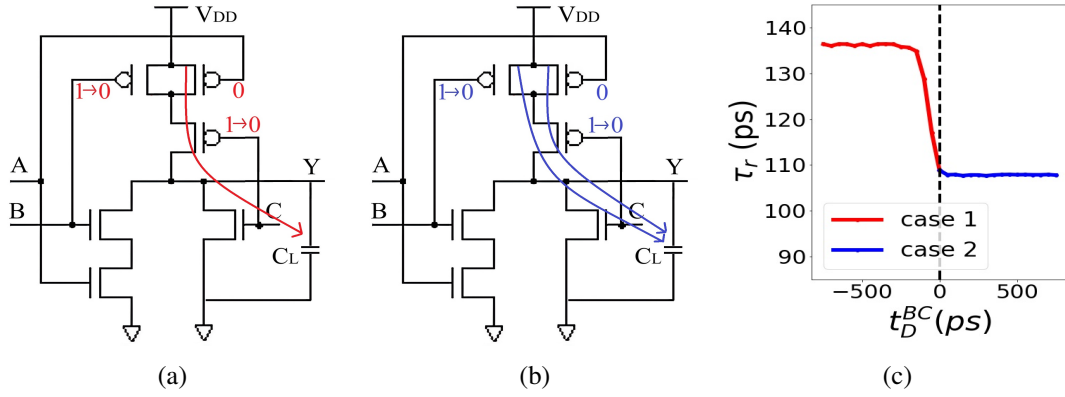


Figure 2.5: State-dependent behavior: (a) charging current when  $B$  arrives late (b) charging current when  $C$  arrives late (c) delay profile.

either input, but is capable of transitioning only when both inputs switch concurrently through a series conduction path in the CMOS transistor stack. Since the conduction path remains identical for both  $t_D^{BC} < 0$  and  $t_D^{BC} > 0$ , a V-shaped delay profile is not exhibited. This type of transition is already captured in conventional timing libraries through SIS arcs (e.g.,  $A = 1, B = 1 \rightarrow 0, C = 0$ ). Therefore, such transitions are not considered for the modeling of MIS effects in this work.

### State-Dependent delay behavior (SD)

Let us assume that the initial input conditions are  $A = 0, B = 1$ , and  $C = 1$ , under which the output is  $Y = 0$ . When  $A = 0$  and  $B = 1$  are held constant, the output is not forced to either 0 or 1, but is instead determined by the value of  $C$ , indicating sensitivity of the output to changes in  $C$ . However, when  $C$  is held at 1, the output remains 0 regardless of the values of  $A$  and  $B$ , indicating that the output is not sensitive to variations in  $A$  and  $B$ .

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

Next, let us consider that  $B$  changes from  $1 \rightarrow 0$  and  $C$  changes from  $1 \rightarrow 0$ , as illustrated in Fig. 2.5a. Under this transition, the output is expected to switch to 1. The output transition is triggered by the transition of  $C$ , whereas the transition at  $B$  is not critical, since the output is insensitive to  $B$ . Consequently, the delay is primarily determined by the arrival time of the signal at the controlling input, namely  $C$ . However, the values at the non-controlling inputs determine the conduction paths through which

the output load capacitor is charged.

As shown in Fig. 2.5a, for  $t_D^{BC} \ll 0$  (corresponding to the red curve in Fig. 2.5c), the transition at  $B$  arrives later and may be treated as remaining at logic 1 when the transition at  $C$  occurs. In this case, the output load capacitor is charged through a single conduction path, as depicted in Fig. 2.5a, resulting in a larger delay value. In contrast, as shown in Fig. 2.5b, for  $t_D^{BC} \gg 0$  (corresponding to the blue curve in Fig. 2.5c), the transition at  $B$  arrives earlier and may be treated as having settled to logic 0 by the time the transition at  $C$  occurs. In this case, the output load capacitor is charged through two parallel conduction paths, as illustrated in Fig. 2.5b, resulting in a smaller delay value. For intermediate values of the temporal distance, an intermediate delay is observed, corresponding to partial overlap of the two charging paths during the output transition. Consequently, a step-like delay profile is exhibited.

The behavior described above arises when the output is sensitive to one input while remaining insensitive to the other input. Although the non-controlling input does not directly determine the output logic value, its transition enables or disables specific conduction paths in the CMOS transistor network, leading to distinct delay values. This type of delay behavior is conventionally captured in standard-cell libraries using 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 the Liberty format, such arcs are modeled separately using distinct “when” conditions.

Under simultaneous switching conditions, the resulting delay remains bounded by these state-dependent delay values. As a result, STA tools select the more pessimistic value, ensuring that no false positives are introduced during timing analysis. Therefore, this class of transitions is not considered for the modeling of MIS effects in this work.

### **MIS-induced speed-up**

Let us assume that the initial input conditions are  $A = 1, B = 1$ , and  $C = 0$ , under which the output  $Y$  is forced to 0. When  $B = 1$  and  $C = 0$  are held constant, the output is not forced to either logic value but is determined by the value of  $A$ , indicating the output is sensitive to changes in  $A$ . Similarly, when  $A = 1$  and  $C = 0$  are held constant, the output is not forced to either logic value but is determined by the value of  $B$ , indicating the output is sensitive to changes in  $B$ . Thus, the output is sensitive to

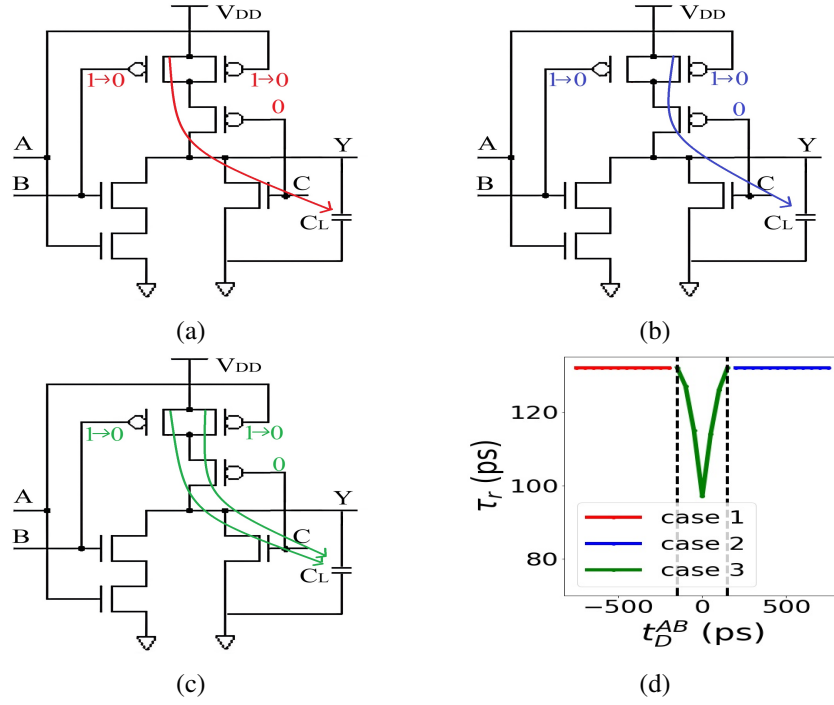


Figure 2.6: MIS-induced speed-up: (a) charging current when  $A$  arrives late (b) charging current when  $A$  and  $B$  arrives together (c) charging current when  $B$  arrives late (d) delay profile.

both  $A$  and  $B$  for the above input combination.

Next, let us consider the transition  $A = 1 \rightarrow 0$  and  $B = 1 \rightarrow 0$ , as shown in Fig. 2.6a. The output is expected to transition to 1. This transition is expected to occur when either  $A$  or  $B$  toggles, since the output is sensitive to both inputs. In Fig. 2.6a, for  $t_D^{AB} \ll 0$  (red curve in Fig. 2.6d),  $B$  arrives early and  $A$  can be treated as 1 (retaining its previous value). Hence, in this case, the output capacitor is charged through a single path, as shown in Fig. 2.6a. Similarly, for  $t_D^{AB} \gg 0$  (blue curve in Fig. 2.6d),  $A$  arrives early and  $B$  can be treated as 1 (retaining its previous value). Accordingly, the output capacitor is charged through a single path, as shown in Fig. 2.6b. However, when  $|t_D^{AB}|$  approaches zero, the output capacitor is charged through two paths, as shown in Fig. 2.6c. Consequently, when both transitions occur simultaneously, the delay is significantly reduced, and the characteristic MIS V-shaped delay profile is observed (green curve in Fig. 2.6d).

The behavior described above occurs when the output is sensitive to both inputs that switch simultaneously. In such cases, simultaneous toggling of both inputs activates multiple conduction paths, charging the output capacitance more rapidly and thereby reducing the delay. In conventional libraries, these transitions are modeled as separate

SIS arcs. However, the speed-up from simultaneous switching results in delays that are significantly lower than those observed with individual arcs. Consequently, although STA tools select pessimistic values, this mismatch can lead to false positives and potential circuit failures. Therefore, this chapter emphasizes identifying and modeling this class of transitions 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 filter transitions capable of causing a circuit failure due to ignored MIS effects. In the following section, the MIS pattern extraction framework is presented, which leverages logic-based techniques to identify transitions expected to produce MIS-induced speed-up.

## 2.4 MIS Pattern Extraction

### 2.4.1 Notations and Problem Definition

Let us represent the logical functionality of a complex logic gate as:

$$Y = f(X)$$

where  $Y$  denotes the output and  $X = x_1, x_2, \dots, x_N$  denotes the set of  $N$  input signals of the logic gate. Let us represent an input transition as  $X' \rightarrow X''$ , where the input changes from  $X' = x'_1, x'_2, \dots, x'_N$  to  $X'' = x''_1, x''_2, \dots, x''_N$ . The MIS pattern extraction algorithm is required to identify all transitions  $X' \rightarrow X''$  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 (these glitch causing transitions will be discussed in detail in the next chapter).

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, the above properties are encoded mathematically, which enables a SAT solver to be employed and the MIS-relevant patterns to be automatically extracted.

## 2.4.2 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 is evaluated to 1:

$$M(X', X'', x_i, x_j) = (x'_i \oplus x''_i) \cdot (x'_j \oplus x''_j) \cdot \prod_{k \neq i, j} [\neg(x'_k \oplus x''_k)] \quad (2.1)$$

The first two terms represent the XOR of the pre- and post-switching values for  $x_i$  and  $x_j$ , ensuring that their values are toggled. 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, it is ensured 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:

$$T(X', X'') = f(X') \oplus f(X'') \quad (2.2)$$

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

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

$$\begin{aligned} \frac{\partial Y}{\partial x_i} &= f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_N) \oplus \\ &\quad f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_N) \end{aligned} \quad (2.3)$$

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  $x_i$ ; otherwise, it evaluates to 0. In other words, when  $\frac{\partial Y}{\partial x_i}$  evaluates to 1 and the value of  $x_i$  is toggled, the output  $Y$  is expected to toggle. Hence, the Boolean difference is capable of capturing the sensitivity of a function with respect to a given variable.

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

$$S(X', x_i, x_j) = \frac{\partial Y}{\partial x_i} \cdot \frac{\partial Y}{\partial x_j} \Big|_{X=X'} \quad (2.4)$$

Here, the logical AND of the two terms on the RHS is taken. The first term ensures that the output changes when  $x_i$  is toggled, while the second term ensures that  $x_j$  affects the output. Together, these terms guarantee that the output is influenced by the transitions of both inputs. The extent of the impact due to simultaneous transitions (relative to a single transition) depends on the transistor-level implementation of  $Y$ , input slews, and the output load, and cannot be estimated from a logical analysis. However, the above condition is used to filter out transitions that are not expected to exhibit MIS-induced delay variations, thereby significantly simplifying the characterization task.

The input transitions that are of interest for MIS-induced delay variation modeling are required to satisfy all three conditions mentioned above. Hence, by combining Eq. 2.1, 2.2, and 2.4, for an MIS transition  $X' \rightarrow X''$  (with  $x_i$  and  $x_j$  toggling), the following function must evaluate to 1:

$$M(X', X'', x_i, x_j) \cdot T(X', X'') \cdot S(X', x_i, x_j) \quad (2.5)$$

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 considered for MIS-induced delay variation modeling:

$$P(X', X'') = \bigvee_{(i,j)} M(X', X'', x_i, x_j) \cdot T(X', X'') \cdot S(X', x_i, x_j) \quad (2.6)$$

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



### 2.4.3 Algorithm

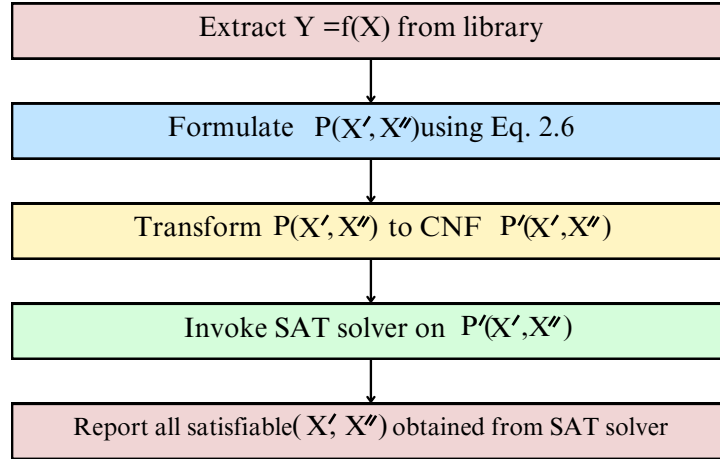


Figure 2.7: Framework for MIS pattern extraction

Figure 2.7 shows the framework for MIS pattern extraction. The Boolean function  $Y = f(X)$  is extracted for a given logic gate from the conventional library. A logic expression  $P(X', X'')$  is constructed using  $f(X)$  and Eq. 2.6. 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 is given to a SAT solver in a suitable format [26]. In this work, MiniSat is used, though any other efficient SAT solver can be employed to extract the MIS pattern [23]. The SAT solver is allowed 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  $f(X)$  is typically a simple function in libraries and contains only a few variables, the problem is quite easy for the state-of-the-art SAT solvers. Hence, the runtime needed in MIS pattern extraction is negligible compared to other tasks involved in creating MIS-augmented libraries.

### 2.4.4 Illustration

The results obtained using the above algorithm are presented and are validated 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

Table 2.1: Results of MIS Pattern Extraction (Proc. = tt, Volt. = 0.7 V,  $T = 25^\circ\text{C}$ ,  $C_L = 25$  fF,  $S_A = S_B = S_C = 160$  ps.)

| (a) AOI: Rising |       |                                                                                   |         |            | (b) AOI: Falling |       |                                                                                     |      |            |
|-----------------|-------|-----------------------------------------------------------------------------------|---------|------------|------------------|-------|-------------------------------------------------------------------------------------|------|------------|
| $X'$            | $X''$ | Delay                                                                             | MIS?    | Extracted? | $X'$             | $X''$ | Delay                                                                               | MIS? | Extracted? |
| 110             | 000   |  | Yes     | Yes        | 000              | 110   |  | No   | No         |
| 011             | 000   |  | No (SD) | No         | 000              | 011   |  | No   | No         |
| 001             | 010   |  | No (SD) | No         | 010              | 001   |  | No   | No         |
| 111             | 100   |  | No      | No         | 100              | 111   |  | Yes  | Yes        |
| 101             | 000   |  | No (SD) | No         | 000              | 101   |  | No   | No         |
| 111             | 010   |  | No      | No         | 010              | 111   |  | Yes  | Yes        |
| 001             | 100   |  | No (SD) | No         | 100              | 001   |  | No   | No         |

| (c) OAI: Rising |       |                                                                                   |      |            | (d) OAI: Falling |       |                                                                                     |         |            |
|-----------------|-------|-----------------------------------------------------------------------------------|------|------------|------------------|-------|-------------------------------------------------------------------------------------|---------|------------|
| $X'$            | $X''$ | Delay                                                                             | MIS? | Extracted? | $X'$             | $X''$ | Delay                                                                               | MIS?    | Extracted? |
| 111             | 001   |  | No   | No         | 001              | 111   |  | Yes     | Yes        |
| 011             | 000   |  | Yes  | Yes        | 000              | 011   |  | No      | No         |
| 111             | 100   |  | No   | No         | 100              | 111   |  | No (SD) | No         |
| 101             | 110   |  | No   | No         | 110              | 101   |  | No (SD) | No         |
| 101             | 000   |  | Yes  | Yes        | 000              | 101   |  | No      | No         |
| 111             | 010   |  | No   | No         | 010              | 111   |  | No (SD) | No         |
| 011             | 110   |  | No   | No         | 110              | 011   |  | No (SD) | No         |

these transitions, 14 involve simultaneous switching of two inputs: 7 leading to rise transitions and 7 to fall transitions. For each case,  $t_D$  (between  $A$ - $B$ ,  $B$ - $C$ , and  $A$ - $C$ ) is varied in the range of  $-750$  ps to  $+750$  ps, and the rise delay ( $\tau_r$ ) or fall delay ( $\tau_f$ ) profiles are observed using SPICE simulations.

The results are summarized in Table 2.1 (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 from the MIS pattern extractor align with the SPICE simulation results (as shown in the last two columns). Thereby validating the effectiveness of the proposed technique for deriving MIS-specific patterns.

## 2.5 Model Development

For each pattern identified by the MIS pattern extractor, a delay model quantifying MIS-induced delay variations is developed by leveraging ML, as described in the following

paragraphs.

### 2.5.1 Dataset generation and optimization

The dataset is generated using SPICE simulations for the given two pins  $i$  and  $j$  of 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 rise delay/output slew ( $\tau_r/S_r$ ) and fall delay/output slew ( $\tau_f/S_f$ ) are treated as labels. For the features  $S_i$ ,  $S_j$ , and  $C_L$ , the starting value ( $x_{\text{start}}$ ) and ending value ( $x_{\text{end}}$ ) 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 user-defined ratio. The progression is continued until  $x_n$  exceeds  $x_{\text{end}}$ . This method generates a desired number of intermediate values between  $x_{\text{start}}$  and  $x_{\text{end}}$ , with the control parameter  $r$  determining the spacing between them. For  $t_D$ , an arithmetic progression is used to uniformly distribute values between a user-defined range (minimum  $t_D^{\min}$  and maximum  $t_D^{\max}$ ):

$$t_D^{(i)} = t_D^{\min} + (i - 1) \cdot \Delta t_D \quad (2.7)$$

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

$$\Delta t_D = \frac{t_D^{\max} - t_D^{\min}}{M} \quad (2.8)$$

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 of  $S_i$ ,  $S_j$ ,  $C_L$ , and  $t_D^{ij}$ , simulations are performed, and the resulting  $\tau_r$ ,  $S_r$ ,  $\tau_f$ , and  $S_f$  are stored in an output file for further processing and analysis.

To reduce characterization efforts, a data optimizer algorithm is also employed as follows. From Fig. 2.8a-2.8b, it can be inferred that the delay profile can be divided into three distinct regions R1, R2, and R3: regions R1 and R3 exhibit negligible variation in delay, while region R2 (around  $t_D^{\text{AC}} = 0$ ) exhibits significant delay variations. The boundaries between these regions within the range  $[t_D^{\min}, t_D^{\max}]$  are identified by the

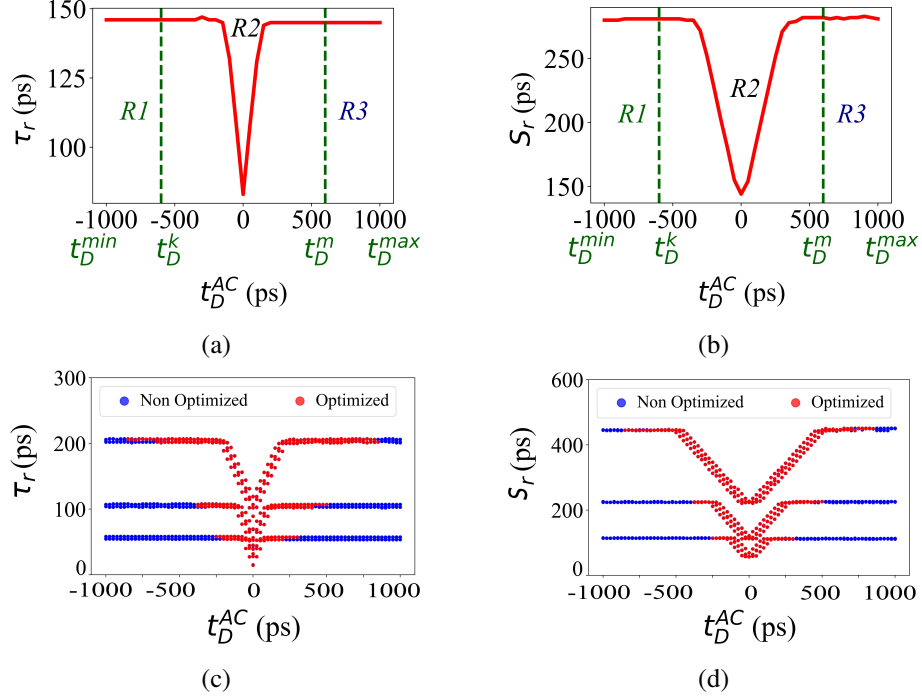


Figure 2.8: (a, b) Definition of regions R1, R2, and R3 for delay and slew data collection (c, d) Effectiveness of optimization strategy for delay and slew.

algorithm, which 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 no variation in delay and output slew is observed for three successive values of  $t_D$ , the corresponding value of  $t_D$  is marked as  $t_D^k$ , representing the boundary between regions R1 and R2. Similarly,  $t_D^m$ , representing the boundary between regions R2 and R3, is determined 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. The effectiveness of the proposed strategy for delay and output slew is illustrated in Fig. 2.8c - 2.8d.

## 2.5.2 Model training and testing

The dataset  $D = \{X, y\}$ , generated by a data generator and optimizer, undergoes pre-processing before being utilized for model training and testing. Here,  $X \in R^{n \times m}$  represents the feature matrix, where  $n$  is the number of samples and  $m$  is the number of features, while  $y \in R^n$  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  $D_{\text{train}} = \{X_{f,\text{train}}, y_{t,\text{train}}\}$  and a testing set  $D_{\text{test}} = \{X_{f,\text{test}}, y_{t,\text{test}}\}$ .

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 considered crucial to enhance model performance. Methods such as 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 employed. The training dataset  $D_{\text{train}}$  is divided into  $k$  equally sized disjoint folds, denoted as  $D_1, D_2, \dots, D_k$ . In each fold  $i$ , the model is trained on the other  $(k - 1)$  folds and is validated on the  $i$ -th fold. This process is repeated  $k$  times, with each fold serving as the validation set exactly once [25]. The model performance is assessed using the Normalized Root Mean Squared Error (NRMSE). First, the Root Mean Squared Error (RMSE) is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{t=1}^N (y_t - \hat{y}_t)^2} \quad (2.9)$$

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:

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{\max} - y_{\min}} \quad (2.10)$$

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

## 2.6 Augmenting Library with MIS annotation

The ML model created by LiMo needs to be integrated with the STA tools. This can be achieved by augmenting the existing SIS-based libraries in Liberty format [30] 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  $y$ . The MIS effect is proposed to be modeled 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, the corresponding MIS-specific ML model is associated with this input pin  $j$ . This information can be extracted by an STA tool from the augmented technology library and utilized to compute MIS-aware timing attributes ( $\tau_r$ ,  $S_r$ ,  $\tau_f$ , and  $S_f$ ) using circuit conditions such as the  $S_i$ ,  $S_j$ ,  $t_D^{ij}$ , and  $C_L$ .

As an illustration, the three-input AOI gate shown in Table 2.1 is considered. 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) {
      ...
    }
  }
}
```

This arc in the library is augmented 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";
      }
    }
  }
}
```

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  $A \rightarrow Y$ . This information can be utilized by an STA tool 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 considered adequate. Thus, the proposed approach ensures that MIS-specific models are used only when required, while maintaining the efficiency of a conventional SIS-based model in non-MIS scenarios. It is worth noting that the “mis\_info” group and the above-named attributes are not supported by standard Liberty, and EDA tool support would require parsing and understanding them.

## 2.7 Results

Experiments were performed on standard cells implemented using the ASAP 7nm process design kit (PDK) [27]. Industry-standard tools Tempus and Virtuoso were used to gather these results [6, 24]. First, the results of MIS model creation are presented, followed by the results of applying the model for timing analysis.

### 2.7.1 MIS Model Creation

Experiments were conducted with various types of ML models, hyperparameters were explored, and optimization strategies were applied to reduce computational effort.

#### Comparative Analysis of ML models

Table 2.2 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. During this tuning process, both training and validation errors were monitored and kept comparable, suggesting that the models generalize well to un-

seen data. As shown in the table, there is no significant difference between the training, validation, and test errors, indicating that the models are well optimized.

Table 2.2: Performance Analysis of Machine Learning Models: Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), and Artificial Neural Network (ANN)

| Gate           | DT     |        |        | RF     |        |        |
|----------------|--------|--------|--------|--------|--------|--------|
|                | Train  | Val    | Test   | Train  | Val    | Test   |
| NAND-2i/p      | 0.0360 | 0.0398 | 0.0450 | 0.0360 | 0.0416 | 0.0482 |
| AND-2i/p       | 0.0281 | 0.0280 | 0.0310 | 0.0201 | 0.0237 | 0.0218 |
| NOR-2i/p       | 0.0263 | 0.0294 | 0.0266 | 0.0260 | 0.0295 | 0.0300 |
| OR-2i/p        | 0.0321 | 0.0350 | 0.0368 | 0.0304 | 0.0335 | 0.0342 |
| NAND-3i/p      | 0.0301 | 0.0343 | 0.0339 | 0.0295 | 0.0327 | 0.0351 |
| AND-3i/p       | 0.0315 | 0.0340 | 0.0367 | 0.0297 | 0.0321 | 0.0340 |
| NOR-3i/p       | 0.0330 | 0.0345 | 0.0372 | 0.0312 | 0.0342 | 0.0361 |
| OR-3i/p        | 0.0342 | 0.0360 | 0.0385 | 0.0320 | 0.0350 | 0.0374 |
| AOI21-3i/p     | 0.0308 | 0.0370 | 0.0317 | 0.0277 | 0.0300 | 0.0325 |
| OAI-3i/p       | 0.0318 | 0.0342 | 0.0371 | 0.0292 | 0.0324 | 0.0355 |
| <b>Average</b> |        |        | 0.0354 |        |        | 0.0344 |

| Gate           | GB     |        |        | ANN    |        |        |
|----------------|--------|--------|--------|--------|--------|--------|
|                | Train  | Val    | Test   | Train  | Val    | Test   |
| NAND-2i/p      | 0.0345 | 0.0378 | 0.0425 | 0.0305 | 0.0319 | 0.0320 |
| AND-2i/p       | 0.0242 | 0.0289 | 0.0257 | 0.0238 | 0.0250 | 0.0244 |
| NOR-2i/p       | 0.0212 | 0.0221 | 0.0223 | 0.0225 | 0.0228 | 0.0225 |
| OR-2i/p        | 0.0290 | 0.0311 | 0.0323 | 0.0281 | 0.0295 | 0.0302 |
| NAND-3i/p      | 0.0313 | 0.0330 | 0.0339 | 0.0332 | 0.0337 | 0.0328 |
| AND-3i/p       | 0.0289 | 0.0304 | 0.0311 | 0.0300 | 0.0314 | 0.0311 |
| NOR-3i/p       | 0.0328 | 0.0339 | 0.0363 | 0.0342 | 0.0356 | 0.0336 |
| OR-3i/p        | 0.0335 | 0.0348 | 0.0371 | 0.0324 | 0.0339 | 0.0345 |
| AOI21-3i/p     | 0.0242 | 0.0255 | 0.0272 | 0.0287 | 0.0290 | 0.0288 |
| OAI-3i/p       | 0.0277 | 0.0293 | 0.0315 | 0.0286 | 0.0298 | 0.0298 |
| <b>Average</b> |        |        | 0.0319 |        |        | 0.0299 |

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



Table 2.3: 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     | <b>(150, 150, 150)</b>    | <b>0.0320</b> |
| (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        |

### Hyperparameter Tuning for ANN

Table 2.3 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 considered crucial in optimizing model performance by selecting the best configurations to minimize prediction errors. In this study, a grid search was used to evaluate different hyperparameter combinations, including the number of neurons, activation functions, optimizers, learning rates, regularization, and batch sizes. The ReLU activation function and the Adam optimizer were selected, with a learning rate of 0.01 and an L2 regularization parameter of 0.01. From Table 2.3, 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 observation suggests that simpler models are sufficient to capture the complex relationships in the data. Hence, balancing model complexity and predictive accuracy is crucial for creating MIS models. Similar hyperparameter tuning was performed for other gates, yielding similar insights into the optimal configurations.

Table 2.4: 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        |
| <b>Average</b>       | <b>1,42,208</b> | <b>1,13,762</b> | <b>28,446</b> | <b>0.0299</b> |

| 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        |
| <b>Average</b>    | <b>43,374</b> | <b>43,374</b> | <b>28,446</b> | <b>0.0318</b> |

### Optimized Gate Level Evaluation

Table 2.4 compares the performance of models with and without dataset optimization across various logic gates. The NRMSE obtained using the optimized training dataset is shown to have a comparable error to that of the model trained with a large, un-optimized dataset. However, the optimized dataset contains significantly fewer data points, thereby 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 considered cru-

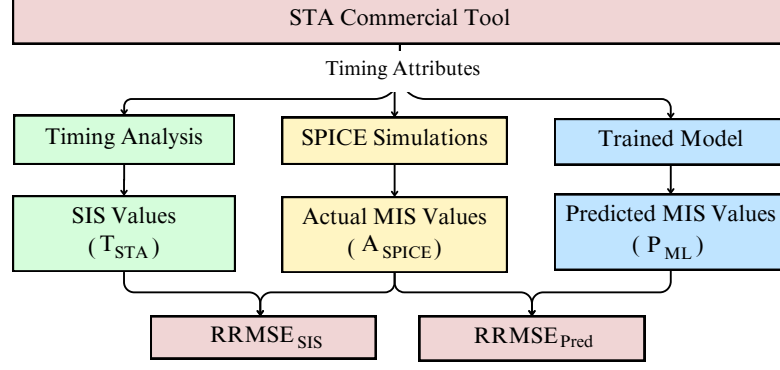


Figure 2.9: Model Evaluation Framework

cial in modeling MIS effects efficiently. It is important to note that the test dataset is kept consistent across both models to ensure a fair comparison.

### 2.7.2 Circuit-Level Model Evaluation

To assess the efficacy of the proposed methodology at the circuit level, the evaluation framework illustrated in Fig. 2.9 is used. First, STA is performed to extract key timing attributes ( $S_i$  (max),  $S_j$  (max),  $AT_i$ (max),  $AT_j$ (max), and  $C_L$ ) from Tempus [6], which are then utilized in SPICE simulations and the trained ML model. The STA tool provides SIS-based timing attributes  $T_{STA}$  (rise delay  $\tau_r^{SIS}$ , rise output slew  $S_r^{SIS}$ , fall delay  $\tau_f^{SIS}$ , and fall output slew  $S_f^{SIS}$ ). The SPICE simulations provide actual values for these timing attributes, denoted as  $A_{SPICE}$  ( $\tau_r^{At}$ ,  $S_r^{At}$ ,  $\tau_f^{At}$ , and  $S_f^{At}$ ). The trained ML model generates predicted values for these attributes, denoted as  $P_{ML}$  ( $\tau_r^{Pred}$ ,  $S_r^{Pred}$ ,  $\tau_f^{Pred}$ , and  $S_f^{Pred}$ ).

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

$$RRMSE_{SIS} = \left( \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (A_{SPICE,i} - T_{STA,i})^2}}{\frac{1}{N} \sum_{i=1}^N A_{SPICE,i}} \right) \quad (2.11)$$

$$RRMSE_{Pred} = \left( \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (A_{SPICE,i} - P_{ML,i})^2}}{\frac{1}{N} \sum_{i=1}^N A_{SPICE,i}} \right) \quad (2.12)$$

where  $N$  represents the total number of samples.

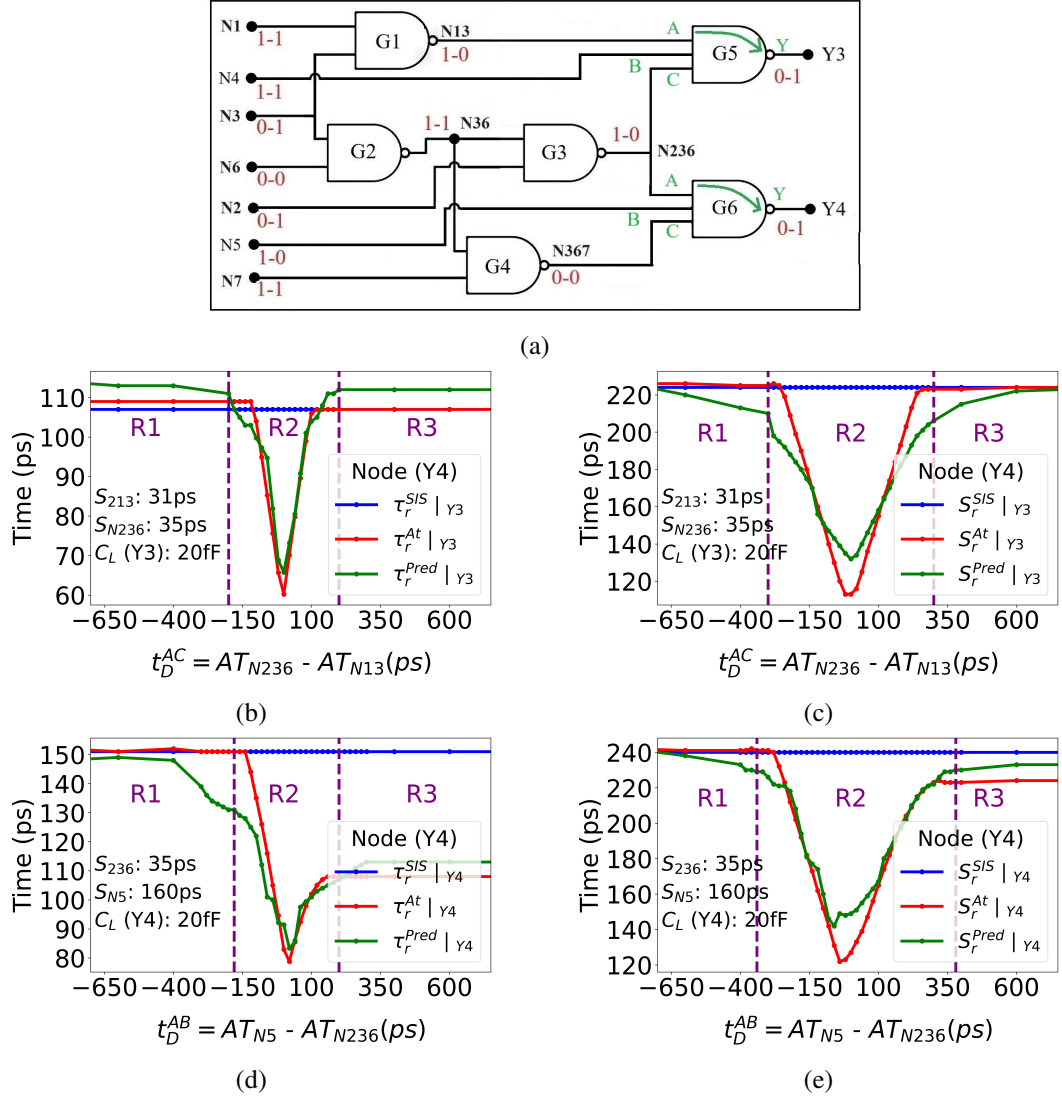


Figure 2.10: (a) ISCAS modified benchmark circuit C17 (added 3 input NAND gates) [28]  
(b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models.

### Evaluation on 3-Input NAND Gate

The accuracy of the proposed ML model is evaluated on the circuit shown in Fig. 2.10a. MIS effects are considered 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. Results obtained using these ML models are presented in the following paragraphs.

The arrival time of the signals at the primary inputs is adjusted such that, for the

Table 2.5: Evaluation on 3-Input Gate (for region  $R2$ )

| Metric         | Y3       |        | Y4       |        | Average       |
|----------------|----------|--------|----------|--------|---------------|
|                | $\tau_r$ | $S_r$  | $\tau_r$ | $S_r$  |               |
| $RRMSE_{SIS}$  | 18.41%   | 59.01% | 39.70%   | 59.01% | <b>44.03%</b> |
| $RRMSE_{Pred}$ | 4.03%    | 6.72%  | 7.24%    | 6.72%  | <b>6.17%</b>  |

gate G5, pins A (net N13) and C (net N236) switch simultaneously, while the pin B (net N4) is held constant. For the gate G6, pins A (net N236) and B (net N5) switch simultaneously, while the pin C (net N367) is held constant.

Figure 2.10b and 2.10c 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. 2.10d and Fig. 2.10e compare the corresponding results for the gate G6 (port Y4). It is observed that, in general, the ML model is able to match the SPICE simulation in region R2, in which the MIS effect is dominant. For quantitative comparison, the RRMSE is shown in Table 2.5 for the SIS model and the proposed model. It is observed that, on average, ignoring MIS effects results in an error of 44.03% in timing attributes, whereas the proposed model predicts them with an error of 6.17%.

### Evaluation on Complex Gate

The accuracy of the proposed ML model is evaluated on the circuit shown in Fig. 2.11a. MIS effects are considered in the three-input AOI gates G5 and G6, driving nets Y5 and Y6, respectively.

For an AOI gate, as discussed earlier, ML models are generated by LiMo to capture MIS effects for transitions extracted from the MIS pattern extractor, and the SIS-based library is augmented with this information. Results predicted using these ML models are presented in the following paragraphs.

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

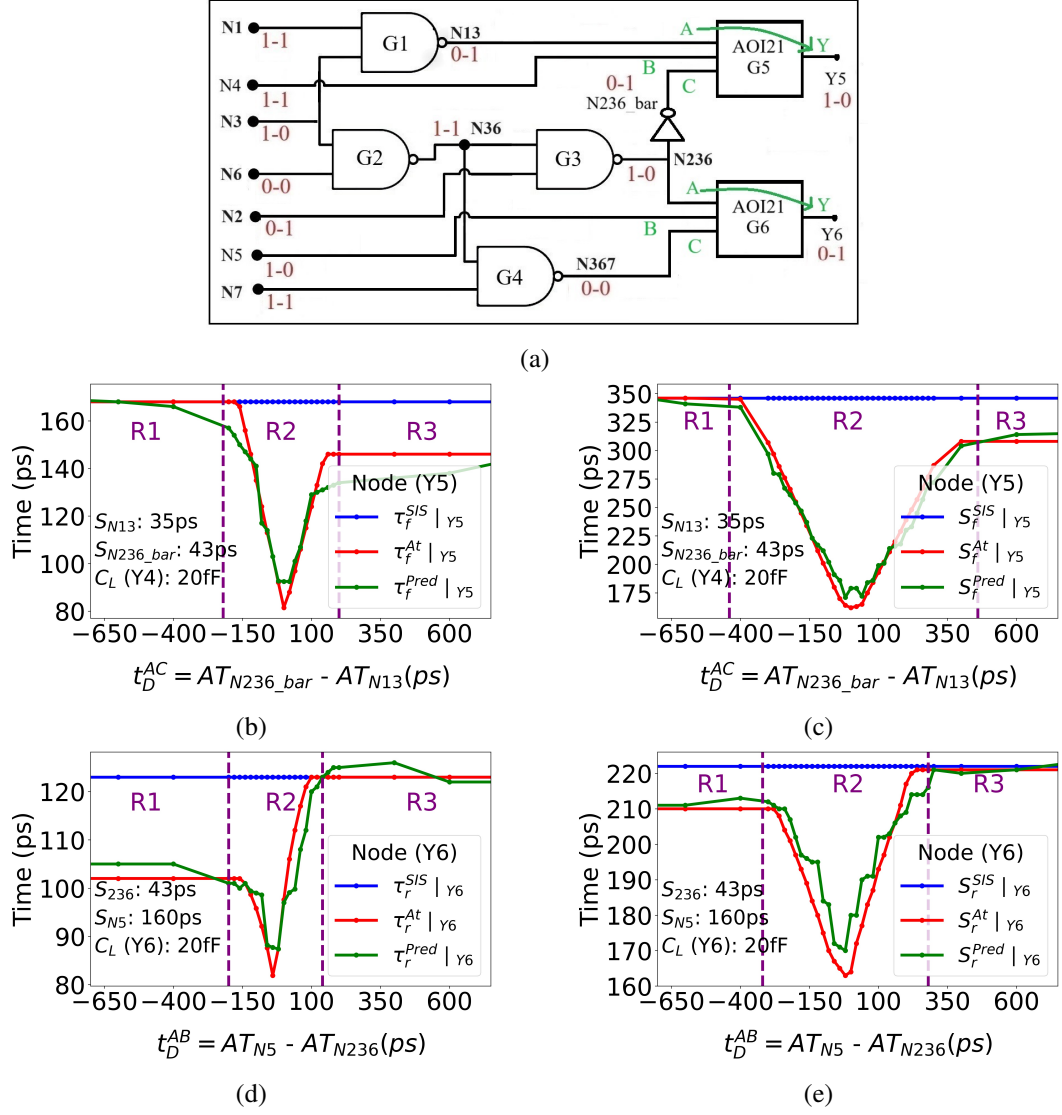


Figure 2.11: (a) ISCAS modified benchmark circuit C17 (added 3 input AOI21 gates) [28]  
(b-e) Comparison of results obtained using existing (SIS), SPICE, and proposed models.

Figure 2.11b and 2.11c 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). Similarly, Fig. 2.11d and Fig. 2.11e compare the corresponding results for the gate G6 (port Y6). It is observed that, in general, the ML model is able to match the SPICE simulation in region R2, in which the MIS effect is dominant. For quantitative comparison, the RRMSE is shown in Table 2.6 for the SIS model and the proposed model. It is observed that, on average, ignoring MIS effects incurs an error of 43.29% in the timing attributes, while the proposed model predicts them with an error of 4.20%.

The above results demonstrate that MIS-induced delay variations are effectively

Table 2.6: Evaluation on Complex Gate (for region  $R2$ )

| Metric         | Y5       |        | Y6       |        | Average       |
|----------------|----------|--------|----------|--------|---------------|
|                | $\tau_f$ | $S_f$  | $\tau_r$ | $S_r$  |               |
| $RRMSE_{SIS}$  | 43.54%   | 85.62% | 17.73%   | 24.30% | <b>43.29%</b> |
| $RRMSE_{Pred}$ | 3.88%    | 4.36%  | 3.79%    | 4.80%  | <b>4.20%</b>  |

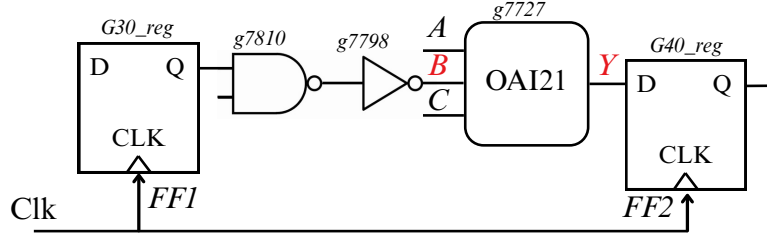


Figure 2.12: Relevant portion of the circuit s1196 for hold violation

captured by the proposed model across a wide range of gates, transitions, temporal distances, slews, loads, and instantiating conditions. Additionally, a significant speed-up in complex logic gates is highlighted by these results, which renders conventional SIS delay models highly unrealistic for MIS scenarios. This discrepancy can result in circuit failures due to incorrect early analysis, despite conservative strategies being adopted by STA tools. In contrast, the proposed model accurately captures MIS effects, making it suitable for timing analysis in industrial designs.

### 2.7.3 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 go undetected. This oversight may result in post-silicon functional errors, underscoring the importance of incorporating MIS-aware delay models into STA flows. To demonstrate the practical significance of MIS modeling, an example from the ISCAS'89 benchmark circuit s1196 [28], shown in Fig. 2.12 and mapped to the ASAP 7nm PDK [27], is considered. STA is performed using delay values extracted under the SIS model. Then, an OAI21 cell is observed: input B has an input slew of 7 ps, input C has an input 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 reduces the effective delay of the OAI21 gate. This be-

Table 2.7: 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 |

havior is confirmed by SPICE simulations, which report the actual delay for the B-to-Y arc under MIS conditions as 49 ps, resulting in a revised slack of -22 ps and indicating a previously undetected hold violation. To address this, the proposed ML-driven delay model is used, as 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, which closely matches the SPICE-derived value, and the corresponding slack is -21 ps. This demonstrates that MIS-induced delay variation is captured with high accuracy by the ML-augmented library, enabling STA tools to identify violations that would otherwise be overlooked when SIS-based models are used.

#### 2.7.4 Evaluation of Runtime Performance

The runtime performance of various computational tasks within the proposed framework was evaluated on a Linux platform equipped with a 1.8 GHz processor, four cores, and 4 GB of RAM, and the results are summarized in Table 2.7.

During library creation, approximately 12 hours were required by SPICE simulations to generate a dataset of 43,374 data points for the 10 cells shown in Table 2.4. The training of the ANN model took around 10 minutes using the 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 the various techniques discussed in the previous sections. Importantly, data collection is a one-time process for a given library (similar to conventional library characterization), and it enables reuse of the model across multiple designs, thereby reducing the overall cost of developing MIS-aware libraries.

During inference, MIS-aware delay for 10,000 queries was predicted by the ANN model in approximately 0.51 seconds. In contrast, querying the same number of MIS-aware delays using SPICE simulations required around 2.8 hours. These results demonstrate that the proposed model is approximately  $\sim 10,000\times$  faster than SPICE simulations during inference while maintaining reasonable accuracy, making it suitable for



incorporating MIS effects in STA for industrial designs.

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

In total, 506 MIS-relevant patterns were produced by the MIS pattern extractor for these 50 multi-input Boolean functions, with an overall SAT solver runtime of approximately 2.64 seconds. MIS-relevant patterns are identified efficiently by the SAT solver, even as cell complexity increases. For example, consider 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 seconds. This demonstrates that the proposed framework is sufficiently efficient 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, nearly a  $\sim 10,000\times$  reduction in ML model development effort is achieved by the proposed approach. This represents a significant improvement over conventional ML-driven techniques, which typically require modeling every possible combination of MIS transitions per cell.

## 2.8 Conclusion

In this chapter, the challenge of developing MIS-aware library models for complex logic gates is addressed. The proposed approach significantly reduces computational overhead by performing quick logical analysis to identify transitions likely to exhibit MIS-induced speed-up, thereby eliminating the need for extensive SPICE simulations

in most cases. Intelligent dataset generation is employed to avoid 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 is used to enhance 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 tighter integration of the STA tool with the library models and on conducting experiments on larger industrial designs to further validate LiMo's efficacy across the entire design flow. Tightly integrating the proposed methodology into commercial STA tools such as PrimeTime and Tempus would require parsing the augmented MIS libraries and extracting the associated ML models. These tools could then incorporate ML models into their native delay computation engines to handle MIS scenarios by adjusting SIS delays based on predicted MIS delays. However, the most effective integration strategy would ultimately depend on the tool's internal architecture and implementation details.

A current limitation of the proposed work is that only pairwise switching is considered, and three or more inputs switching simultaneously are not explicitly handled. 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 required.

It is important to note that simultaneous switching of three or more inputs does not necessarily imply 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 scenarios involving three or more switching inputs, the effective delay may still depend on only one or two signals (with other signals held at state-dependent constant values), making SIS or the proposed MIS delay models potentially sufficient. In practice, when the proposed model is applied to cases involving three or more input switching, all pairwise switching combinations can be considered, and the worst-case value is selected to ensure conservative timing

analysis. Nevertheless, a deeper investigation into scenarios involving three or more simultaneous switching events is needed, and future work may require extending the proposed model to handle such cases more robustly.

## CHAPTER 3

# Modeling Glitches due to Multi-input Switching (MIS) in Complex Logic Gates

This chapter focuses on an ML-driven, SAT-guided framework for modeling glitches induced by MIS in complex logic gates. It presents a systematic workflow for efficient dataset generation, ML-based glitch prediction, and the integration of these models to enable more accurate timing and noise analysis.

The organization of this chapter is as follows. Section 3.1 presents the background and related work. A top-level overview of the proposed framework is provided in Section 3.2. Section 3.3 explores the key factors influencing the characteristics of glitches caused by MIS. Section 3.4 focuses on identifying glitch-inducing input patterns by formulating the problem as a SAT problem. Section 3.5 introduces the proposed glitch modeling approach and details the methodology used for its development. Section 3.6 presents the experimental results, demonstrating the effectiveness of the proposed model across various circuit conditions. Finally, Section 3.7 concludes the chapter.

The work presented in this chapter has been published in [29].

### 3.1 Background and Related Work

In Chapter 2, the limitations of conventional technology library models were addressed in the context of MIS-induced delay variations, and a framework was proposed to accurately model these effects. Beyond delay variations, MIS can also give rise to another critical challenge: glitch generation at the gate output. When signals undergo transitions at multiple input pins, glitches can be produced at the output, as illustrated in Fig. 3.1, and this effect is the focus of the present chapter. The input at pin A transitions from  $0 \rightarrow 1$  while the input at pin B transitions from  $1 \rightarrow 0$ , resulting in a glitch at the output Y. These glitches can lead to increased dynamic power consumption, can also propagate through downstream logic, triggering unintended switching activity and potentially causing functional errors. As process technologies scale and circuits oper-

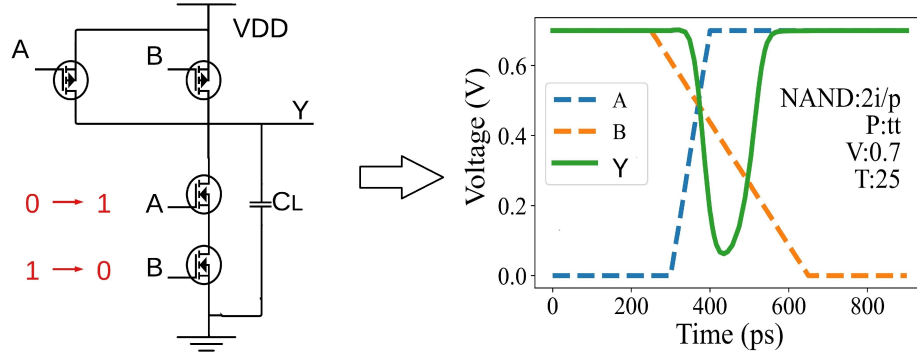


Figure 3.1: Glitch at 2 input NAND gate

ate at higher frequencies with tighter margins, the impact of MIS-induced behaviours becomes more pronounced. Therefore, capturing glitch characteristics in technology libraries is essential for ensuring the accuracy and robustness of modern timing and power analysis tools. While MIS-induced delay variations have received a great deal of attention [31]-[21], glitch modelling under MIS conditions remains relatively underexplored, particularly concerning its implications on signal integrity (SI).

Glitch modeling in existing literature has predominantly focused on power estimation, with limited attention to timing analysis or SI. Early efforts such as [34] and [35] introduced gate-level glitch models to refine estimates of switching activity. However, these models relied on simplified linear ramp approximations and neglected glitch width, failing to accurately capture the non-linear temporal behavior of real glitch waveforms. In a more timing-aware approach, [36] presented a timing-dependent power estimation framework that accounts for both glitches and coupling effects. Their study highlighted that neglecting input timing skew could lead to power estimation errors up to 25%. Despite these advancements, integrating MIS-induced glitches into gate-level timing and power analysis tools remains a challenge. The work in [37] brought attention to the inadequacy of SIS-based libraries for power estimation in the presence of multi-input switching. It demonstrated that simultaneous transitions at multiple inputs produce internal energy profiles that differ significantly from those observed under SIS assumptions, especially in designs with deep logic levels and well-balanced paths. To overcome this, [38] proposed extending standard-cell lookup tables by adding multi-input switching event data, including analog output voltage and supply current waveforms for each MIS-induced glitch scenario. While this method improves estimation accuracy, it significantly increases the computational complexity, characterization

effort, and library size due to the need to represent a large number of input-event combinations.

Additionally, several works [39], [40], [41] have focused on glitch analysis from a reliability perspective, where glitches originate from radiation-induced Single Event Transients rather than MIS. These approaches primarily target glitch propagation, timing, and masking effects at the design level using event-driven or analytical techniques. Although such methods provide accurate modeling of glitch propagation and electrical masking, they are not library-centric and do not address MIS-induced glitch generation, nor the scalability challenges associated with characterizing complex logic gates under MIS.

## 3.2 Proposed Work and Framework

A promising direction for addressing the limitations of existing glitch modeling approaches is to leverage ML. ML models are well suited to capture complex dependencies among multiple input parameters and glitch characteristics, offering a compact representation of high-dimensional data. In this context, an ANN-based glitch modeling approach capable of predicting glitch attributes was proposed in [42]. While [42] demonstrates the feasibility of using ML models for glitch modeling, several practical challenges remain to be addressed.

First, generating datasets for training ML models incurs a high computational cost. Moreover, there is a trade-off between the ML model's accuracy and the size of the training dataset. Hence, methods that reduce the effort required to build an ML model while maintaining its accuracy are critical to this approach to glitch modeling. Second, the shapes of glitches induced by MIS can vary from triangular to trapezoidal, depending on the skew of the input signals. Therefore, ML models should account for varying glitch shapes, and this variation can be leveraged to optimize model creation. Third, ML-driven modeling of MIS-induced glitches in complex logic gates with many inputs remains largely unexplored. As the number of inputs increases, the number of MIS combinations grows exponentially, rendering exhaustive SPICE simulations. Hence, creating datasets and developing ML models for complex logic gates becomes challenging. This work aims to address the above challenges in developing ML models

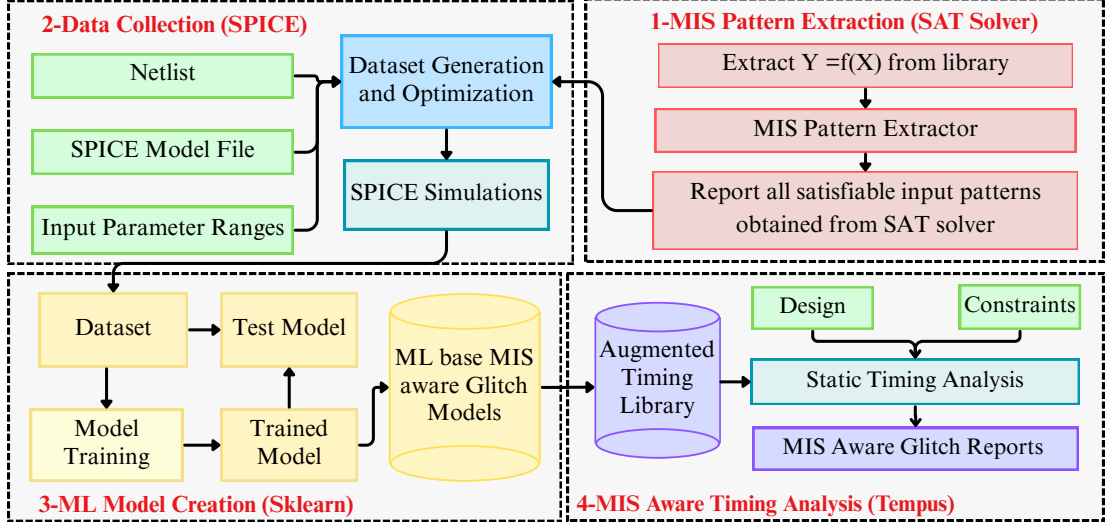


Figure 3.2: Proposed Methodology: SAT-Guided Framework for MIS Glitch Modeling

for MIS-induced glitches by pruning the search space, generating minimal yet effective training datasets, exploring and tuning various ML models, and integrating the resulting models into standard cell libraries. The proposed framework is further examined for applications such as glitch attenuation, delay computation, and SI analysis.

In this work, a methodology is proposed to efficiently develop and deploy ML models for MIS-induced glitches in complex logic gates as shown in Fig. 3.2. The overall framework for MIS-aware glitch characterization follows the same LiMo methodology presented in Chapter 2 for delay characterization. Accordingly, Fig. 3.2 is structurally identical to Fig. 2.2, with the primary difference being that the ML model is trained to predict glitch characteristics rather than delay. The novelty of this work does not lie in the ANN architecture itself, but rather in a systematic methodology for glitch modeling that integrates SAT-based logical analysis with ML. SAT solvers are leveraged to extract glitch-prone transitions from among all MIS possibilities. This search-space pruning significantly reduces the number of SPICE simulations required for dataset generation and enhances the scalability of glitch-aware modeling, particularly for gates with many inputs. Additionally, the glitch shape is explored across an extended window of skew for incoming signals, and both triangular and trapezoidal glitch waveforms are captured. Moreover, an efficient dataset generation strategy is proposed that dynamically adjusts and refines sampling in glitch-prone regions. The complete workflow is composed of a well-defined pipeline for SAT-based pattern extraction, dataset optimization, and ML model training. The resulting models are compact, accurate, and generalizable, making them suitable for integration into glitch-aware cell libraries, SI analysis, and

design-level transformations.

A top-level view highlighting the major tasks in the proposed methodology is shown in Fig. 3.2. For a given complex gate, MIS-relevant patterns for generating glitches are extracted by formulating this task as a SAT problem and employing a highly efficient SAT solver [23]. For each identified pattern, data is collected using SPICE [24] simulations to capture the impact of MIS on the glitches. This data is subsequently used to experiment with and create an augmented technology library, by leveraging ML techniques through the scikit-learn libraries [25]. Thereafter, MIS-aware STA is performed using an industrial tool [6], which is loosely coupled with the generated libraries.

### 3.3 Design Attributes Impacting Glitch Generation

This work is focused on two critical attributes of glitches: height ( $G_H$ ) and area ( $G_A$ ). The parameter  $G_H$  is defined as the peak magnitude of the glitch waveform, while  $G_A$  quantifies the total area of the glitch waveform. These attributes are used to characterize glitches and are employed as labels in the proposed ML model. To effectively model these characteristics, features are selected through systematic experiments, which capture essential aspects of the dataset and establish significant correlations with the labels. In the following paragraphs, findings are presented for a 2-input NAND gate with input pins  $A$  and  $B$ , along with the output pin  $Y$ . The same approach is applied to other logic gates, thereby ensuring consistency and validation across different gate types. All experiments in this section are conducted using the ASAP 7nm process design kit (PDK) [27] and Virtuoso [24]. These experiments are used to explain how design attributes influence glitch generation, providing critical insights for developing the proposed ML model.

#### 3.3.1 Transition Time ( $t_t$ )

Figure 3.3a and 3.3b illustrate the behavior of  $G_H$  and  $G_A$  for a scenario in which signal  $A$  transitions from  $0 \rightarrow 1$ , while signal  $B$  transitions from  $1 \rightarrow 0$ . These figures display the variations of  $G_H$  and  $G_A$  across different transition times ( $t_t$ ). It is observed that both  $G_H$  and  $G_A$  become prominent when the transition time of signal  $A$  ( $t_A$ ) is low, while the transition time of signal  $B$  ( $t_B$ ) is high. This observation implies that, for a



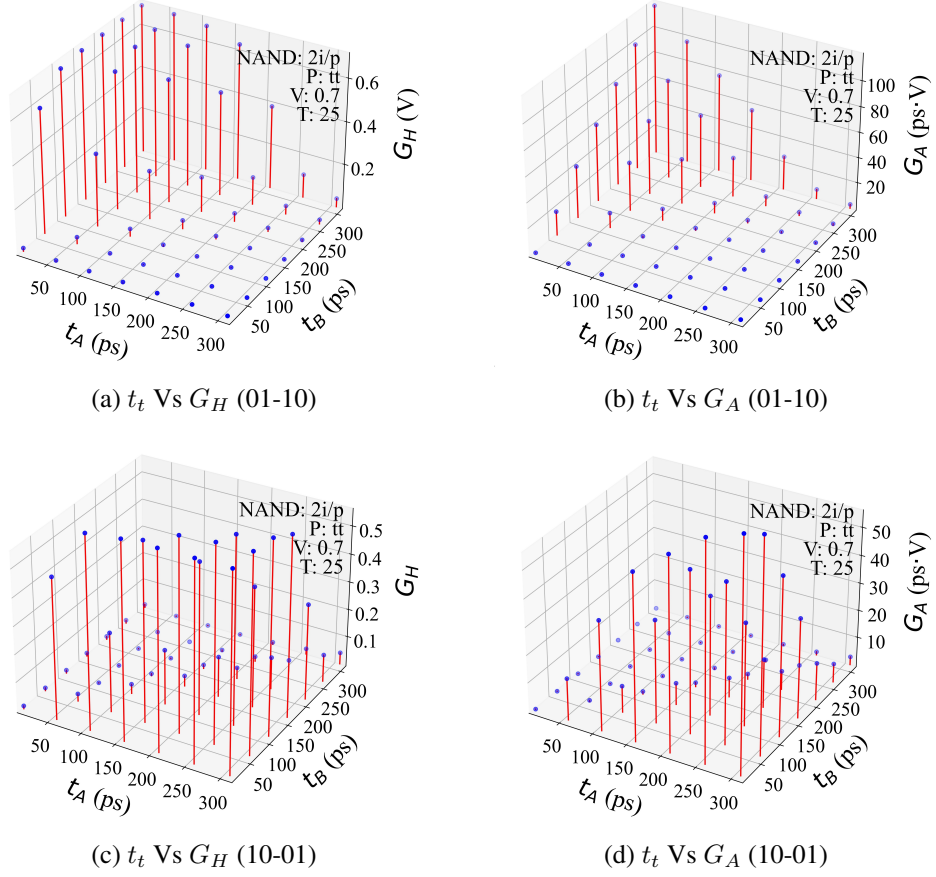


Figure 3.3: Correlation of  $G_H$  and  $G_A$  with  $t_t$

significant glitch to appear,  $t_A$  must be smaller than  $t_B$ .

Similarly, Fig. 3.3c and 3.3d present the changes in  $G_H$  and  $G_A$  for the case in which signal  $A$  transitions from  $1 \rightarrow 0$ , while signal  $B$  transitions from  $0 \rightarrow 1$ , for various transition times ( $t_t$ ). In this scenario, the trends are reversed. Large values of  $G_H$  and  $G_A$  are observed when  $t_A$  is high and  $t_B$  is low. For a significant glitch to appear in this case, the transition time  $t_A$  must be larger than  $t_B$ . These patterns suggest that  $t_t$  can serve as an effective feature for modeling  $G_H$  and  $G_A$ .

### 3.3.2 Temporal Distance ( $t_D$ )

In Fig. 3.4a, the impact of the temporal distance  $t_D$  on the characteristics of the glitch observed at the output of the NAND gate is shown. The temporal distance  $t_D$  is defined as the difference between the arrival times of transitions at pins  $A$  and  $B$ :  $t_D = AT_B - AT_A$  (also referred to as the skew between the signals). When a transition at  $B$  occurs before  $A$ , no glitch is observed at the output. The early arrival of the transition

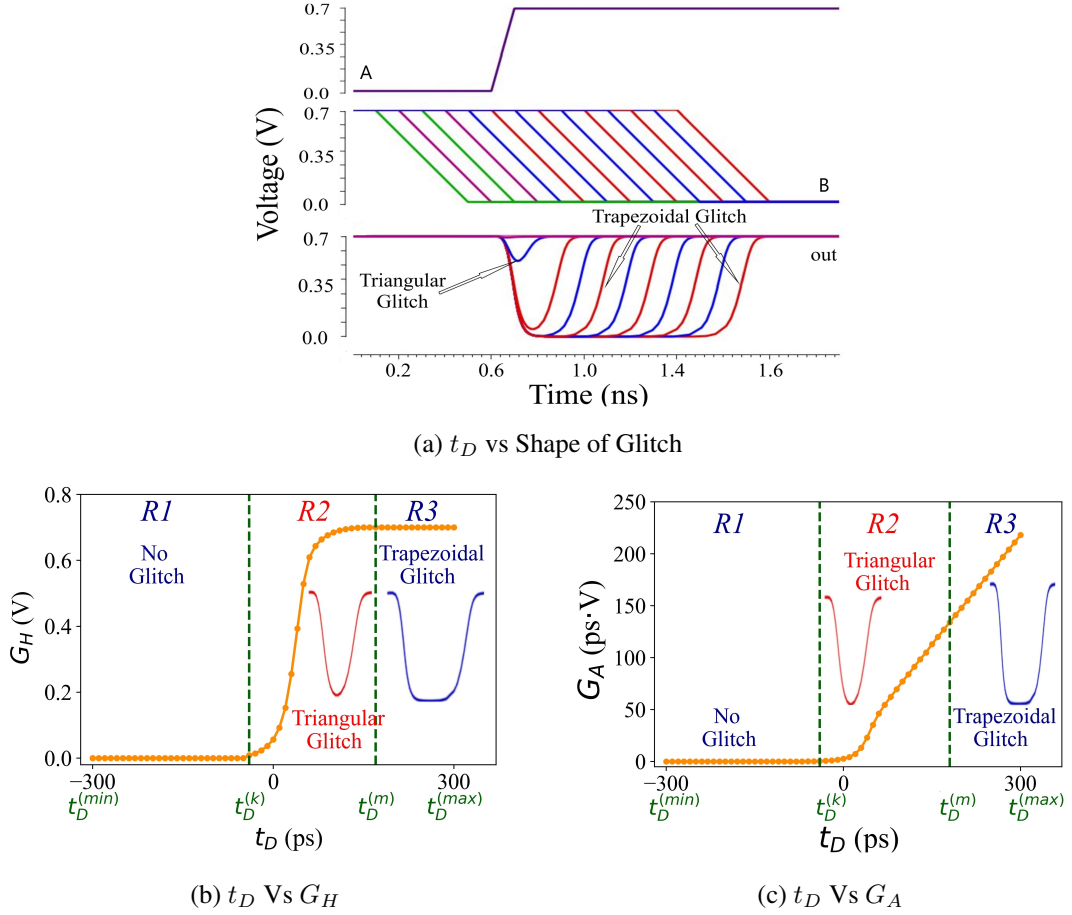


Figure 3.4: Correlation of  $G_H$  and  $G_A$  with  $t_D$

at  $B$  ensures that there is no time instant at which both  $A$  and  $B$  are simultaneously ON (at voltage  $V_{DD}$ ), thereby preventing the occurrence of a glitch at the gate output. However, a glitch is observed when the transition at  $A$  arrives simultaneously with or later than the transition at  $B$ . Initially, with an increase in  $t_D$ , a triangular glitch shape is observed. As  $t_D$  continues to increase, the glitch height increases until it reaches  $V_{DD}$ . Further increases in  $t_D$  result in the transformation of the glitch shape into a trapezoid.

In Fig. 3.4b and 3.4c, the effects of varying  $t_D$  from  $t_D^{(min)} = -300$  ps to  $t_D^{(max)} = +300$  ps on both  $G_H$  and  $G_A$  are examined. Three distinct regions are identified in the analysis, each characterized by unique patterns in the behavior of  $G_H$  and  $G_A$ :

1. **Region 1 (R1):** In this region, where  $t_D^{(min)} < t_D < t_D^{(k)}$ , no glitch is observed, resulting in negligible values for both  $G_H$  and  $G_A$ .
2. **Region 2 (R2):** For  $t_D^{(k)} \leq t_D \leq t_D^{(m)}$ , a triangular glitch is observed, with both  $G_H$  and  $G_A$  increasing as  $t_D$  increases.
3. **Region 3 (R3):** When  $t_D^{(m)} < t_D < t_D^{(max)}$ , a trapezoidal glitch shape is observed.

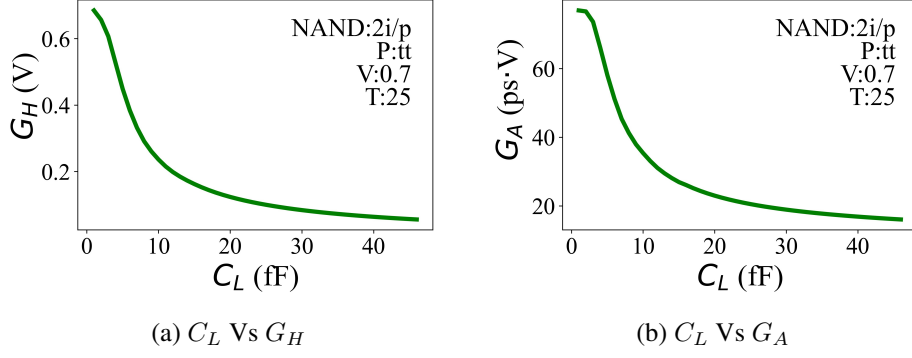


Figure 3.5: Correlation of  $G_H$  and  $G_A$  with  $C_L$

In this region, the glitch height  $G_H$  stabilizes at  $V_{DD}$ , while  $G_A$  continues to increase with increasing  $t_D$ .

The method used to derive the parameters  $t_D^{(min)}$ ,  $t_D^{(max)}$ ,  $t_D^{(k)}$ , and  $t_D^{(m)}$  is described in Section 3.4.1.

These observations reveal a strong correlation between  $t_D$  and the glitch characteristics  $G_H$  and  $G_A$ . Additionally, they suggest that glitches can be modeled as triangular or trapezoidal shapes, depending on the dynamics involved in their formation.

### 3.3.3 Capacitive Load ( $C_L$ )

Figure 3.5a and 3.5b show the relationship between capacitive load ( $C_L$ ) and the glitch characteristics. With increasing  $C_L$ , both  $G_H$  and  $G_A$  consistently decrease, indicating a negative correlation between  $C_L$  and the glitch metrics. This trend suggests that higher values of  $C_L$  introduce damping effects on glitch characteristics, thereby reducing  $G_H$  and  $G_A$ .

The above observations suggests that  $t_t$ ,  $t_D$ , and  $C_L$  can be taken as input features in modeling  $G_H$  and  $G_A$ . Other potential factors, such as input logic state or gate type, are implicitly accounted for by training separate models per gate type and switching scenario. By focusing on these features, the core dependencies governing glitch behavior are captured, ensuring a compact yet expressive input space for ML-driven glitch modeling.

The methodology for developing the proposed ML glitch model using these design attributes is outlined in the next section.

### 3.4 Identification of Glitch-Inducing MIS Input Patterns in Multi-Input Logic Gates

To identify MIS patterns that can induce glitches at the output, the problem is modeled as a SAT problem as follows. Let the logic function of a gate be defined as:

$$Y = f(X), \quad \text{where } X = \{x_1, x_2, \dots, x_N\} \text{ is the set of inputs.} \quad (3.1)$$

A transition is represented as  $X' \rightarrow X''$ , indicating a change in the input vector from state  $X' = \{x'_1, x'_2, \dots, x'_N\}$  to  $X'' = \{x''_1, x''_2, \dots, x''_N\}$ .

A glitch can be induced at the output when a pair of inputs  $(x_i, x_j)$  toggles and the following Boolean expression is evaluated to be true:

$$G(X', X'', x_i, x_j) = M(X', X'', x_i, x_j) \cdot T(X', X'') \cdot S(X', x_i, x_j) \quad (3.2)$$

Each term on the right-hand side of the expression has the following meaning and significance:

- **Pairwise Input Toggle Constraint:** The following expression is used to capture that two inputs  $x_i$  and  $x_j$  toggle their values, while all other inputs are held constant:

$$M(X', X'', x_i, x_j) = (x'_i \oplus x''_i)(x'_j \oplus x''_j) \cdot \prod_{k \neq i, j} \neg(x'_k \oplus x''_k) \quad (3.3)$$

This constraint arises from the MIS scenario considered in this work, where only two inputs are assumed to switch simultaneously.

- **Output Remains Constant Constraint:** A glitch causes a temporary deviation in the output, while the initial and final output values remain the same. Accordingly, the following expression is used to enforce the condition that the output does not change across the transition:

$$T(X', X'') = \neg(f(X') \oplus f(X'')) \quad (3.4)$$

- **Sensitivity Constraint:** A glitch can occur at the output if sensitivity to either of the toggling inputs exists. The sensitivity of a Boolean function to a specific input is determined using its Boolean difference with respect to that variable. By computing the logical OR of the Boolean differences with respect to both toggling inputs, the possibility of a glitch at the output is determined:

$$S(X', x_i, x_j) = \left( \frac{\partial Y}{\partial x_i} \vee \frac{\partial Y}{\partial x_j} \right) \Big|_{X=X'} \quad (3.5)$$

The Boolean difference of a function  $Y$  with respect to a Boolean variable  $x_i$  is defined as:

$$\frac{\partial Y}{\partial x_i} = f(X_{x_i=0}) \oplus f(X_{x_i=1}) \quad (3.6)$$

where  $X_{x_i=0}$  and  $X_{x_i=1}$  denote the input vector  $X$  with  $x_i$  set to 0 and 1, respectively, while all other inputs are kept unchanged.

To identify all possible glitch-inducing MIS input patterns using a SAT solver, a logical OR is applied over all valid input pairs:

$$G(X', X'') = \bigvee_{(i,j)} G(X', X'', x_i, x_j) \quad (3.7)$$

Subsequently,  $G(X', X'')$  is converted to Conjunctive Normal Form (CNF) using Tseitin transformation and is passed to a SAT solver to identify all satisfying assignments. These solutions are then decoded to extract all valid glitch-producing MIS input patterns  $(X', X'')$ . In this work, MiniSat is used for extracting the satisfying assignments [23].

### 3.4.1 Illustration

The application of the proposed Boolean formulation is demonstrated and the results are validated using SPICE simulations for two representative gates: an AOI gate defined as  $Y = \neg(A \cdot B + C)$  and an OAI gate defined as  $Y = \neg((A + B) \cdot C)$ . For an  $n$ -input gate, there are  $2^n$  unique input combinations, resulting in  $2^n \times (2^n - 1)$  possible transitions when self-transitions (i.e.,  $X' \neq X''$ ) are excluded. Thus, for the 3-input AOI and OAI gates, a total of 56 transitions are possible.

Table 3.1: Extraction of Glitch-Inducing MIS Input Patterns in Multi-Input Gates

(a) AOI-21

| $X'$ | $X''$ | $Y'$ | $Y''$ | Waveform                                                                          | $G(X', X'') = 1$ |
|------|-------|------|-------|-----------------------------------------------------------------------------------|------------------|
| 010  | 110   | 1    | 1     |  | Yes              |
| 110  | 010   | 1    | 1     |  | Yes              |
| 011  | 110   | 0    | 0     |  | Yes              |
| 110  | 011   | 0    | 0     |  | Yes              |
| 011  | 101   | 0    | 0     |  | No               |
| 101  | 011   | 0    | 0     |  | No               |
| 001  | 111   | 0    | 0     |  | No               |
| 111  | 001   | 0    | 0     |  | No               |
| 101  | 110   | 0    | 0     |  | Yes              |
| 110  | 101   | 0    | 0     |  | Yes              |

(b) OAI-21

| $X'$ | $X''$ | $Y'$ | $Y''$ | Waveform                                                                            | $G(X', X'') = 1$ |
|------|-------|------|-------|-------------------------------------------------------------------------------------|------------------|
| 011  | 101   | 0    | 0     |    | Yes              |
| 101  | 011   | 0    | 0     |    | Yes              |
| 010  | 100   | 1    | 1     |    | No               |
| 100  | 010   | 1    | 1     |    | No               |
| 000  | 110   | 1    | 1     |    | No               |
| 110  | 000   | 1    | 1     |  | No               |
| 001  | 100   | 1    | 1     |  | Yes              |
| 100  | 001   | 1    | 1     |  | Yes              |
| 001  | 010   | 1    | 1     |  | Yes              |
| 010  | 001   | 1    | 1     |  | Yes              |

Out of these, 10 transitions satisfy the condition of pairwise input switching (i.e., exactly two inputs toggle) while the output remains constant. These transitions are listed in Table 3.1, shown separately for each gate. Each row corresponds to a transition  $X' \rightarrow X''$ , and the SPICE-simulated waveform for each case is presented in the corresponding column. For each such transition, the temporal distance  $t_D$  between the toggling inputs (e.g., A–B, A–C, B–C) is varied and the occurrence of a glitch is observed.

The final column of the table indicates whether the transition satisfies the Boolean formulation  $G(X', X'') = 1$ , signifying a predicted glitch. Out of the 10 transitions, 6 are observed to exhibit glitches, as confirmed by both the SAT-based prediction and the SPICE waveform results. These observations validate the effectiveness and accuracy of the proposed SAT-based method for identifying glitch-prone transitions in multi-input gates.

A limitation of the above approach is that the focus is restricted to pairwise input transitions, without explicitly handling cases in which three or more inputs switch si-

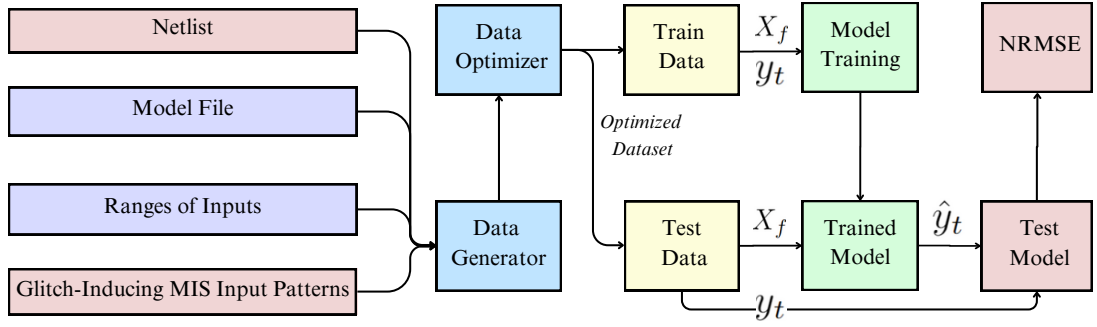


Figure 3.6: Overall flow of modeling glitches using ML

multaneously. While such events are less common, they can still occur in real circuits. Therefore, a more thorough analysis of higher-order switching is required. Future work will aim to extend the model to capture these cases and to assess their impact on glitch behavior.

## 3.5 Modeling Glitches using ML

The methodology for developing the proposed ML glitch model is outlined in Fig. 3.6. Initially, the range of stimulus (input) parameters is defined, and the combinational circuit's transistor-level model file and netlist are provided to the data generator. These inputs are used by the data generator, in conjunction with the data optimizer, to construct an optimized dataset. Once the dataset is generated, it is split into training and test sets. The model is then trained using the training set and subsequently evaluated on the test set. This structured approach ensures that glitch behavior can be accurately predicted for unseen data. The tasks shown in Fig. 3.6 are described in detail in the following paragraphs.

### 3.5.1 Dataset Generation and Optimization

The dataset is constructed by combining features ( $t_A$ ,  $t_B$ ,  $C_L$ , and  $t_D$ ) and labels ( $G_H$  and  $G_A$ ). Comprehensive data for each combinational cell are systematically extracted through SPICE simulations using a data generator algorithm. For the features  $t_A$ ,  $t_B$ , and  $C_L$ , the starting ( $x_{\text{start}}$ ) and ending ( $x_{\text{end}}$ ) points are selected from the standard cell library. Intermediate values for these parameters are generated using a geometric progression (GP), formulated as  $x_n = a \cdot r^{n-1}$ , where  $x_n$  represents the  $n$ -th value in the

progression,  $a = x_{\text{start}}$  denotes the starting value, and  $r$  is a user-defined ratio. The progression is continued until  $x_n$  exceeds  $x_{\text{end}}$ . This method enables the generation of a desired number of intermediate values between  $x_{\text{start}}$  and  $x_{\text{end}}$  by varying the ratio  $r$ , while providing fine-grained control over the distribution of values within the range.

For  $t_D$ , after extensive experimentation, an arithmetic progression with user-defined minimum ( $t_D^{(\min)}$ ) and maximum ( $t_D^{(\max)}$ ) values is employed. The values are uniformly distributed between these limits, as given by

$$t_D^{(i)} = t_D^{(\min)} + (i - 1) \cdot \Delta t_D, \quad (3.8)$$

$$\Delta t_D = \frac{t_D^{(\max)} - t_D^{(\min)}}{M}, \quad (3.9)$$

where  $i = 1, \dots, M$ , and  $\Delta t_D$  denotes the step size. The total number of divisions  $M$  is selected to ensure adequate granularity in modeling MIS effects..

As discussed in Section 3.2.1, glitch characteristics are strongly affected by the relative sharpness of the input transitions. Significant glitches are more likely to be observed when one input transitions sharply (low  $t_t$ ) and the other transitions slowly (high  $t_t$ ). For fast transitions, even small changes in the transition time can alter glitch characteristics. This high sensitivity in the lower transition-time range necessitates capturing finer variations in this region. Similarly, as demonstrated in Section 3.2.3, an inverse relationship exists between  $C_L$  and glitch behavior, wherein increases in  $C_L$  lead to consistent reductions in both  $G_H$  and  $G_A$  due to damping effects. MIS-induced glitches therefore have a stronger impact at lower values of  $C_L$ . By sampling  $t_t$  and  $C_L$  using a GP, a higher density of samples is placed in the lower range and fewer samples in the higher range. For  $t_D$ , an arithmetic progression is first employed to uniformly sweep from  $t_D^{(\min)}$  to  $t_D^{(\max)}$ . This uniform sampling facilitates the identification of the three glitch regions R1, R2, and R3. After these regions are identified, the proposed data optimizer directs SPICE simulations exclusively to the R2 region, where sharp variations in glitch characteristics are observed.

The data generator algorithm iterates over all combinations of  $t_A$ ,  $t_B$ ,  $C_L$ , and  $t_D$  based on the methodology described above. For each combination, a simulation is performed. The simulated data, including the input parameters ( $t_A$ ,  $t_B$ ,  $C_L$ , and  $t_D$ ) and the resulting glitch characteristics ( $G_H$  and  $G_A$ ), are then written to an output file for



subsequent processing.

We also employ a data optimizer algorithm that generates an optimized dataset to reduce characterization efforts. As described in Section 2, sweeping the parameter  $t_D$  over its entire range reveals three distinct regions of glitch behavior (R1, R2, and R3). The algorithm is designed to identify the boundaries between these regions by considering a range of  $t_D \in [t_{D,\min}, t_{D,\max}]$ .

The parameter  $t_D$  is explored from 0 to  $t_{D,\min}$  with a step size of  $\Delta t_D$ . For each value of  $t_D^{(i)}$ , a SPICE simulation is performed to observe glitch occurrence and to measure the corresponding  $G_H(t_D^{(i)})$ . The boundary between Region R1 and Region R2 is identified when the following condition is satisfied for four consecutive values of  $t_D$ :

$$G_H(t_D^{(i)}) \leq \alpha \times V_{DD}, \quad \alpha = 0.05, \quad i = k, k+1, k+2, k+3. \quad (3.10)$$

The value of  $t_D$  at which this condition is first satisfied is denoted as  $t_D^{(k)}$ .

Subsequently, the parameter  $t_D$  is explored from 0 to  $t_{D,\max}$  with the same step size  $\Delta t_D$ . For each value  $t_D^{(i)}$ , a SPICE simulation is carried out and the corresponding  $G_H(t_D^{(i)})$  is measured. The boundary between Region R2 and Region R3 is identified when the following condition holds for four consecutive values of  $t_D$ :

$$G_H(t_D^{(i)}) \geq \beta \times V_{DD}, \quad \beta = 1, \quad i = m, m+1, m+2, m+3. \quad (3.11)$$

The value of  $t_D$  at which this condition is first satisfied is denoted as  $t_D^{(m)}$ .

For  $t_D \in [t_D^{(\min)}, t_D^{(k)}]$ , the glitch metrics satisfy  $G_H \approx 0$  and  $G_A \approx 0$ . As no significant glitch behavior is observed in this region (R1), simulations are not performed. For  $t_D \in [t_D^{(k)}, t_D^{(m)}]$ , both  $G_H$  and  $G_A$  exhibit significant variation. In this region (R2), glitch behavior transitions from negligible to substantial, thereby requiring dense simulations to accurately capture the glitch characteristics. For  $t_D \in [t_D^{(m)}, t_{D,\max}]$ , the glitch height stabilizes at  $G_H \approx V_{DD}$ . Although  $G_H$  remains constant in this region,  $G_A$  varies approximately linearly with  $t_D$ . Since the glitch behavior in this region (R3) is predictable, additional simulations are not required.

By omitting simulations in Regions R1 and R3 and focusing simulation efforts exclusively on Region R2, characterization and model training efforts are significantly reduced while maintaining comprehensive coverage of glitch behavior. This strategy

ensures that variations in  $t_D$  are systematically explored and that glitch occurrences are accurately characterized, resulting in an optimized dataset for glitch estimation.

### 3.5.2 Model Training

The dataset used for model training is generated using the procedure described in Section 3.5.1. Briefly, the input features  $((t_A), (t_B), (C_L), \text{ and } (t_D))$  are systematically generated over the specified operating ranges, and SPICE simulations are performed to obtain the corresponding glitch characteristics  $((G_H) \text{ and } (G_A))$ . The generated data are then processed using the proposed data optimization methodology, which selectively retains samples from the most informative glitch region (R2), resulting in an optimized dataset that is subsequently used for training and testing of the ML models. The dataset, generated through data generation and optimization, is preprocessed and normalized using min–max scaling to the range  $[0, 1]$ . Here,  $X_f \in R^{n \times m}$  represents the feature matrix and  $y_t \in R^n$  the target labels. The dataset is then split into training  $D_{\text{train}}$  and testing  $D_{\text{test}}$  sets.

Various ML models with different levels of complexity are trained using the training dataset  $((D_{\text{train}}))$ . Candidate models are evaluated to identify an appropriate balance between prediction accuracy, computational cost, and model complexity. For each model, hyperparameters are optimized using a grid search over a predefined search space. To improve model robustness and reduce the risk of overfitting, (k)-fold cross-validation is employed, wherein the training dataset is partitioned into (k) folds, with each fold used once for validation and the remaining folds used for training. The detailed model training methodology, candidate ML models, hyperparameter optimization strategy, and model selection procedure are described in Section 3.6.1.

Model performance is evaluated using the NRMSE. The RMSE is computed as:

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

and normalized by the range of target values:

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

Lower NRMSE values indicate better predictive performance and generalization ability.

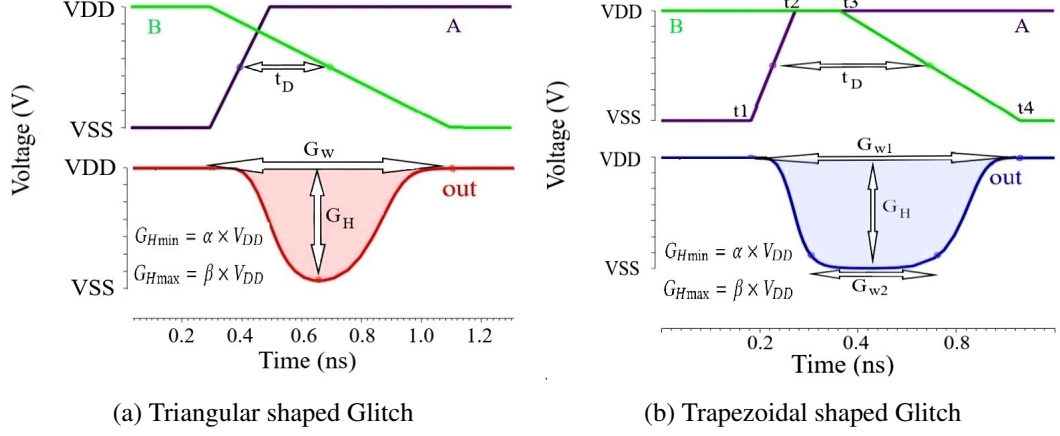


Figure 3.7:  $t_D$  vs Shape of Glitch

### 3.5.3 Deriving the Glitch Width ( $G_W$ )

The trained model is used to provide the predicted glitch height ( $G_H^{\text{Pred}}$ ) and glitch area ( $G_A^{\text{Pred}}$ ), which are subsequently utilized to derive the predicted glitch width ( $G_W^{\text{Pred}}$ ) using the equations described in the following paragraphs. As discussed in Section 3.2.2, the glitch behavior is classified into three distinct regions:  $R_1$ , in which both  $G_H$  and  $G_A$  are negligible;  $R_2$ , which corresponds to transitional conditions resulting in a triangular-shaped glitch; and  $R_3$ , in which  $G_H$  saturates while  $G_A$  varies linearly. To model  $G_W$  across these regions, two thresholds,  $G_{H\min} = \alpha \times V_{DD}$  and  $G_{H\max} = \beta \times V_{DD}$ , are employed, where  $\alpha = 0.05$  and  $\beta = 1$ . The parameters  $\alpha$  and  $\beta$  are user-defined, thereby providing flexibility in modeling the glitch width. Based on these thresholds, the glitch width ( $G_W$ ) for the three regions is computed as follows.

In region  $R_1$ , where  $G_H \leq G_{H\min}$  and  $G_A$  is negligible, the glitch width is taken as:

$$G_W = 0. \quad (3.12)$$

In region  $R_2$ , where  $G_{H\min} < G_H < G_{H\max}$ , the glitch waveform is approximated as a triangle (as illustrated in Fig. 3.7a), and the glitch width is calculated as:

$$G_W = \frac{2 \times G_A}{G_H}. \quad (3.13)$$

In region  $R_3$ , where  $G_H \geq G_{H\max}$ , the triangular waveform transitions to a trape-

zoidal shape, resulting in two glitch width components, as depicted in Fig. 3.7b. The computation of  $G_{W1}$  is performed by adding  $t_D$  to half of the total transition time of input  $A$  and half of the total transition time of input  $B$ , as expressed below:

$$G_{W1} = t_D + \frac{t_2 - t_1}{2} + \frac{t_4 - t_3}{2}. \quad (3.14)$$

Half of the total transition times of inputs  $A$  and  $B$  are determined using the measured transition times  $t_A$  and  $t_B$ . The transition time is measured using a linear ramp approximation, by evaluating the time interval between two points on the waveform. In this work, the transition time is measured using the 10%–90% thresholds of  $V_{DD}$ . Accordingly, the total transition times of  $A$  and  $B$  are computed as:

$$t_2 - t_1 = \frac{t_A}{0.80}, \quad t_4 - t_3 = \frac{t_B}{0.80}. \quad (3.15)$$

By combining the above expressions, the final form of  $G_{W1}$  is obtained as:

$$G_{W1} = t_D + \frac{t_A}{2 \times 0.80} + \frac{t_B}{2 \times 0.80}. \quad (3.16)$$

Similarly, the second component  $G_{W2}$  is derived by approximating the waveform as a trapezoid in region  $R_3$ , yielding:

$$G_{W2} = \frac{2 \times G_A}{G_H} - G_{W1}. \quad (3.17)$$

The glitch width  $G_W$ , characterized by triangular and trapezoidal waveform approximations, can thus be determined using the above formulations.

### 3.5.4 Integration of MIS-Aware Glitch Modeling into STA Flow

Fig. 3.8 illustrates the proposed integration of MIS-aware glitch modeling into a conventional STA workflow. The overall flow is organized into two distinct phases: (a) library augmentation and (b) STA-based timing and SI analysis. In the library augmentation phase, user inputs and SPICE-level netlists of standard cells are used to train ML-driven glitch models that capture the behavior of glitches induced by MIS. These trained models are then incorporated into the conventional timing library. Specifically,

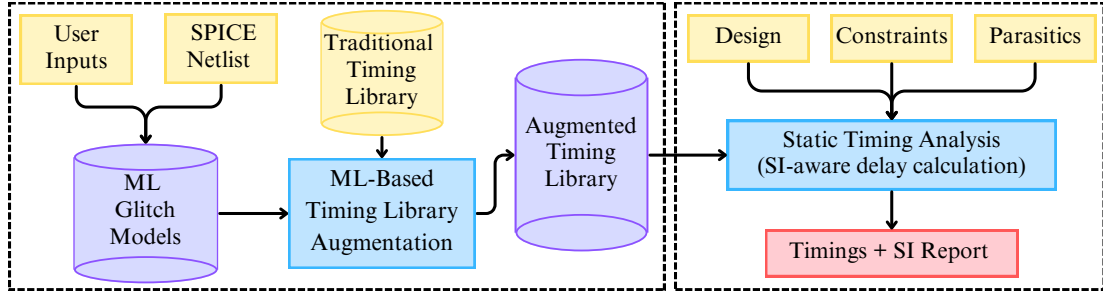


Figure 3.8: Overall flow of integration of MIS-Aware glitch modeling into STA

the conventional Liberty timing arcs are extended with MIS-specific metadata that reference the corresponding ML glitch models, resulting in an augmented timing library.

In the analysis phase, the augmented timing library is directly used by the STA tool together with the design netlist, timing constraints, and extracted parasitics. SI-aware delay calculation is performed by the STA engine, during which the MIS-related information stored in the library can be accessed to account for glitch-induced effects during timing arc evaluation. As a result, the analysis framework generates reports that include not only conventional timing information but also MIS-induced glitch attributes, thereby enabling a more accurate assessment of delay and noise effects caused by these glitches.

To enable ML-driven timing library augmentation, each standard cell in the characterized library is annotated with MIS-specific glitch information in addition to the conventional timing arcs. This is achieved by extending the standard Liberty format by including user-defined named groups within the `cell_rise` or `cell_fall` timing sections. Although these groups are not natively supported by the Liberty specification, they can be accessed and interpreted by external tools by parsing the custom attributes.

A snippet of the augmented library file in Liberty syntax is shown below:

```
pin(Y) {
  timing() {
    related_pin : "A";
    cell_rise(delay_template) {
      ...
      mis_glitch_info("AB_011_101") {
        mis_glitch_pin : "B";
        mis_glitch_model : "OAI21_AB_011_101_model.json";
      }
    }
  }
}
```

```

    }
}

```

For each output pin (e.g., pin(Y)), the timing() block specifies the related\_pin (e.g., “A”), followed by the conventional cell\_rise or cell\_fall definitions. Within these, a new user-defined named group mis\_glitch\_info (AB\_011\_101”) is introduced, where the string key “AB\_011\_101” encodes the input pins and the associated input pattern (e.g., from 011 to 101). This group further includes user-defined attributes mis\_glitch\_pin that indicates the second pin involved in creating the glitch (e.g., “B”), and mis\_glitch\_model, which points to the file (e.g., OAI21\_AB\_011\_101\_model.json) containing the pre-trained ML-driven model for predicting the glitch behavior. During STA, the tool traverses the netlist and, for each gate, queries the library to check for the presence of any MIS conditions. If an MIS transition is detected for a gate’s input, the tool predicts the glitch characteristics with the help of the referenced mis\_glitch\_model. Since the computations performed during STA are reused in glitch prediction, no additional overhead is introduced in generating features for the ML model. Subsequently, the glitch attributes predicted by the ML model can be reported to designers or utilized by other tools for tasks such as glitch attenuation and SI analysis.

At present, commercial STA tools do not support MIS-aware glitch attributes within the Liberty format. Consequently, the integration shown in Fig. 3.8 represents a prototype analysis flow. The STA path traversal, MIS-condition detection, and glitch impact evaluation are emulated using a custom analysis framework that mirrors the sequence of operations performed by conventional STA engines. Delay and noise impacts are evaluated using SPICE simulations guided by the glitch attributes predicted by the proposed ML models. This approach enables a quantitative assessment of the accuracy and effectiveness of MIS-aware glitch modeling within an STA/SI context without requiring modifications to commercial tools.

## 3.6 Results

In this section, the findings are presented, demonstrating the efficiency of the proposed ML model. All experiments were conducted using the ASAP 7nm process design kit

(PDK) [27] in conjunction with an STA tool [6] and Virtuoso [24]. The dataset was generated using the methodology described in Section 3.4.1. The parameters  $t_A$  and  $t_B$  follow a geometric progression (GP) with  $a = 5$  ps and  $r = 4$ , resulting in values of 5 ps, 20 ps, 80 ps, and 320 ps. The load capacitance  $C_L$  also follows a GP starting at  $a = 0.7$  fF with  $r = 4$ , yielding values of 0.7 fF, 2.8 fF, 11.2 fF, and 44.8 fF. The temporal distance  $t_D$  ranges from  $t_D^{(\min)} = -700$  ps to  $t_D^{(\max)} = 700$  ps, with  $M = 71$  values, resulting in intervals of  $\Delta t_D = 20$  ps. The dataset was generated for every combination of these parameters.

### 3.6.1 Gate-Level Model Evaluation

The generated dataset is partitioned into training, validation, and test sets. The training set is used for model fitting. The validation set is obtained through  $k$ -fold cross-validation, in which  $D_{\text{train}}$  is divided into  $k$  subsets  $\{D_1, D_2, \dots, D_k\}$ . The model is trained on  $k - 1$  subsets and validated on the remaining subset, enabling hyperparameter tuning and overfitting detection. A significant gap between the training and validation errors indicates that the model fits the training data well but may not generalize effectively to unseen data. The test set is used to evaluate model performance on previously unseen data; low error values indicate good generalization.

Grid search is employed for model selection and hyperparameter optimization, systematically exploring predefined parameter values to improve model performance. The evaluated models include Support Vector Regressor (SVR), Decision Trees, Random Forests, Gradient Boosting, XGBoost, AdaBoost, LightGBM, and Artificial Neural Networks (ANNs), each exhibiting varying levels of effectiveness.

### Comparative Analysis of ML Glitch Model

Table 3.2 presents a comprehensive comparison of various ML models for glitch regression across a variety of logic gates. Among these models, the SVR consistently exhibits the highest NRMSE across the training, validation, and test datasets, indicating limited capability to capture the non-linear glitch behavior. The Decision Tree model is shown to provide a marked improvement over SVR, as it captures glitch behavior more effectively due to its hierarchical decision-making. Ensemble models, such as Random

Table 3.2: Performance of Regression Models on Various Logic Gates

| Gate       | Metric | SVR    | Decision Tree | Random Forest | Gradient Boost | XG Boost | Ada Boost | Light GBM | ANN    |
|------------|--------|--------|---------------|---------------|----------------|----------|-----------|-----------|--------|
| NAND-2i/p  | Train  | 0.0719 | 0.0325        | 0.0265        | 0.0286         | 0.0303   | 0.0248    | 0.0192    | 0.0161 |
|            | Val    | 0.0727 | 0.0365        | 0.0295        | 0.0326         | 0.0317   | 0.0280    | 0.0216    | 0.0164 |
|            | Test   | 0.0738 | 0.0369        | 0.0301        | 0.0336         | 0.0345   | 0.0283    | 0.0217    | 0.0168 |
| AND-2i/p   | Train  | 0.0791 | 0.0315        | 0.0258        | 0.0287         | 0.0300   | 0.0246    | 0.0190    | 0.0161 |
|            | Val    | 0.0808 | 0.0307        | 0.0210        | 0.0286         | 0.0302   | 0.0266    | 0.0194    | 0.0181 |
|            | Test   | 0.0816 | 0.0365        | 0.0280        | 0.0333         | 0.0332   | 0.0293    | 0.0199    | 0.0187 |
| NOR-2i/p   | Train  | 0.0748 | 0.0294        | 0.0243        | 0.0321         | 0.0328   | 0.0225    | 0.0188    | 0.0175 |
|            | Val    | 0.0770 | 0.0318        | 0.0270        | 0.0339         | 0.0351   | 0.0261    | 0.0216    | 0.0189 |
|            | Test   | 0.0802 | 0.0357        | 0.0312        | 0.0354         | 0.0361   | 0.0291    | 0.0244    | 0.0193 |
| OR-2i/p    | Train  | 0.0751 | 0.0309        | 0.0185        | 0.0299         | 0.0305   | 0.0243    | 0.0188    | 0.0146 |
|            | Val    | 0.0770 | 0.0350        | 0.0195        | 0.0319         | 0.0326   | 0.0219    | 0.0191    | 0.0172 |
|            | Test   | 0.0792 | 0.0404        | 0.0206        | 0.0333         | 0.0418   | 0.0259    | 0.0204    | 0.0181 |
| XOR-2i/p   | Train  | 0.0975 | 0.0569        | 0.0513        | 0.0330         | 0.0379   | 0.0386    | 0.0289    | 0.0201 |
|            | Val    | 0.1034 | 0.0569        | 0.0511        | 0.0375         | 0.0407   | 0.0444    | 0.0325    | 0.0228 |
|            | Test   | 0.1064 | 0.0598        | 0.0544        | 0.0399         | 0.0432   | 0.0462    | 0.0358    | 0.0225 |
| XNOR-2i/p  | Train  | 0.1043 | 0.0593        | 0.0499        | 0.0384         | 0.0383   | 0.0426    | 0.0287    | 0.0249 |
|            | Val    | 0.1122 | 0.0604        | 0.0504        | 0.0406         | 0.0399   | 0.0439    | 0.0320    | 0.0230 |
|            | Test   | 0.1152 | 0.0617        | 0.0530        | 0.0404         | 0.0414   | 0.0426    | 0.0374    | 0.0270 |
| NAND-3i/p  | Train  | 0.0718 | 0.0300        | 0.0252        | 0.0252         | 0.0322   | 0.0303    | 0.0178    | 0.0118 |
|            | Val    | 0.0713 | 0.0344        | 0.0281        | 0.0281         | 0.0330   | 0.0305    | 0.0190    | 0.0146 |
|            | Test   | 0.0730 | 0.0347        | 0.0285        | 0.0285         | 0.0346   | 0.0330    | 0.0203    | 0.0163 |
| AND-3i/p   | Train  | 0.0578 | 0.0167        | 0.0146        | 0.0216         | 0.0216   | 0.0211    | 0.0163    | 0.0148 |
|            | Val    | 0.0595 | 0.0177        | 0.0155        | 0.0223         | 0.0227   | 0.0229    | 0.0161    | 0.0147 |
|            | Test   | 0.0607 | 0.0210        | 0.0190        | 0.0233         | 0.0235   | 0.0249    | 0.0171    | 0.0153 |
| NOR-3i/p   | Train  | 0.0763 | 0.0249        | 0.0210        | 0.0283         | 0.0299   | 0.0191    | 0.0159    | 0.0137 |
|            | Val    | 0.0760 | 0.0269        | 0.0226        | 0.0294         | 0.0313   | 0.0215    | 0.0179    | 0.0144 |
|            | Test   | 0.0790 | 0.0279        | 0.0233        | 0.0306         | 0.0316   | 0.0217    | 0.0182    | 0.0164 |
| OR-3i/p    | Train  | 0.0567 | 0.0202        | 0.0181        | 0.0245         | 0.0251   | 0.0178    | 0.0154    | 0.0149 |
|            | Val    | 0.0574 | 0.0209        | 0.0187        | 0.0245         | 0.0254   | 0.0179    | 0.0113    | 0.0163 |
|            | Test   | 0.0585 | 0.0210        | 0.0185        | 0.0247         | 0.0261   | 0.0179    | 0.0163    | 0.0156 |
| AOI21-3i/p | Train  | 0.0717 | 0.0347        | 0.0283        | 0.0328         | 0.0303   | 0.0255    | 0.0199    | 0.0105 |
|            | Val    | 0.0739 | 0.0388        | 0.0325        | 0.0351         | 0.0332   | 0.0299    | 0.0225    | 0.0124 |
|            | Test   | 0.0787 | 0.0412        | 0.0329        | 0.0360         | 0.0353   | 0.0329    | 0.0234    | 0.0149 |
| OAI21-3i/p | Train  | 0.0752 | 0.0316        | 0.0263        | 0.0328         | 0.0302   | 0.0238    | 0.0199    | 0.0150 |
|            | Val    | 0.0734 | 0.0327        | 0.0275        | 0.0336         | 0.0314   | 0.0262    | 0.0200    | 0.0177 |
|            | Test   | 0.0781 | 0.0348        | 0.0285        | 0.0347         | 0.0332   | 0.0270    | 0.0224    | 0.0188 |
| Average    | Train  | 0.0754 | 0.0332        | 0.0213        | 0.0296         | 0.0307   | 0.0262    | 0.0198    | 0.0158 |
|            | Val    | 0.0718 | 0.0352        | 0.0286        | 0.03150        | 0.0320   | 0.0283    | 0.0210    | 0.0172 |
|            | Test   | 0.0804 | 0.0335        | 0.0306        | 0.0328         | 0.0345   | 0.0299    | 0.0231    | 0.0183 |

Forest and Gradient Boosting, are found to further enhance performance by aggregating multiple weak learners, thereby improving accuracy and robustness. Importantly, the



training, validation, and test NRMSE values for these models remain close, suggesting good generalization and minimal overfitting.

XGBoost, an advanced gradient boosting algorithm with built-in regularization, performs slightly better than traditional Gradient Boosting across most gates, particularly for simpler structures such as NAND and AND. AdaBoost performs moderately well, though it remains slightly inferior to XGBoost and LightGBM. LightGBM demonstrates strong performance across nearly all gates, achieving consistently low NRMSE values. The use of histogram-based learning and leaf-wise tree growth contributes to its computational efficiency and predictive accuracy.

The ANN model is observed to outperform all other models across all gate types, achieving the lowest NRMSE values for the training, validation, and test sets. This behavior reflects its ability to learn complex, non-linear relationships that are critical for modeling glitch characteristics. The minimal variation observed across datasets further confirms strong generalization performance. Owing to its superior accuracy and reliability, the ANN model is selected for further analysis in this work.

### **Hyperparameter Tuning for ANN**

Table 3.3 reports the NRMSE values obtained from a systematic hyperparameter tuning of the ANN model across a wide range of hidden-layer configurations. Hyperparameter tuning is required to identify an architecture that minimizes prediction error while maintaining generalization. Unlike hit-and-trial approaches, which rely on randomly testing a few architectures, a structured grid-search strategy is adopted in this study. The process is initiated with shallow networks having limited representational capacity, and the depth and width are progressively increased to observe performance trends and identify saturation points.

For example, shallow architectures such as (10, 10) and (10, 10, 10) are observed to exhibit high training, validation, and test errors (NRMSE  $\approx 0.18$  and  $0.036$ , respectively), indicating underfitting and insufficient capacity to model the nonlinear relationships between MIS parameters and glitch characteristics. As network complexity increases, performance improves. Moderately deep networks such as (50, 50, 50), (100, 100, 100), and (150, 100, 50) achieve substantially lower and more consistent NRMSE values ( $\approx 0.0209$ ,  $0.0186$ , and  $0.0175$ , respectively), demonstrating sufficient expres-

Table 3.3: Hyperparameter Tuning Results for ANN (NAND-2i/p)

| Hidden Layer Sizes                 | Train NRMSE   | Val NRMSE     | Test NRMSE    |
|------------------------------------|---------------|---------------|---------------|
| (10, 10)                           | 0.1765        | 0.1797        | 0.1805        |
| (10, 10, 10)                       | 0.0316        | 0.0355        | 0.0357        |
| (10, 10, 10, 10)                   | 0.0251        | 0.0269        | 0.0344        |
| (50, 10)                           | 0.0233        | 0.0256        | 0.0334        |
| (50, 50)                           | 0.0236        | 0.0251        | 0.0304        |
| (50, 50, 50)                       | 0.0195        | 0.0192        | 0.0209        |
| (50, 50, 50, 50)                   | 0.0162        | 0.0168        | 0.0194        |
| (100, 50, 10)                      | 0.0157        | 0.0170        | 0.0185        |
| (100, 100, 100)                    | 0.0167        | 0.0182        | 0.0186        |
| (100, 100, 100, 100)               | 0.0165        | 0.0162        | 0.0175        |
| (250, 250)                         | 0.0270        | 0.0288        | 0.0288        |
| (500, 500)                         | 0.0175        | 0.0193        | 0.0201        |
| (500, 500, 500)                    | 0.0157        | 0.0155        | 0.0202        |
| (1000, 500, 250, 150, 100, 50, 10) | 0.0201        | 0.0216        | 0.0275        |
| (100, 100)                         | 0.0227        | 0.0220        | 0.0227        |
| (150, 150)                         | 0.0203        | 0.0203        | 0.0222        |
| (150, 100, 50)                     | 0.0161        | 0.0167        | 0.0175        |
| (150, 100, 50, 10)                 | 0.0213        | 0.0216        | 0.0225        |
| (150, 150, 150)                    | 0.0154        | 0.0147        | 0.0142        |
| (150, 150, 150, 150)               | 0.0191        | 0.0190        | 0.0194        |
| <b>(250, 150, 100)</b>             | <b>0.0161</b> | <b>0.0160</b> | <b>0.0168</b> |
| (250, 150, 100, 50, 10)            | 0.0184        | 0.0185        | 0.0189        |
| (250, 250, 250)                    | 0.0153        | 0.0145        | 0.0184        |
| (250, 250, 250, 250)               | 0.0175        | 0.0184        | 0.0194        |
| (500, 250, 150)                    | 0.0155        | 0.0159        | 0.0169        |
| (500, 250, 150, 100, 50)           | 0.0143        | 0.0142        | 0.0199        |
| (1000, 1000)                       | 0.0197        | 0.0197        | 0.0288        |
| (1000, 1000, 1000)                 | 0.0163        | 0.0168        | 0.0196        |

siveness while maintaining good generalization.

However, very deep or wide networks, such as (1000, 1000, 1000) or (1000, 500, 250, 150, 100, 50, 10), are observed to fail to provide significant improvements and, in some cases, exhibit slightly degraded test performance ( $\text{NRMSE} \approx 0.0196\text{--}0.0275$ ). This behavior reflects over-parameterization, where excess complexity fails to improve generalization and introduces unnecessary computational overhead.

Table 3.4: Performance Comparison for Glitch-Inducing MIS Input Patterns

(a) Without Optimization (WO)

| Gate      | #NGMIP | #Total Data | #Train Data | #Test Data | NRMSE  |
|-----------|--------|-------------|-------------|------------|--------|
| NAND-2i/p | 2      | 9,090       | 7,272       | 1,818      | 0.0168 |
| AND-2i/p  | 2      | 9,090       | 7,272       | 1,818      | 0.0187 |
| NOR-2i/p  | 2      | 9,090       | 7,272       | 1,818      | 0.0193 |
| OR-2i/p   | 2      | 9,090       | 7,272       | 1,818      | 0.0181 |
| XOR-2i/p  | 4      | 18,180      | 14,544      | 3,636      | 0.0225 |
| XNOR-2i/p | 4      | 18,180      | 14,544      | 3,636      | 0.0270 |
| NAND-3i/p | 6      | 27,270      | 21,816      | 5,454      | 0.0163 |
| AND-3i/p  | 6      | 27,270      | 21,816      | 5,454      | 0.0153 |
| NOR-3i/p  | 6      | 27,270      | 21,816      | 5,454      | 0.0164 |
| OR-3i/p   | 6      | 27,270      | 21,816      | 5,454      | 0.0156 |
| AOI-3i/p  | 6      | 27,270      | 21,816      | 5,454      | 0.0149 |
| OAI-3i/p  | 6      | 27,270      | 21,816      | 5,454      | 0.0188 |

(b) With Optimization (W)

| Gate      | #NGMIP | #Total Data | #Train Data | #Test Data | NRMSE  |
|-----------|--------|-------------|-------------|------------|--------|
| NAND-2i/p | 2      | 3,540       | 3,540       | 1,818      | 0.0217 |
| AND-2i/p  | 2      | 2,930       | 2,930       | 1,818      | 0.0355 |
| NOR-2i/p  | 2      | 3,156       | 3,156       | 1,818      | 0.0332 |
| OR-2i/p   | 2      | 3,250       | 3,250       | 1,818      | 0.0370 |
| XOR-2i/p  | 4      | 8,396       | 8,396       | 3,636      | 0.0453 |
| XNOR-2i/p | 4      | 7,756       | 7,756       | 3,636      | 0.0457 |
| NAND-3i/p | 6      | 11,040      | 11,040      | 5,454      | 0.0283 |
| AND-3i/p  | 6      | 15,360      | 15,360      | 5,454      | 0.0288 |
| NOR-3i/p  | 6      | 11,046      | 11,046      | 5,454      | 0.0220 |
| OR-3i/p   | 6      | 15,744      | 15,744      | 5,454      | 0.0293 |
| AOI-3i/p  | 6      | 10,272      | 10,272      | 5,454      | 0.0336 |
| OAI-3i/p  | 6      | 10,634      | 10,634      | 5,454      | 0.0316 |

**Note:** #NGMIP denotes the number of Glitch-Inducing MIS Input Patterns identified using the Proposed Boolean formulation. **WO** = Without Optimization: includes NGMIP filtering only. **W** = With Optimization: includes NGMIP filtering and region-based data optimization.

Based on these systematic observations, the architecture with three hidden layers of 250, 150, and 100 neurons achieves the lowest and most stable NRMSE values (0.0168), representing a Pareto-optimal balance between model expressiveness, generalization, and computational efficiency. By avoiding excessively large networks, training time and resource usage are reduced without compromising accuracy. This progression from shallow to deeper networks ensures that the final architecture is selected through principled evaluation, guided by performance trends across datasets rather than arbitrary hit-and-trial selection.

## Optimized Gate-Level Model Evaluation

Table 3.4 presents a performance comparison of glitch prediction models for glitch-inducing MIS input patterns, with and without optimization. Here, #NGMIP denotes the number of such patterns extracted for multi-input logic gates. Two scenarios are evaluated: *WO* (Without Optimization), in which only NGMIP-based filtering is used, and *W* (With Optimization), in which both NGMIP filtering and region-based data optimization are applied.

Comparable NRMSE values are achieved by the optimized model relative to the unoptimized version, while the dataset size and characterization time are significantly reduced. Specifically, the optimized model requires 1,03,124 training points across 12 logic gates, compared to 2,29,072 in the unoptimized case, with only a marginal increase in the average NRMSE from 0.0183 (1.83%) to 0.0329 (3.29%). The same test dataset is used for both cases to ensure a fair comparison. This reduction in the number of data points leads to a decrease in SPICE characterization time from approximately  $\sim 52$  hours to  $\sim 28$  hours and also accelerates the model training time from  $\sim 36$  minutes to  $\sim 17$  minutes. All simulations and model training are performed on a Linux platform equipped with a 1.8 GHz processor and 4 GB of RAM, without multiprocessing. The use of multiprocessing and multithreading is expected to further accelerate the training and characterization process.

### 3.6.2 Circuit-Level Model Evaluation

To validate the efficiency of the proposed methodology at the circuit level, it is applied to the ISCAS'85 C17 benchmark circuit [28], shown in Fig. 3.9a. This circuit consists of five inputs and two outputs, and the analysis focuses on two specific glitches, denoted as  $G_1$  and  $G_2$ . The evaluation framework is illustrated in Fig. 3.9b. The ISCAS'85 C17 benchmark is a well-established and widely used circuit in the EDA community for validating glitch propagation and analyzing gate-level behavior. Its small size enables detailed observation of glitch interactions while still preserving realistic timing characteristics.

First, the Boolean functions of logic gates g5 and g6 are extracted, and the glitch-inducing MIS input patterns ( $01 \rightarrow 10$  and  $10 \rightarrow 01$ ) are identified. This identification

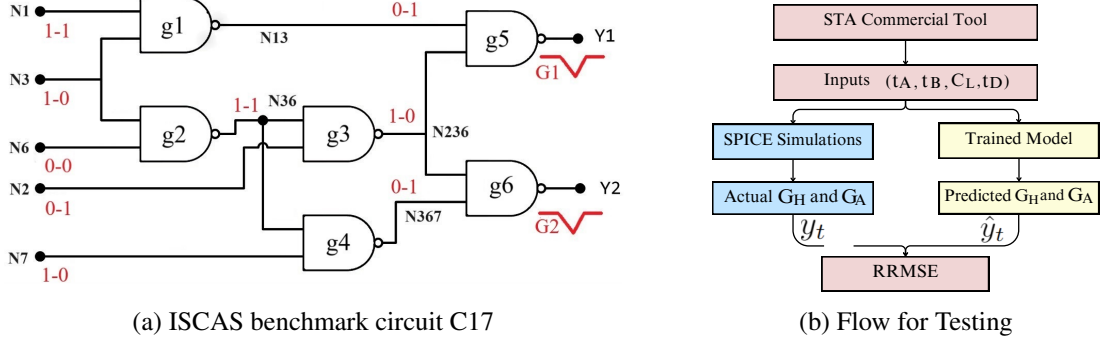


Figure 3.9: ISCAS benchmark circuit C17 and Flow for Testing.

is achieved by solving the Boolean formulation described earlier using a SAT solver. For these glitch-inducing patterns, dedicated trained ML models are already stored in the augmented library with appropriate tags. During STA, for the  $A \rightarrow B$  arc of gates  $g5$  and  $g6$ , two possible transitions exist for each gate:  $01 \rightarrow 10$  and  $10 \rightarrow 01$ . Based on the corresponding tag, the appropriate model from the library is employed for computation during STA. For demonstration, the transition  $01 \rightarrow 10$  is evaluated for the  $g5$ , and the transition  $10 \rightarrow 01$  is evaluated for the  $g6$ . STA is performed to extract the key parameters  $t_A$ ,  $t_B$ ,  $C_L$ , and  $t_D$ . These parameters are subsequently used in SPICE simulations for validation and are also provided to the trained ML model. The SPICE simulations produce the actual glitch characteristics, including glitch height ( $G_H^{At}$ ), glitch area ( $G_A^{At}$ ), and glitch width ( $G_W^{At}$ ). The trained model produces predicted values for glitch height ( $G_H^{Pred}$ ) and glitch area ( $G_A^{Pred}$ ), from which the glitch width ( $G_W^{Pred}$ ) is derived.

As shown in Fig. 3.9a, two specific glitches in the ISCAS’85 C17 circuit are evaluated:  $G1$ , occurring at the output of the gate  $g5$ , and  $G2$ , occurring at the output of the gate  $g6$ . The proposed method predicts  $G_H$  and  $G_A$  using gate-level features, specifically the transition times  $t_t$  of the gate inputs, the temporal distance  $t_D$ , and the capacitive load  $C_L$  at the gate output. The global input nodes  $N1 - N7$  are not directly used for prediction. Instead, the method relies on localized, gate-specific parameters extracted from the STA tool, which are influenced by the transition times and temporal differences of the global input nodes.

For  $G1$ , the inputs to the gate  $g5$  are  $N13$  and  $N236$ . The transition times  $t_{N13}$  and  $t_{N236}$ , the temporal difference  $t_D = AT_{N236} - AT_{N13}$ , and the capacitive load  $C_L$  at the output of  $g5$  are extracted from the STA tool. These features are provided to the ML model, which predicts the glitch height  $G_H$  and glitch area  $G_A$  at the output of  $g5$ .

Similarly, for the  $G2$ , the inputs to the gate  $g6$  are  $N236$  and  $N367$ . The transition times  $t_{N236}$  and  $t_{N367}$ , the temporal difference  $t_D = AT_{N367} - AT_{N236}$ , and the capacitive load  $C_L$  at the output of  $g6$  are extracted from the STA tool. These gate-specific parameters are used as inputs to the model to predict the corresponding glitch height and glitch area at the output of  $g6$ .

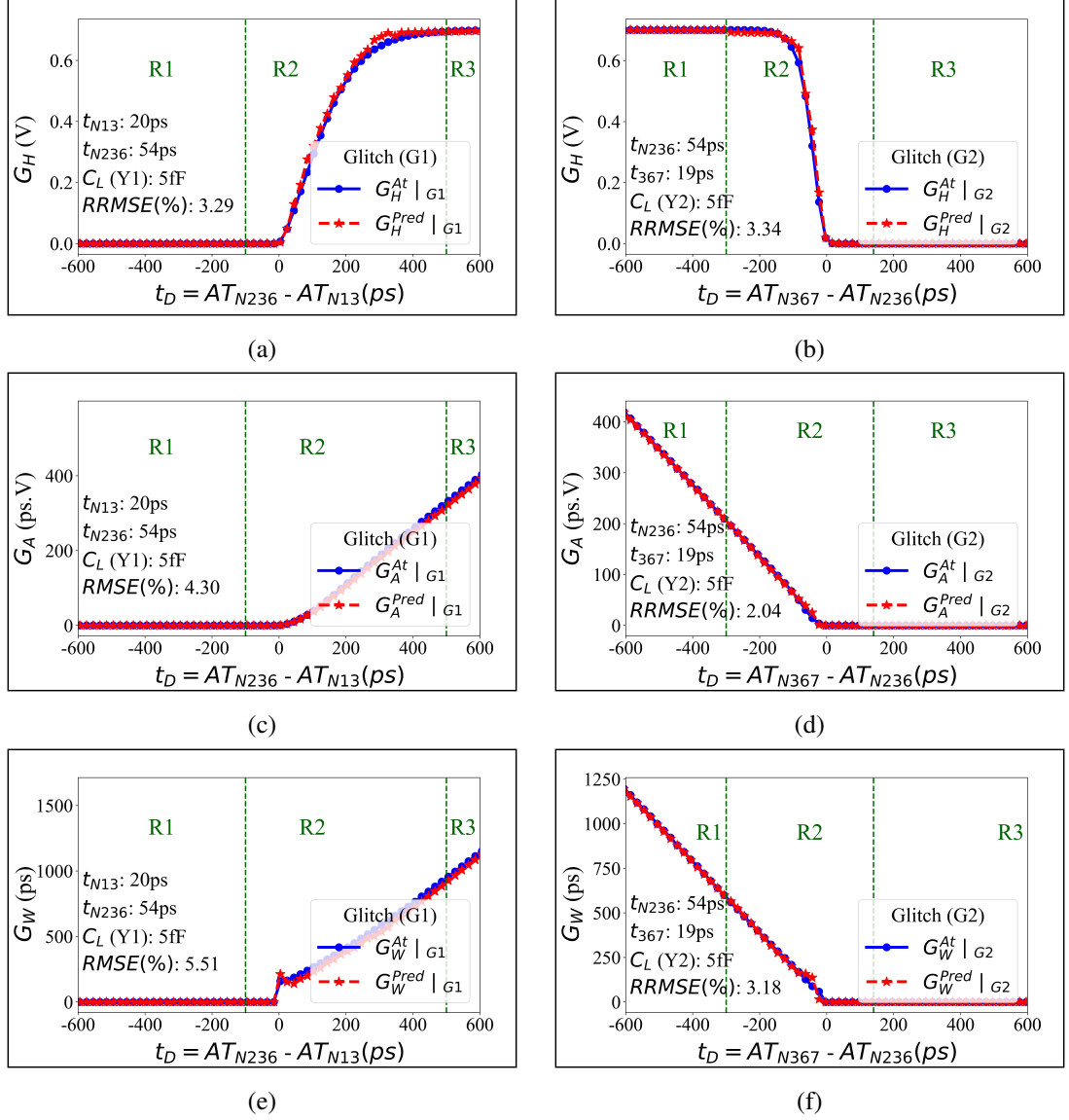


Figure 3.10: Evaluation of Glitches in C17 ISCAS Benchmark

To quantitatively assess the performance of the proposed methodology at the circuit level, the Relative Root Mean Square Error ( $RRMSE$ ) is employed as the evaluation metric. The mathematical formulation of  $RRMSE$ , expressed as a percentage, is given by

$$RRMSE (\%) = \left( \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (A_i - P_i)^2}}{\frac{1}{N} \sum_{i=1}^N A_i} \right) \times 100$$

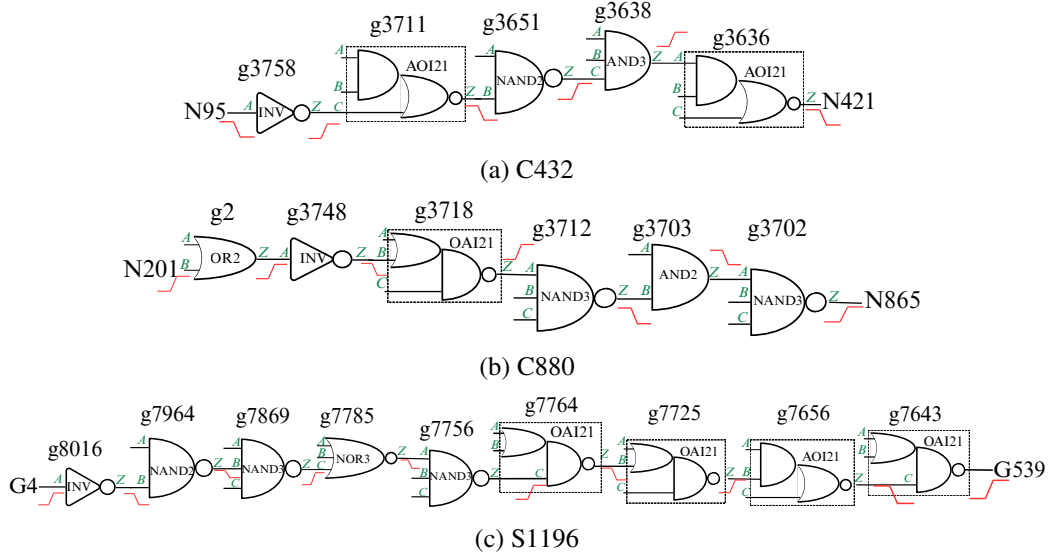


Figure 3.11: ISCAS Benchmark Paths

where  $N$  denotes the total number of samples,  $A_i$  represents the actual values, and  $P_i$  indicates the predicted values. By normalizing the  $RMSE$  using the mean of the actual values, a dimensionless metric is obtained. This enables an intuitive comparison of model performance, independent of the data scale.

For glitch  $G1$ , the results are presented in Figs. 3.10a, 3.10c, and 3.10e. As illustrated in Fig. 3.10a, close alignment is observed between the actual height ( $G_H^{At} |_{G1}$ ) and the predicted height ( $G_H^{Pred} |_{G1}$ ), resulting in an  $RRMSE$  of 3.29%. A similar trend is observed in Fig. 3.10c, where the  $RRMSE$  for the area comparison between the actual ( $G_A^{At} |_{G1}$ ) and predicted ( $G_A^{Pred} |_{G1}$ ) values is 4.30%. Furthermore, the width, calculated using the equation developed in Section 3.5.3, is shown in Fig. 3.10e. In this case, strong agreement is demonstrated between the actual width ( $G_W^{At} |_{G1}$ ) and the calculated width ( $G_W^{Pred} |_{G1}$ ), yielding an  $RRMSE$  of 5.51%.

The analysis of glitch  $G2$  follows a similar trend, as shown in Figs. 3.10b, 3.10d, and 3.10f. In Fig. 3.10b, a high degree of similarity is observed between the actual height ( $G_H^{At} |_{G2}$ ) and the predicted height ( $G_H^{Pred} |_{G2}$ ), resulting in an  $RRMSE$  of 3.34%. A similar trend is observed in Fig. 3.10d, where the  $RRMSE$  for the area between the actual ( $G_A^{At} |_{G2}$ ) and predicted ( $G_A^{Pred} |_{G2}$ ) values is 2.04%. In addition, Fig. 3.10f demonstrates that close correlation is maintained between the actual width ( $G_W^{At} |_{G2}$ ) and the calculated width ( $G_W^{Pred} |_{G2}$ ), with an  $RRMSE$  of 3.18%.

To evaluate the effectiveness of the proposed model on gates with more than two inputs, it is applied to three different designs (C432, C880, and S1196) and selected

Table 3.5: Evaluation of Glitches in ISCAS Benchmark Paths

(a) C432 ISCAS Benchmark Path

| STA Tool  |            |                              | SAT Solver  |                       |           | STA Tool   |            |                 |            | SPICE          |                   |                 | ML Model         |                     |                   |
|-----------|------------|------------------------------|-------------|-----------------------|-----------|------------|------------|-----------------|------------|----------------|-------------------|-----------------|------------------|---------------------|-------------------|
| Gate Name | Gate Type  | Arc ( $\rightarrow$ )        | Pattern Num | Input Pattern         | Pair Used | $t_x$ (ps) | $t_y$ (ps) | $t_{xy}^D$ (ps) | $C_L$ (fF) | $G_H^{At}$ (V) | $G_A^{At}$ (ps.V) | $G_W^{At}$ (ps) | $G_H^{Pred}$ (V) | $G_A^{Pred}$ (ps.V) | $G_W^{Pred}$ (ps) |
| g3711     | AOI-3 i/p  | $C \uparrow \rightarrow Z$   | 1           | 110 $\rightarrow$ 101 | X=B, Y=C  | 57         | 53         | 44              | 3          | 0.477          | 30                | 126             | 0.452            | 29                  | 128               |
|           |            |                              | 2           | 110 $\rightarrow$ 011 | X=A, Y=C  | 158        | 53         | -232            | 3          | 0.002          | 0                 | 0               | 0.008            | 0                   | 0                 |
| g3651     | NAND-2 i/p | $B \downarrow \rightarrow Z$ | 3           | 01 $\rightarrow$ 10   | X=A, Y=B  | 45         | 49         | 31              | 2          | 0.281          | 10                | 70              | 0.294            | 13                  | 88                |
| g3638     | AND-3 i/p  | $C \uparrow \rightarrow Z$   | 4           | 110 $\rightarrow$ 101 | X=B, Y=C  | 99         | 27         | -92             | 5          | 0.606          | 67                | 220             | 0.645            | 86                  | 267               |
|           |            |                              | 5           | 110 $\rightarrow$ 011 | X=A, Y=C  | 118        | 27         | -273            | 5          | 0.700          | 199               | 569             | 0.700            | 215                 | 614               |
| g3636     | AOI-3 i/p  | $A \uparrow \rightarrow Z$   | 6           | 011 $\rightarrow$ 110 | X=A, Y=C  | 121        | 75         | -167            | 15         | 0.555          | 130               | 468             | 0.532            | 132                 | 496               |
|           |            |                              | 7           | 010 $\rightarrow$ 100 | X=A, Y=B  | 121        | 57         | -224            | 15         | 0              | 0                 | 0               | 0.009            | 0                   | 0                 |

(b) C880 ISCAS Benchmark Path

| STA Tool  |            |                              | SAT Solver  |                       |           | STA Tool   |            |                 |            | SPICE          |                   |                 | ML Model         |                     |                   |
|-----------|------------|------------------------------|-------------|-----------------------|-----------|------------|------------|-----------------|------------|----------------|-------------------|-----------------|------------------|---------------------|-------------------|
| Gate Name | Gate Type  | Arc ( $\rightarrow$ )        | Pattern Num | Input Pattern         | Pair Used | $t_x$ (ps) | $t_y$ (ps) | $t_{xy}^D$ (ps) | $C_L$ (fF) | $G_H^{At}$ (V) | $G_A^{At}$ (ps.V) | $G_W^{At}$ (ps) | $G_H^{Pred}$ (V) | $G_A^{Pred}$ (ps.V) | $G_W^{Pred}$ (ps) |
| g2        | OR-2 i/p   | $B \uparrow \rightarrow Z$   | 1           | 10 $\rightarrow$ 01   | X=A, Y=B  | 91         | 30         | -93             | 5          | 0              | 0                 | 0               | 0.003            | 0                   | 0                 |
| g3718     | OAI-3 i/p  | $B \downarrow \rightarrow Z$ | 2           | 011 $\rightarrow$ 101 | X=A, Y=B  | 60         | 31         | -43             | 8          | 0.247          | 21                | 170             | 0.254            | 26                  | 205               |
|           |            |                              | 3           | 010 $\rightarrow$ 001 | X=B, Y=C  | 31         | 32         | -39             | 8          | 0.163          | 77                | 945             | 0.169            | 84                  | 994               |
| g3712     | NAND-3 i/p | $A \uparrow \rightarrow Z$   | 4           | 011 $\rightarrow$ 110 | X=A, Y=C  | 82         | 47         | -78             | 4          | 0.003          | 0                 | 0               | 0.001            | 0                   | 0                 |
|           |            |                              | 5           | 011 $\rightarrow$ 101 | X=A, Y=B  | 82         | 98         | 1               | 4          | 0.030          | 14                | 933             | 0.032            | 15                  | 938               |
| g3703     | AND-2 i/p  | $B \downarrow \rightarrow Z$ | 6           | 01 $\rightarrow$ 10   | X=A, Y=B  | 18         | 54         | 2               | 2          | 0.001          | 0                 | 0               | 0.006            | 0                   | 0                 |
| g3702     | NAND-3 i/p | $A \uparrow \rightarrow Z$   | 7           | 011 $\rightarrow$ 110 | X=A, Y=C  | 25         | 72         | 29              | 15         | 0.068          | 16                | 471             | 0.068            | 18                  | 529               |
|           |            |                              | 8           | 011 $\rightarrow$ 101 | X=A, Y=B  | 25         | 68         | -9              | 15         | 0.002          | 0                 | 0               | 0.001            | 0                   | 0                 |

(c) S1196 ISCAS Benchmark Path

| STA Tool  |            |                              | SAT Solver  |                       |           | STA Tool   |            |                 |            | SPICE          |                   |                 | ML Model         |                     |                   |
|-----------|------------|------------------------------|-------------|-----------------------|-----------|------------|------------|-----------------|------------|----------------|-------------------|-----------------|------------------|---------------------|-------------------|
| Gate Name | Gate Type  | Arc ( $\rightarrow$ )        | Pattern Num | Input Pattern         | Pair Used | $t_x$ (ps) | $t_y$ (ps) | $t_{xy}^D$ (ps) | $C_L$ (fF) | $G_H^{At}$ (V) | $G_A^{At}$ (ps.V) | $G_W^{At}$ (ps) | $G_H^{Pred}$ (V) | $G_A^{Pred}$ (ps.V) | $G_W^{Pred}$ (ps) |
| g7964     | AND-2 i/p  | $B \downarrow \rightarrow Z$ | 1           | 01 $\rightarrow$ 10   | X=A, Y=B  | 76         | 47         | 23              | 1          | 0.579          | 18                | 62              | 0.578            | 19                  | 65                |
| g7869     | NAND-3 i/p | $B \downarrow \rightarrow Z$ | 2           | 011 $\rightarrow$ 101 | X=A, Y=B  | 24         | 15         | 25              | 4          | 0.225          | 75                | 66              | 0.238            | 85                  | 71                |
|           |            |                              | 3           | 110 $\rightarrow$ 101 | X=B, Y=C  | 15         | 16         | -10             | 4          | 0.068          | 2                 | 58              | 0.070            | 0                   | 0                 |
| g7785     | NOR-3 i/p  | $C \uparrow \rightarrow Z$   | 4           | 010 $\rightarrow$ 001 | X=B, Y=C  | 77         | 12         | 61              | 2          | 0.700          | 46                | 131             | 0.695            | 48                  | 138               |
|           |            |                              | 5           | 100 $\rightarrow$ 001 | X=A, Y=C  | 59         | 12         | 88              | 2          | 0.700          | 66                | 189             | 0.702            | 87                  | 248               |
| g7756     | NAND-3 i/p | $B \downarrow \rightarrow Z$ | 6           | 011 $\rightarrow$ 101 | X=A, Y=B  | 19         | 65         | 6               | 1          | 0.088          | 2                 | 45              | 0.100            | 0                   | 0                 |
|           |            |                              | 7           | 110 $\rightarrow$ 101 | X=B, Y=C  | 65         | 39         | -27             | 1          | 0.053          | 2                 | 75              | 0.050            | 0                   | 0                 |
| g7746     | OAI-3 i/p  | $B \uparrow \rightarrow Z$   | 8           | 101 $\rightarrow$ 011 | X=A, Y=B  | 38         | 162        | 21              | 19         | 0.025          | 4                 | 75              | 0.017            | 7                   | 0                 |
|           |            |                              | 9           | 001 $\rightarrow$ 010 | X=B, Y=C  | 162        | 15         | -12             | 19         | 0.005          | 0                 | 0               | 0.008            | 2                   | 0                 |
| g7725     | OAI-3 i/p  | $B \downarrow \rightarrow Z$ | 10          | 011 $\rightarrow$ 101 | X=A, Y=B  | 11         | 303        | 159             | 8          | 0.004          | 0                 | 0               | 0.006            | 0                   | 0                 |
|           |            |                              | 11          | 010 $\rightarrow$ 001 | X=B, Y=C  | 303        | 9          | -230            | 8          | 0.657          | 172               | 524             | 0.660            | 165                 | 500               |
| g7656     | AOI-3 i/p  | $B \uparrow \rightarrow Z$   | 12          | 100 $\rightarrow$ 010 | X=A, Y=B  | 9          | 142        | 267             | 13         | 0.002          | 0                 | 694             | 0.000            | 0                   | 709               |
|           |            |                              | 13          | 101 $\rightarrow$ 110 | X=B, Y=C  | 142        | 15         | -297            | 13         | 0.700          | 243               | 0               | 0.697            | 247                 | 0                 |
| g7643     | OAI-3 i/p  | $C \downarrow \rightarrow Z$ | 14          | 001 $\rightarrow$ 100 | X=A, Y=C  | 43         | 230        | 462             | 19         | 0.664          | 317               | 955             | 0.685            | 324                 | 946               |
|           |            |                              | 15          | 001 $\rightarrow$ 010 | X=B, Y=C  | 15         | 230        | 309             | 19         | 0.693          | 231               | 667             | 0.657            | 213                 | 648               |

paths from the benchmark circuit [28], as shown in Fig. 3.11, which are mapped to the ASAP 7nm PDK [27]. Gate-level information, including gate name, gate type, and timing arc, is extracted using the STA tool for each path. Inverters are excluded from the analysis, as they are single-input gates and therefore cannot exhibit MIS-induced glitches.



The evaluation and analysis method is described using the C432 benchmark circuit. The path from input port N95 to output port N421 (Fig. 3.11a) is analyzed, and the corresponding gate-level details are summarized in Table 3.5a. For each multi-input gate along the path, its Boolean function is extracted and processed using the proposed SAT-based formulation to identify MIS transitions capable of producing output glitches. The STA tool provides the transition times of the switching inputs and the output load capacitance, from which the temporal separation  $t_D^{XY}$  is computed.

The values shown in Table 3.5a are extracted or computed as follows. Consider the gate instance g3711, which is a 3-input AOI gate. For the arc from input  $C \uparrow \rightarrow Z$ , two glitch-prone MIS transitions are identified by the solver:  $110 \rightarrow 101$  and  $110 \rightarrow 011$ . For each transition, the STA tool provides glitch-relevant parameters, including arrival times  $AT_X$  and  $AT_Y$ , transition times  $t_X$  and  $t_Y$ , and load capacitance  $C_L$ . The temporal distance is computed as  $t_D^{XY} = AT_Y - AT_X$ . These values are subsequently used (i) in SPICE simulations to obtain the actual glitch characteristics and (ii) as input features for computing glitch characteristics using the pre-trained ML model retrieved from the augmented library described in Section 3.5.4. The transition encoded as the key ("BC\_110\_101") & ("AC\_110\_011") in the mis\_glitch\_info named group serves as a mapping between the relevant input condition and the corresponding ML model.

These parameters are used both for SPICE simulations to obtain ground-truth glitch characteristics and as input features for the pre-trained ML model to generate predicted values. SPICE simulations yield ground-truth glitch characteristics, including glitch height  $G_H^{At}$ , glitch area  $G_A^{At}$ , and glitch width  $G_W^{At}$ . In parallel, the ML model produces the predicted values  $G_H^{Pred}$ ,  $G_A^{Pred}$ , and  $G_W^{Pred}$ . This process is repeated for all gates along the selected path, and the final glitch characteristics are summarized in Table 3.5a. As shown in Table 3.5a, the predicted values closely match the SPICE results. Among the identified MIS transitions, the transition causing the worst-case behavior can be selected for worst-case analysis.

The same evaluation and analysis method is applied to the C880 and S1196 benchmark paths shown in Fig. 3.11b and 3.11c, respectively. The extracted gate information, identified MIS transitions, STA-derived parameters, and the corresponding actual and predicted glitch metrics are reported in Tables 3.5b and 3.5c. Across the C432,

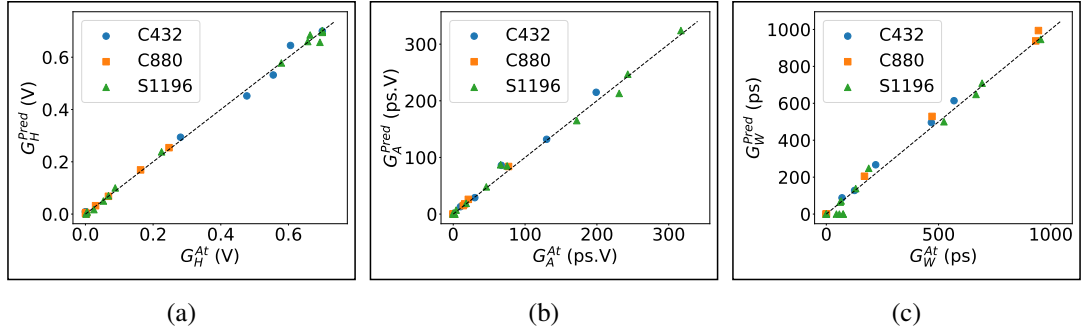


Figure 3.12: Comparison of glitch characteristics obtained using SPICE and proposed model in S1196 ISCAS benchmark path.

C880, and S1196 paths, a similar trend is observed between the actual glitch characteristics obtained from SPICE simulations and the values predicted by the ML model. Figure 3.12 further illustrates the accuracy of the proposed ML model through a direct comparison between SPICE-extracted and predicted glitch characteristics for the C432, C880, and S1196 benchmark paths. Each subfigure plots the predicted value against the corresponding SPICE result, with the dashed line representing ideal prediction. For all three metrics, i.e., glitch height, glitch width, and glitch area, the data points across all benchmarks cluster tightly around the ideal line, indicating strong agreement between the prediction and the ground truth. The proposed ML model demonstrates strong predictive accuracy across all benchmarks, with NRMSE values ranging from 1.62% to 2.94% for glitch height, 2.59% to 4.77% for glitch area, and 3.13% to 4.86% for glitch width, indicating close agreement with SPICE-simulated values.

### 3.6.3 Runtime Evaluation

The runtime performance was evaluated on a Linux platform with a 1.8 GHz processor and 4 GB RAM, without enabling multiprocessing or multithreading. The incorporation of these parallel processing techniques is expected to further accelerate both model training and SPICE characterization. SPICE simulations were conducted to generate a dataset of 50,000 data points, requiring approximately 14 hours. Training of the Multi-Layer Perceptron Regressor (ANN) required about 12 min using a three-layer architecture with 250, 150, and 100 neurons, respectively.

Inference times for predicting 50,000 data points are summarized as follows:

- Approximately 2.55 seconds were taken by the ML model.

- Approximately 14 hours were taken by SPICE simulations.

Hence, the proposed model can be efficiently employed in the inner loop of a glitch attenuation algorithm to estimate the impact of proposed circuit modifications. While SPICE maintains high accuracy, the speed of the ML model is advantageous for static timing analysis. Although an initial training effort is required, the proposed model can be reused across multiple designs, thereby reducing overall library creation costs.

### 3.6.4 Glitch Attenuation

Over time, several strategies have been proposed to address glitches in logic circuits, thereby enhancing circuit performance and reliability. Gate resizing has been employed for glitch power minimization [44]. In this approach, path delays in a combinational logic network are equalized by adjusting gate sizes and slowing down fast paths without affecting the worst-case delay. However, achieving perfect equalization is often challenging, particularly when path delays vary widely. Gate sizing within a statistical delay model has also been explored, where nonlinear programming is used to optimize delay [45]. Another technique, referred to as the variable input delay method, introduces gates with varying delays by modifying basic cell designs [46]. In addition, recent work [47] has emphasized simulation-based classification of glitches into generated and propagated types, while incorporating net capacitance and gate drive strength for improved delay and glitch activity estimation. The proposed metrics for identifying glitch-critical nets further support effective netlist-level mitigation strategies. This work demonstrates that predicting  $G_H$  and  $G_A$  using the proposed model enables effective glitch attenuation.

### Load Enhancement

It has been shown earlier that the capacitive load at the output of a logic gate reduces the glitch height. Hence, this approach can be employed to mitigate glitches in a circuit, provided that other effects of increased load, such as enhanced delay, are not critical. The impact of load enhancement on glitch attributes is illustrated for glitch G1 in Fig. 3.13a and for glitch G2 in Fig. 3.13b. A consistent reduction in  $G_H$  is observed for both gates  $G_1$  and  $G_2$  across various  $G_L$  values ranging from 5fF to 15fF, underscoring

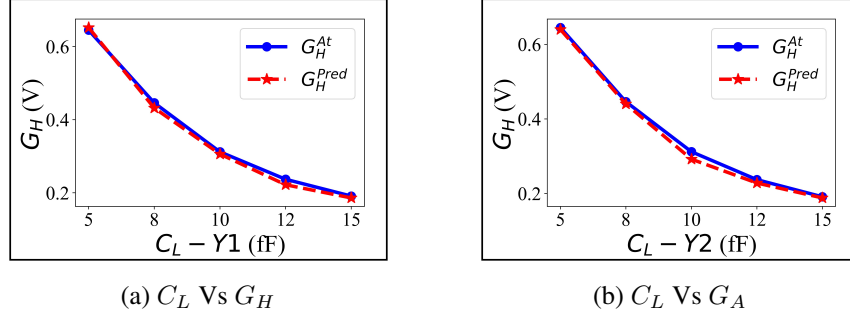


Figure 3.13: Load Enhancement ( $t_{N13}$ : 20ps,  $t_{N236}$ : 54ps,  $t_{N367}$ : 18ps,  $AT_{N13}$ - $AT_{N236}$ : 100ps,  $AT_{N236}$ - $AT_{N367}$ : -100ps)

the effectiveness of load enhancement in attenuating glitches. Moreover, the  $G_H$  predicted by the proposed model agrees reasonably well with SPICE simulations. Hence, instead of relying on SPICE simulations, the proposed model can be used to explore multiple design possibilities during glitch attenuation.

### Arrival Time Adjustment

Glitch attenuation strategies also involve delay balancing, which is achieved by strategically placing buffers along specific paths and modulating the arrival times [48]. Using this strategy to address glitch G1, a buffer (B1) was introduced at N13, as shown in Fig. 3.14. This decision was based on the observation that N13 arrived earlier than

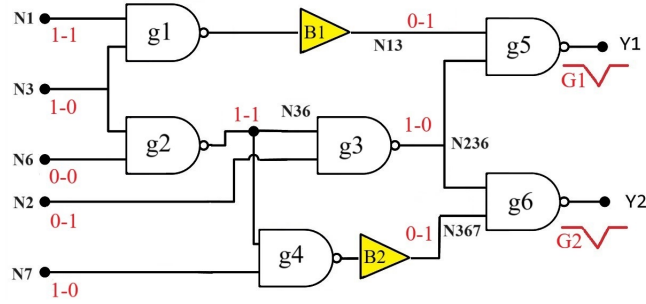


Figure 3.14: Arrival Time Adjustment

N236, and that mitigation of the glitch required N13 to arrive later than N236. The introduction of B1 increased the load for gate G1, thereby increasing its delay and, consequently, the arrival time at its output. By incrementally increasing the size of B1 at N13, the arrival time of N13 was increased, which reduced the magnitude of difference between  $AT_{N13}$  and  $AT_{N236}$ . This reduction in temporal difference resulted in a decrease in the height of  $G_H$ , as shown in Fig. 3.15a.

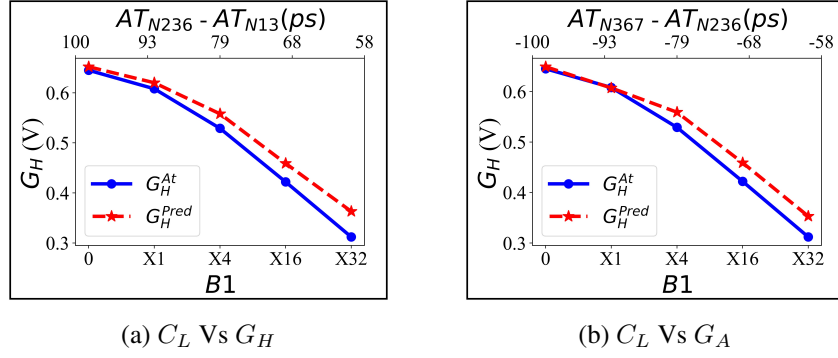


Figure 3.15: Arrival Time Adjustment ( $t_{N13}$ :20ps,  $t_{N236}$ :54ps,  $t_{N367}$ :18ps,  $C_L$  (Y1):5fF,  $C_L$  (Y2):5fF)

Similarly, for glitch G2, it was identified that the glitch occurred because N367 arrived earlier than N236. To mitigate this effect, a buffer (B2) was introduced at N367 to delay its arrival relative to N236, as shown in Fig. 3.14. By progressively increasing the size of the buffer at N367, the arrival time of N367 was increased, thereby diminishing the magnitude of difference between  $AT_{N236}$  and  $AT_{N367}$ , which ultimately resulted in a reduction in the height of  $G_H$ , as illustrated in Fig. 3.15b. It can be observed from Fig. 3.15 that the  $G_H$  values predicted using the proposed model agree well with SPICE simulations. Hence, the proposed model can be used in place of computationally expensive SPICE simulations to predict the impact of arrival-time modulation during glitch attenuation and to guide appropriate gate resizing.

### 3.6.5 Delay and Noise Impact of MIS-induced glitches

Glitches play an important role in SI analysis. A glitch on an aggressor line can impact the voltage levels and delays on a victim line, potentially disrupting circuit functionality or timing. Glitches on the aggressor line can be generated due to MIS. In this section, it is illustrated that the characteristics of glitches produced due to MIS can be predicted using the proposed model, which can further be used to examine SI issues. The circuit shown in Fig. 3.16a is employed for this illustration. Gate G1 is assigned a higher drive strength than gate G2. The net out1 acts as an aggressor, while out2 serves as a victim. A coupling capacitance ( $C_C$ ) exists between out1 and out2, with out1 having a load capacitance ( $C_L$ ) and out3 connected to a ground capacitance ( $C_G$ ). The presence of  $C_C$  modifies the base delay at out1.

First, the base delay of G2 was determined by keeping the aggressor net constant,

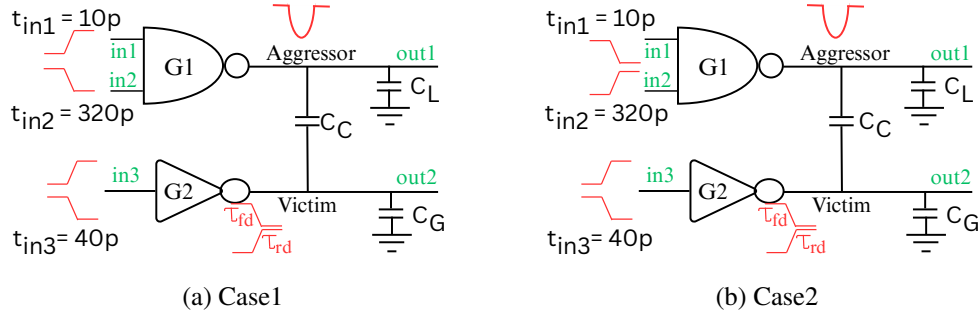


Figure 3.16: Example circuit ( $C_L:5fF$ ,  $C_C:5fF$ ,  $C_G:5fF$ )

and a rise delay ( $\tau_{rd}$ ) of 58 ps and a fall delay ( $\tau_{fd}$ ) of 90 ps were observed. Subsequently, reduced delay was observed by switching the victim and aggressor in the same direction: the negative incremental  $\tau_{rd}$  was 20 ps and the negative incremental  $\tau_{fd}$  was 26 ps. Next, enhanced delay was observed by switching the victim and aggressor in opposite directions: the positive incremental  $\tau_{rd}$  was 100 ps and the positive incremental  $\tau_{fd}$  was 166 ps. These delays were calculated with SIS considered at gate G1.

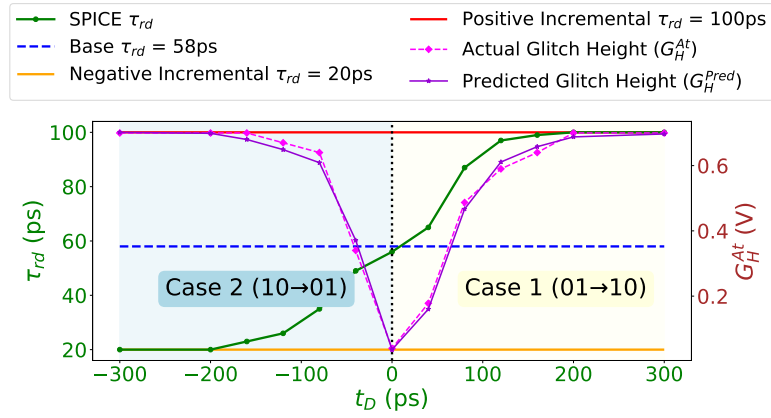
Next, MIS occurring at gate G1 instead of SIS is considered. First, the case where in1 transitions from  $0 \rightarrow 1$  and in2 from  $1 \rightarrow 0$  is analyzed, as shown in Fig. 3.16a. In this scenario, a glitch occurs at the aggressor rather than a smooth transition between states typically considered by SI tools. The attributes of the MIS-induced glitches and their impact on the victim delay are summarized in Table 3.6a. It is noted that the proposed model predicts glitch characteristics for varying temporal distance  $t_D$  in close agreement with SPICE results. Furthermore, SPICE simulations indicate that the glitch induced by MIS impacts the victim delay: as  $t_D$  increases,  $G_H$  increases, causing  $\tau_{rd}$  to increase until its worst positive incremental value is reached. At this point,  $\tau_{rd}$  stabilizes, as the aggressor glitch has reached its maximum possible transition. In contrast,  $\tau_{fd}$  exhibits the opposite trend. The  $\tau_{fd}$  decreases due to the aggressor glitch assisting the victim transition and reaches its worst negative incremental value before stabilizing, which occurs once the aggressor has undergone its maximum possible transition.

Next, the case where in1 transitions from  $1 \rightarrow 0$  and in2 from  $0 \rightarrow 1$  is considered, as shown in Fig. 3.16b, and the corresponding observations are summarized in Table 3.6b. The attributes of the MIS-induced glitches and their impact on the victim delay exhibit trends similar to those reported in Table 3.6a.

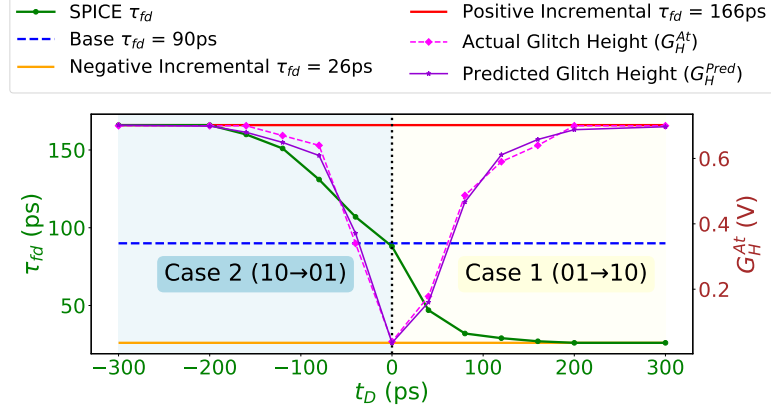
Figure 3.17a and 3.17b illustrate the impact of MIS-induced glitches on the victim delay for both rise and fall transitions. The primary Y-axis represents the delay ( $\tau_{rd}$

Table 3.6: Delay Impact of MIS Induced Glitches

| (a) Case 1: $in1(0 \rightarrow 1), in2(1 \rightarrow 0)$ |                   |                     |                     |                     | (a) Case 2: $in1(1 \rightarrow 0), in2(0 \rightarrow 1)$ |                   |                     |                     |                     |
|----------------------------------------------------------|-------------------|---------------------|---------------------|---------------------|----------------------------------------------------------|-------------------|---------------------|---------------------|---------------------|
| $t_D = \mathbf{AT}_{in2} - \mathbf{AT}_{in1}$<br>(ps)    | $G_H^{At}$<br>(V) | $G_H^{Pred}$<br>(V) | $\tau_{rd}$<br>(ps) | $\tau_{fd}$<br>(ps) | $t_D = \mathbf{AT}_{in2} - \mathbf{AT}_{in1}$<br>(ps)    | $G_H^{At}$<br>(V) | $G_H^{Pred}$<br>(V) | $\tau_{rd}$<br>(ps) | $\tau_{fd}$<br>(ps) |
| 0                                                        | 0.040             | 0.036               | 56                  | 88                  | 0                                                        | 0.100             | 0.143               | 56                  | 95                  |
| 40                                                       | 0.160             | 0.178               | 65                  | 47                  | -40                                                      | 0.340             | 0.371               | 49                  | 107                 |
| 80                                                       | 0.487             | 0.467               | 87                  | 32                  | -80                                                      | 0.640             | 0.609               | 35                  | 131                 |
| 120                                                      | 0.590             | 0.611               | 97                  | 29                  | -120                                                     | 0.670             | 0.649               | 26                  | 151                 |
| 160                                                      | 0.640             | 0.658               | 99                  | 27                  | -160                                                     | 0.700             | 0.680               | 23                  | 160                 |
| 200                                                      | 0.700             | 0.688               | 100                 | 26                  | -200                                                     | 0.700             | 0.699               | 20                  | 166                 |
| 300                                                      | 0.700             | 0.697               | 100                 | 26                  | -300                                                     | 0.700             | 0.702               | 20                  | 166                 |



(a) Rise Delay: Case 1 (01→10)



(b) Fall Delay: Case 2 (10→01)

Figure 3.17: Delay impact due to MIS-induced glitches for different input switching cases.

or  $\tau_{fd}$ ) as a function of  $t_D$ , while the secondary Y-axis represents  $G_H$ . Shaded regions distinguish the two transition scenarios: Case 1 ( $01 \rightarrow 10$ ) and Case 2 ( $10 \rightarrow 01$ ). Reference lines indicating the base delay and the negative and positive incremental delay bounds are also included. These figures highlight the correlation between  $G_H$ , the type of input transition, and the resulting delay variations. For Case 1 ( $01 \rightarrow 10$ ), as  $t_D$  increases,  $G_H$  increases, leading to an increase in  $\tau_{rd}$  and a corresponding decrease in  $\tau_{fd}$ . Conversely, for Case 2 ( $10 \rightarrow 01$ ), as  $t_D$  decreases,  $G_H$  increases, resulting in

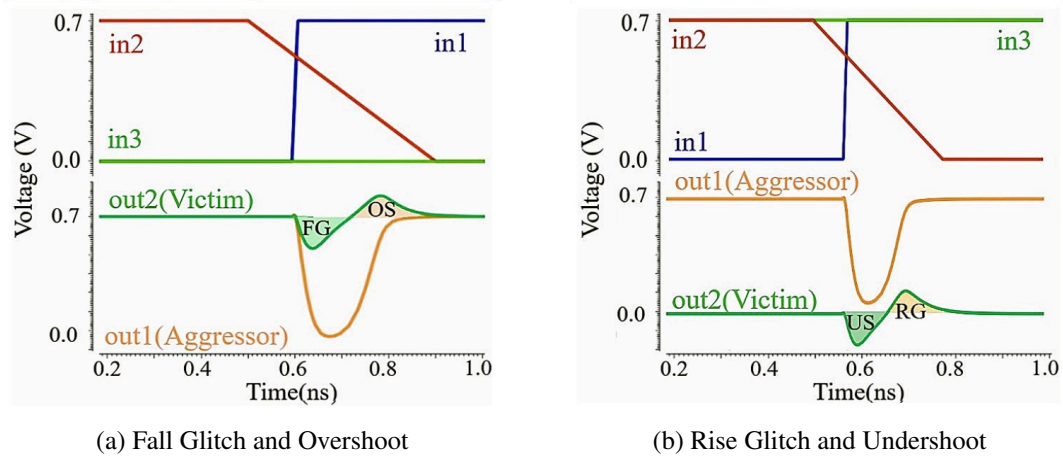


Figure 3.18: Noise Impact of MIS induced glitches ( $G_H^{At} = 0.635V$  & Aggressor  $G_H^{Pred} = 0.648V$ )

a decrease in  $\tau_{rd}$  and an increase in  $\tau_{fd}$ . These delay variations remain bounded within the range defined by the negative and positive incremental delays.

The proposed glitch model predicts  $G_H$  with high accuracy, as evidenced by close agreement with SPICE simulation results. Current SI tools typically consider only worst-case delays (positive and negative) and do not account for the range of delay variation introduced by MIS-induced glitches at the aggressor net. In future work, the proposed glitch model can address this limitation by enabling accurate prediction of glitch heights, thereby allowing emerging tools to estimate a broader and more realistic spectrum of delay variations and improving the robustness of SI analysis.

Glitches at the aggressor net can also induce a fall glitch (FG)/overshoot (OS) at a victim net held at logic 1 or a rise glitch (RG)/undershoot (US) at a victim net held at logic 0, as illustrated in Fig. 3.18, respectively.

The above observations demonstrate that the proposed model can predict the attributes of MIS-induced glitches at the aggressor, and that this information can subsequently be leveraged in SI and noise analysis, thereby further mitigating their harmful effects.

It should be noted that the goal of this work is not to propose a new formulation for glitch attenuation, but rather to enable fast, accurate glitch prediction to enhance existing mitigation strategies. The model provides a compact, ML-driven alternative to costly simulations and can help identify critical scenarios in which MIS-induced glitches may degrade timing or SI. In the future work, this predictive capability can be integrated into a complete SI analysis and optimization flow.



### 3.7 Conclusion

In this chapter, an efficient ML-driven model for predicting glitches caused by MIS in combinational logic gates has been proposed. The methodology was evaluated on ISCAS benchmark circuits, and its effectiveness in predicting and mitigating glitch effects was demonstrated. The results showed that the neural network model consistently achieved the lowest  $NRMSE$  across NAND, AND, NOR, and OR gates. A hyperparameter-tuned architecture with 250–150–100 neurons yielded the best performance, achieving an  $NRMSE$  of 0.0138. The dataset optimization strategy was shown to significantly reduce runtime without sacrificing accuracy.

At the circuit level, the model was validated on the ISCAS'85 C17 benchmark circuit, where the predicted glitch characteristics, including height, area, and width, closely matched SPICE simulation results, with an average  $RRMSE$  of 3.44% and a maximum  $RRMSE$  of 5.51%. On a Linux platform equipped with a 1.8 GHz processor and 4 GB RAM, dataset generation for 12 logic gates required approximately 28 hours, while training the ANN models required around 17 minutes. During inference, the models achieved a speedup of approximately  $\sim 10,000\times$  over SPICE for one million glitch predictions, demonstrating their suitability for rapid design iterations.

Glitch attenuation techniques, such as load enhancement and arrival time adjustment, were also explored. Both methods were shown to effectively mitigate glitches, and the model predictions closely aligned with SPICE results, confirming the suitability of the proposed approach for glitch attenuation. Finally, the results demonstrate that the ML model can accurately predict glitch characteristics and assist in assessing the resulting SI and noise impact. Integrating the proposed model into SI and noise analysis tools can therefore enable more robust and realistic timing analysis.

Future research can explore more advanced ML techniques to further enhance prediction accuracy and interpretability. In particular, models such as Physics-Informed Neural Networks and Graph Neural Networks are promising due to their ability to naturally incorporate physical laws, spatial relationships, and circuit topology. Additionally, future work can focus on developing more sophisticated dataset optimization strategies, extending the model to support more than two simultaneous switching events in complex multi-input combinational logic gates, and integrating the proposed glitch prediction framework with timing analysis, SI, and noise verification tools. Such extensions

would enable more comprehensive and glitch-aware digital design flows.

## CHAPTER 4

### **Timing Model for Latching Probability in a Flip-flop and its Application**

This chapter proposes ML-driven flip-flop timing models. It develops a probabilistic framework to model correct data latching for a flip-flop including the impact of process-induced variations. The proposed models implicitly capture the interdependencies among setup time, hold time, clock-to-Q delay and variations. The proposed models are further applied to safely resolve marginal timing violations produced by traditional STA tools during signoff.

The organization of this chapter is as follows. Section 4.1 presents the background and related work. A top-level overview of the proposed framework is provided in Section 4.2. Section 4.3 discusses the key factors influencing flip-flop timing behavior. Section 4.4 presents the proposed ML-driven flip-flop timing model. Section 4.5 compares the proposed probabilistic approach with deterministic models. Section 4.6 describes the violation resolution mechanism. Section 4.7 presents the results obtained by applying the proposed framework to benchmark circuits. Finally, Section 4.8 concludes the chapter.

The work presented in this chapter has been published in [49].

#### **4.1 Background and Related Work**

Conventional timing models of a flip-flop consist of three key attributes: the setup time (ST), the hold time (HT), and the clock-to-Q (C2Q) delays [50, 51]. These attributes are typically modeled conservatively as functions of clock slew and data slew or output load, represented using three separate two-dimensional tables. This modeling approach simplifies characterization, model development, and timing analysis.

However, in reality, these attributes are interrelated, and ignoring their mutual dependence can introduce unnecessary pessimism into the timing analysis, making timing closure more difficult [52]. Such pessimism often manifests as marginal timing viola-

tions reported by STA tools employing traditional technology libraries, but would not violate constraints in a real circuit. These false violations complicate timing closure by triggering additional design iterations, leading to schedule delays and unnecessary costs in area, power, or performance. Therefore, alleviating such violations through more accurate and less pessimistic timing models, rather than through design modifications, can significantly ease the timing closure process.

To alleviate the above shortcoming, several works have proposed interdependent modeling techniques. Early efforts [53]–[57] developed methodologies for simultaneous setup–hold characterization and demonstrated that exploiting their interdependence reduces pessimism and improves STA accuracy. In [53] and [55], methodologies for interdependent setup–hold characterization were proposed, and it was demonstrated that exploiting their tradeoff prevents both optimism and unnecessary pessimism in timing analysis, thereby reducing slack violations when integrated into sign-off tools. Numerical formulations, such as Euler–Newton curve tracing and Newton–Raphson-based approaches, were employed to enable efficient exploration of setup–hold tradeoffs, achieving more than an order-of-magnitude speedup in latch characterization without sacrificing accuracy [54, 56].

In addition to deterministic methods, statistical frameworks were developed to capture the probability distribution of codependent setup–hold contours under process variability, thereby enabling statistical static timing analysis (SSTA) and providing probabilistic guarantees of timing correctness [57]. Subsequent works extended interdependent modeling beyond STA to improve circuit robustness and enable optimization. In [58], it was shown that exploiting setup–hold interdependence throughout the design flow reduces delay uncertainty, while [59] introduced a nonlinear flip-flop model that enabled iterative timing analysis and clock period reduction. Flexible timing models were later proposed to recover slack margins [60], optimize clock skew [61], and minimize clock period using convex formulations [62]. Analytical models in [64]–[65] further improved accuracy and efficiency, demonstrating that flexible interdependent models can both reduce pessimism and drive circuit-level optimization.

These techniques model the interdependence between timing attributes (ST, HT, and C2Q delays) using linear, polynomial, or piece-wise linear functions. However, because the flip-flop timing behavior (e.g., C2Q delay) is highly nonlinear when data transitions

occur close to the clock edge, these models cannot fully exploit the available flexibility in flip-flop timing models.

Recently, ML approaches have been explored to address the limitations of analytical and piecewise models in capturing highly nonlinear flip-flop behavior. In [19], an ANN-based timing model was proposed that encodes the dependence of the C2Q delay on clock slew, data slew, output load, and setup/hold skew. The model was shown to achieve less than 3% error compared to SPICE simulations across 65 nm and 15 nm technologies. Extending this direction, an ANN-based interdependent timing model tailored for near-threshold voltage (NTV) design was developed in [66], where nonlinearities are more pronounced. By integrating the model into an STA flow with a genetic-algorithm-based iterative optimization, a mean absolute relative error below 0.7% was achieved, and minimum clock periods were reduced by 1.7–6.3% on ISCAS’89 benchmark circuits.

## 4.2 Proposed Work and Framework

While prior work has significantly advanced interdependent and ML-driven flip-flop timing modeling, several critical limitations remain unaddressed. First, most existing approaches primarily capture setup/hold interdependence and C2Q delay, but fall short of incorporating process-induced variations, which significantly impact timing behavior at advanced process nodes. Second, conventional ML-driven models rely on deterministic timing attributes and do not directly model the probabilistic nature of correct data latching under real circuit conditions. As a result, they cannot effectively distinguish genuine timing violations from pessimism-induced artifacts during sign-off analysis. Third, although ANN-based timing models reduce pessimism, their applicability in timing closure remains limited; marginal violations identified at sign-off often persist because existing frameworks lack mechanisms to safely waive false violations without risking chip failure. These limitations underscore the need for a more comprehensive ML-driven flip-flop timing-violation waiver framework that not only captures the multidimensional interdependence among timing attributes but also accounts for process variability and the probabilistic nature of data latching. Furthermore, resolving marginal timing violations must be carried out with extreme caution, as incorrectly waiving a real violation can lead to chip failure. Hence, such a framework must ensure

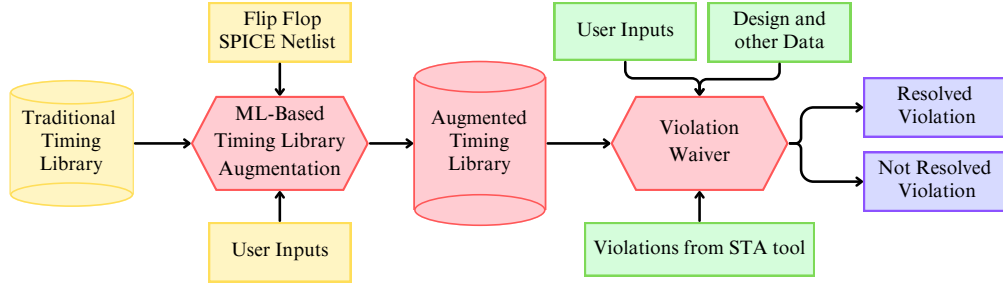


Figure 4.1: Proposed Methodology: Top-level view of the modeling and resolution framework.

safe violation filtering while integrating seamlessly into timing sign-off flows.

In this work, an ML-driven approach is proposed to develop an efficient timing model for flip-flops that addresses the aforementioned deficiencies of existing methods. Additionally, the application of the proposed models in safely resolving or waiving marginal timing violations is demonstrated. Given the circuit conditions and a flip-flop, unlike existing techniques, the setup time (ST) and hold time (HT) constraints are not explicitly modeled. Instead, the probability of correct data latching is modeled directly. The ability of ML to capture complex multi-dimensional dependencies is exploited in developing this model. Moreover, the impact of localized process-induced variations on timing attributes is inherently random; consequently, accounting for process-induced variation is naturally accommodated in the proposed probabilistic latching framework. Although earlier ML-driven flip-flop timing models offer high accuracy, waiving timing violations identified by these models is nontrivial. Furthermore, a flip-flop that appears to fail under a conventional timing model but latches data correctly under the proposed model may exhibit a higher C2Q delay than predicted by the conventional model. As a result, while a timing violation may be safely waived for a given flip-flop, new violations can potentially be introduced along its fanout paths.

To address these challenges, this work proposes a hierarchical violation filtering framework. The framework is designed to eliminate false positives by ensuring that real violations are never waived, while simultaneously accounting for the ramifications of any waiver on fanout paths. The hierarchical violation filtering framework employs a second ML-driven model to predict the increased C2Q delay of a marginally failing flip-flop that is determined to latch data correctly. It further verifies that sufficient slack exists in the flip-flop’s fanout to accommodate the increased C2Q delay before a violation is declared waived.

The top-level view of the proposed framework for flip-flop model development and

marginal violation resolution is shown in Fig. 4.1. Given transistor-level netlists for flip-flops, ML-driven models are developed using SPICE simulations and Monte Carlo analysis that account for process-induced variations. These models capture the probability of correct data latching and the corresponding C2Q delay of a flip-flop under various circuit conditions. The existing timing library is augmented with these ML-driven models to form an augmented timing library. The augmented timing library is generated once for a given standard cell library and can subsequently be reused across multiple designs for marginal timing violation resolution.

For a given design, an initial list of failing flip-flops is obtained using a commercial STA tool and the existing timing libraries. Necessary information is then extracted from the STA tool to analyze the failing paths, and the ML models from the augmented timing library are utilized to resolve or waive violations. Finally, a list of resolved and unresolved violations is produced for the design.

The key contributions of this work are as follows. A novel framework is proposed for capturing flip-flop timing characteristics based on the probability of correct data latching, and this model is utilized to assess the validity of violations identified using conventional timing models. An efficient ML-driven algorithm is introduced to derive such probabilistic timing models. Additionally, a violation resolution framework is proposed to waive relevant marginal timing violations with 100% precision. The effectiveness of the proposed methodology is demonstrated on benchmark circuits from TAU CONTEST'19 [67], implemented using 45nm standard cell libraries and SPICE models. Moreover, the proposed approach is sufficiently general to allow broader applicability. With continued advancements in ML techniques, the methodology can be extended to filter marginal violations arising from SI effects and IR-drop in supply lines [68, 69, 70].

### 4.3 Factors Impacting Timing in a Flip-flop

A flip-flop is expected to latch the data present at its  $D$  pin upon the arrival of the clock edge and to produce the output after a clock-to-Q (C2Q) delay ( $\tau_{C2Q}$ ), as shown in Fig. 4.2. Besides the data slew, clock slew, and output load, both the C2Q delay and the data latching behavior are dependent on the setup skew and the hold skew. The time

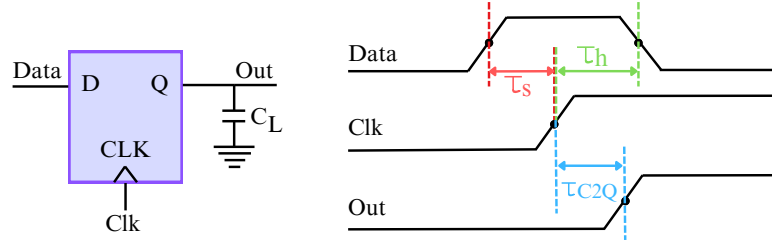


Figure 4.2: Definition of circuit parameters in a flip-flop.

interval between the signal arriving at the  $D$  pin and the subsequent arrival of the clock edge at the clock pin is defined as the setup skew ( $\tau_s$ ). Since signals may arrive at the  $D$  pin through multiple paths, the last arriving signal is considered for measuring the setup skew [19]. The hold skew ( $\tau_h$ ) is defined as the time interval between the clock signal arriving at the clock pin and the subsequent change in the signal value at the  $D$  pin. For measuring the hold skew, the earliest change in the signal value at the  $D$  pin is considered [19].

As the data signal transition is shifted closer to the clock edge, the  $\tau_{C2Q}$  delay can increase, and in extreme cases, data latching can fail. Moreover, process-induced variations can affect both the  $\tau_{C2Q}$  delay and the latching behavior of a flip-flop. This behavior is examined using SPICE simulations employing Monte Carlo (MC) method, which accounts for process-induced local variations and device mismatches. For various combinations of  $\tau_s$  and  $\tau_h$ , the probability of data latching ( $P_L$ ) and the corresponding  $\tau_{C2Q}$  are observed, and the results are presented in Fig. 4.3.

Figure 4.3a illustrates the impact of the setup skew  $\tau_s$  on the probability of data latching ( $P_L$ ) for varying values of hold skew  $\tau_h$ . When  $\tau_s$  is large,  $P_L$  remains at 100%, indicating reliable data latching. As  $\tau_s$  is reduced,  $P_L$  decreases sharply to 0%, indicating a failure to latch the data correctly. A threshold  $\theta_{\text{SPICE}} = 0.5$  can be defined, such that  $P_L > \theta_{\text{SPICE}}$  indicates that the flip-flop is more likely to latch the data correctly. The value of setup skew at which  $\theta_{\text{SPICE}}$  is reached is considered as the latching threshold for the setup skew and is denoted as  $L_{\text{setup}}$ . From Fig. 4.3b, it can be observed that  $L_{\text{setup}}$  depends on  $\tau_h$  and decreases as  $\tau_h$  increases.

It is noted that existing timing libraries model the setup time independently of  $\tau_h$  by assuming conservative or small hold skews during setup time characterization. However, this assumption introduces pessimism for flip-flops operating under circuit conditions that impose larger  $\tau_h$  values than those originally characterized. One of the



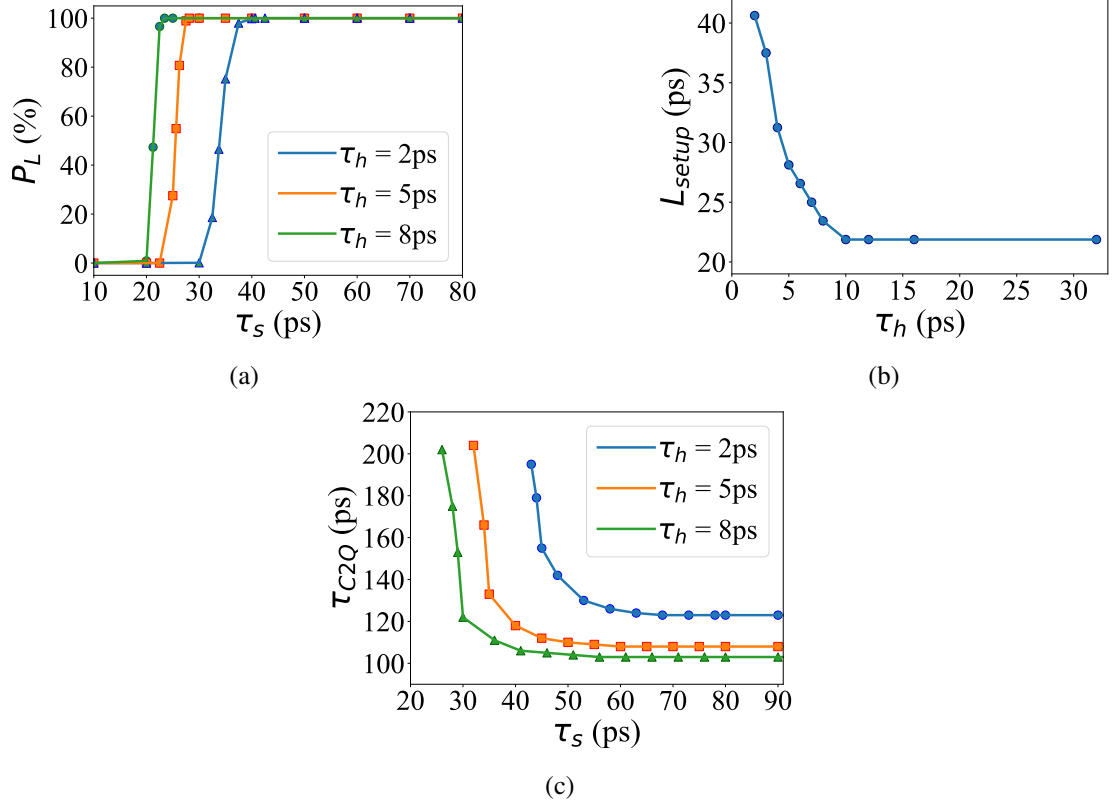


Figure 4.3: (a) Setup skew ( $\tau_s$ ) vs. probability of latching ( $P_L$ ), (b) latching threshold for the setup skew  $L_{setup}$  vs. hold skew ( $\tau_h$ ), (c) C2Q delay ( $\tau_{C2Q}$ ) vs. setup skew ( $\tau_s$ ).

objectives of this work is to resolve marginal timing violations by eliminating such pessimism.

Fig. 4.3c shows the dependence of the clock-to-Q delay ( $\tau_{C2Q}$ ) on  $\tau_s$  and  $\tau_h$ . It is observed that reductions in both  $\tau_s$  and  $\tau_h$  can lead to an increase in  $\tau_{C2Q}$ . Conventional flip-flop timing models typically neglect the dependence of C2Q delay on  $\tau_s$  and  $\tau_h$ . To compensate for this limitation, setup and hold time constraints are conservatively defined, thereby allowing these dependencies to be ignored. However, such constraints restrict the available flip-flop timing space within a design. While this restricted timing space is safe, it introduces unnecessary pessimism. This work aims to expand the permissible timing space by reducing this pessimism, thereby enabling better optimized design solutions.

## 4.4 ML-driven Flip-flop Timing Models

The methodology for developing the proposed flip-flop timing models is illustrated in Fig. 4.4. It involves creating two separate ML models: (a) *LatchPredict*: to compute

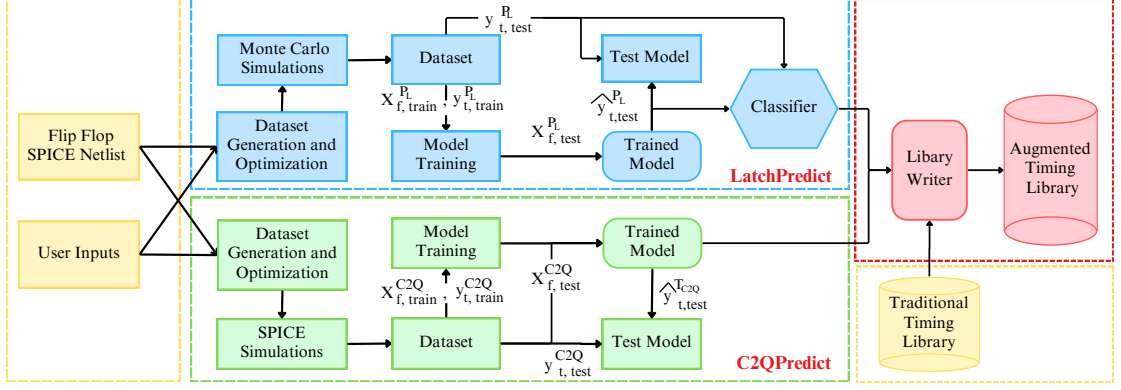


Figure 4.4: Model creation framework

the probability of latching data correctly in a flip-flop and an in-built classifier, and (b) *C2QPredict*: to compute C2Q delay for the expanded timing space for the flip-flops. These models are created from a dataset generated by circuit simulation for the given transistor-level SPICE netlist. The datasets are then preprocessed and split into training and test subsets, with the training data used for model development and the test data reserved for evaluation. Once the ML models are trained, the existing technology libraries are augmented with these models and are ready for deployment for various designs. The performance of these models are assessed using the Normalized Root Mean Squared Error (NRMSE), which is calculated as:

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2}}{\max(y) - \min(y)}$$

where  $y_t$  denotes the actual values,  $\hat{y}_t$  the predicted values,  $n$  the total number of data points, and  $\max(y)$ ,  $\min(y)$  represent the range of actual values. Details of these models and their implementation are described in the following section.

#### 4.4.1 LatchPredict

*LatchPredict* is a regression model that is used to predict the probability  $P_L$  of correctly latching data for a given flip-flop under a specified set of features. Through experimental analysis, the following circuit parameters were identified as having a significant impact on  $P_L$ :

$$P_L = F(t_d, t_c, \tau_s, \tau_h), \quad (4.1)$$

where  $t_d$  denotes the data slew,  $t_c$  denotes the clock slew,  $\tau_s$  represents the setup skew, and  $\tau_h$  represents the hold skew. These circuit parameters are treated as input features, while  $P_L$  is treated as the label for the model. Subsequently, Monte Carlo (MC)-based SPICE simulations are performed using these features to generate the dataset required for training the model.

As illustrated in Fig. 4.3a, the behaviour of a flip-flop can be classified into three distinct regions: (a) the correctly latching region ( $P_L = 100\%$ ), Fig. 4.3b an intermediate region in which the flip-flop may become metastable ( $0 < P_L < 100\%$ ), and (c) the failure region ( $P_L = 0\%$ ). From a modeling perspective, the intermediate region is critical because it exhibits a sharp transition in  $P_L$ . Due to the narrow extent of this region, there exists a risk of under-sampling, which can result in insufficient representation within the dataset. To mitigate this issue, Algorithm 1 is designed to capture the complete timing space, including the intermediate region. The algorithm dynamically adjusts the sampling interval to ensure that at least one sample point lies within the intermediate region. This strategy optimizes the dataset while minimizing the overall characterization effort.

Algorithm 1 systematically constructs the dataset by iterating over different combinations of timing parameters. The procedure begins by initializing  $\tau_s$  to its starting value, followed by the execution of Monte Carlo simulations for each combination of  $t_c$ ,  $t_d$ ,  $\tau_h$ , and  $\tau_s$  to compute  $P_L$ . Based on the computed values of  $P_L$ , the sampling strategy is adaptively refined. If  $P_L$  is below 10%, indicating that the simulation was likely performed in the failure region, the upper bound of  $\tau_s$  is updated and the simulations are repeated. If  $P_L$  lies between 40% and 60%, the sample is classified as belonging to the intermediate region, and the algorithm proceeds to the next parameter combination. If  $P_L$  exceeds 60%, suggesting that the simulation was conducted in the correctly latching region, the lower bound of  $\tau_s$  is adjusted and the simulations are repeated. This adaptive sampling approach ensures adequate coverage of the intermediate region while maintaining a minimal dataset size. The final dataset is stored with the computed  $P_L$  values corresponding to each unique combination of  $t_c$ ,  $t_d$ ,  $\tau_h$ , and  $\tau_s$ , thereby ensuring comprehensive coverage of all operating regions with minimal computational overhead.

The dataset generated using the above algorithm is preprocessed prior to its use for model training and evaluation. Let  $X_f^{P_L} \in R^{n \times m}$  denote the feature matrix, where

---

**Algorithm 1** Dataset generation algorithm for *LatchPredict*

---

**Require:** Input parameters:  $t_c, t_d, \tau_h, \tau_{s,\text{start}}, \tau_{s,\text{end}}$ , and  $N$ .

**Ensure:** The  $P_L$  is computed and stored for every unique combination of  $t_c, t_d, \tau_h$ , and  $\tau_s$ .

Initialize  $\tau_{s,\text{curr}} = \tau_{s,\text{start}}$ .

**for** each combination of  $t_c, t_d, \tau_h$  **do**

Set  $iterate = true$

**while**  $iterate == true$  **do**

Run  $N$  Monte Carlo simulations using  $t_c, t_d, \tau_h$ , and  $\tau_{s,\text{curr}}$  and compute corresponding  $P_L$

**if**  $P_L < 40\%$  **then**

Set  $\tau_{s,\text{end}} = \tau_{s,\text{curr}}$  and update  $\tau_{s,\text{curr}} = \frac{\tau_{s,\text{start}} + \tau_{s,\text{curr}}}{2}$

**else if**  $40\% \leq P_L \leq 60\%$  **then**

Set  $iterate = false$

**else if**  $P_L > 60\%$  **then**

Set  $\tau_{s,\text{start}} = \tau_{s,\text{curr}}$  and update  $\tau_{s,\text{curr}} = \frac{\tau_{s,\text{curr}} + \tau_{s,\text{end}}}{2}$

**end if**

**end while**

**end for**

Store  $P_L$  for each combination of  $t_c, t_d, \tau_h, \tau_{s,\text{curr}}$ .

---

$n = 1230$  represents the number of samples and  $m = 4$  represents the number of features, and let  $y_t^{P_L} \in R^n$  denote the corresponding target labels. The feature matrix  $X_f^{P_L}$  is subjected to min-max normalization to scale all features within the range  $[0, 1]$ . Subsequently, the dataset is partitioned into a training set  $(X_{f,\text{train}}^{P_L}, y_{t,\text{train}}^{P_L})$  and a testing set  $(X_{f,\text{test}}^{P_L}, y_{t,\text{test}}^{P_L})$ . The regression model is trained using  $(X_{f,\text{train}}^{P_L}, y_{t,\text{train}}^{P_L})$ , and its performance is evaluated on  $(X_{f,\text{test}}^{P_L}, y_{t,\text{test}}^{P_L})$ .

Following the generation of the training and testing datasets, multiple ML models with varying levels of complexity and data representation requirements were developed and evaluated for model training. The training process was conducted while ensuring that overfitting was avoided. A comparative performance analysis of eight ML models, based on their NRMSE values, is presented in Table 4.1. To achieve optimal performance, exhaustive hyperparameter tuning was performed for all models. After evaluating multiple configurations, the best-performing models were identified.

As shown in Table 4.1, the prediction errors obtained from individual models were relatively high, indicating the need for a more robust modeling approach. To address this limitation, an ensemble learning strategy was adopted that integrates predictions from Gradient Boosting Regression (GBR), CatBoost, and an Artificial Neural Network (ANN) using a weighted averaging technique. The ensemble model was constructed

using an optimized weight distribution of 0.2 for GBR, 0.2 for CatBoost, and 0.6 for ANN. This combination effectively leverages the strengths of each constituent model while mitigating their individual weaknesses, resulting in improved generalization. The results demonstrate that the ensemble model outperforms all single-model approaches, achieving the lowest Test NRMSE. This improvement highlights the effectiveness of model aggregation in enhancing predictive accuracy.

Table 4.1: Performance Comparison of Single Models

| Model                           | Train NRMSE   | Test NRMSE    |
|---------------------------------|---------------|---------------|
| Decision Tree                   | 0.1426        | 0.1657        |
| Random Forest                   | 0.1327        | 0.1598        |
| XGBoost                         | 0.1201        | 0.1534        |
| LightGBM                        | 0.1078        | 0.1488        |
| Gradient Boosting (GB)          | 0.1201        | 0.1400        |
| Cat Boosting (CB)               | 0.1332        | 0.1216        |
| Artificial Neural Network (ANN) | 0.1019        | 0.1238        |
| <b>Ensemble (GB + CB + ANN)</b> | <b>0.0860</b> | <b>0.0945</b> |

The hyperparameters utilized for each model are detailed in Table 4.2.

Table 4.2: Hyperparameter Tuning

| GBR Hyperparams   |       | CB Hyperparams |       | ANN Hyperparams    |               |
|-------------------|-------|----------------|-------|--------------------|---------------|
| Parameter         | Value | Parameter      | Value | Parameter          | Value         |
| n-estimators      | 700   | n-estimators   | 700   | hidden layers      | 500, 250, 100 |
| learning rate     | 0.02  | learning rate  | 0.05  | learning rate init | 0.001         |
| max depth         | 3     | depth          | 3     | alpha              | 0.01          |
| min samples split | 5     | subsample      | 0.9   | batch size         | 64            |
| min samples leaf  | 5     | random state   | 42    | max iter           | 5000          |

After training and testing the model, a classifier is implemented as shown in Fig. 4.5 to convert the continuous predictions of  $P_L$  from the regression model into binary labels. The actual probabilities of latching at a given circuit condition are obtained using Monte Carlo SPICE simulations, and a flip-flop is classified as *Pass* at that circuit condition if the probability exceeds  $\theta_{SPICE}$ , otherwise, it is classified as *Fail*. Similarly, predicted values  $\hat{P}_L$  obtained using the trained ML model are used to classify as *Pass* if it exceeds  $\theta_{Model}$ , otherwise, they are classified as *Fail*.

Initially, for illustration purposes, the actual *Pass/Fail* labels are obtained by selecting a threshold of  $\theta_{SPICE} = 50\%$ , i.e., a flip-flop is considered to pass if it has a higher probability of latching data than failing to do so. Similarly, the predicted *Pass/Fail* labels are obtained by selecting  $\theta_{Model} = 50\%$ . Based on these criteria, the

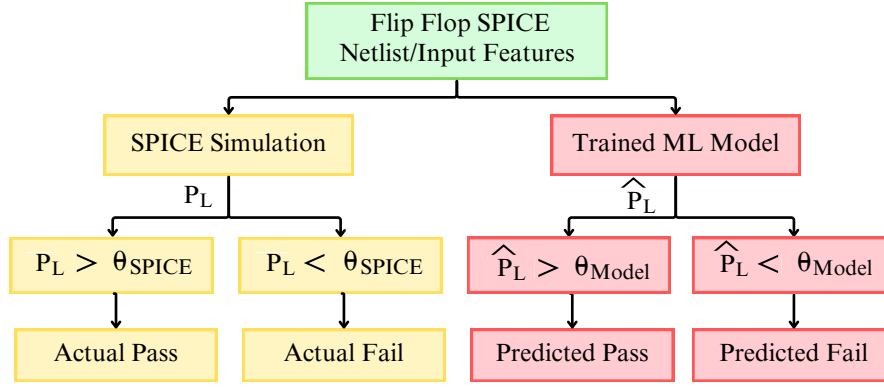


Figure 4.5: LatchPredict Classification Flow

Table 4.3: Confusion Matrix and Performance Metrics ( $\theta_{SPICE} = 50\%$  and  $\theta_{Model} = 50\%$ )

| Confusion Matrix    |                             |                |
|---------------------|-----------------------------|----------------|
|                     | Predicted PASS              | Predicted FAIL |
| Actual PASS         | TP = 136                    | FN = 2         |
| Actual FAIL         | FP = 6                      | TN = 103       |
| Performance Metrics |                             |                |
| Metric              | Formula                     | Value          |
| Accuracy            | $\frac{TP+TN}{TP+TN+FP+FN}$ | 0.96           |
| Precision           | $\frac{TP}{TP+FP}$          | 0.97           |
| Sensitivity         | $\frac{TP}{TP+FN}$          | 0.99           |
| Specificity         | $\frac{TN}{TN+FP}$          | 0.94           |

corresponding confusion matrix and classification performance metrics are presented in Table 4.3.

From the confusion matrix, 136 instances are correctly identified as *Pass* (True Positives, TP), and 103 instances are correctly identified as *Fail* (True Negatives, TN). However, 2 *Pass* instances are incorrectly classified as *Fail* (False Negatives, FN), and 6 *Fail* instances are misclassified as *Pass* (False Positives, FP). The resulting overall accuracy of 0.96 indicates that 96% of the predictions are correct. The precision, which measures the proportion of correctly predicted *Pass* instances among all predicted *Pass* cases, is 0.97, indicating that 97% of the instances predicted as *Pass* are indeed correct. The sensitivity (recall) is 0.99, demonstrating that nearly all actual *Pass* cases are successfully identified by the model. The specificity, which evaluates the ability to correctly classify *Fail* cases, is 0.94, indicating that 94% of the *Fail* instances are correctly identified.

In the proposed methodology, precision is identified as the most critical metric, since the primary objective is the elimination of marginal violations. Predicting a *Fail* out-

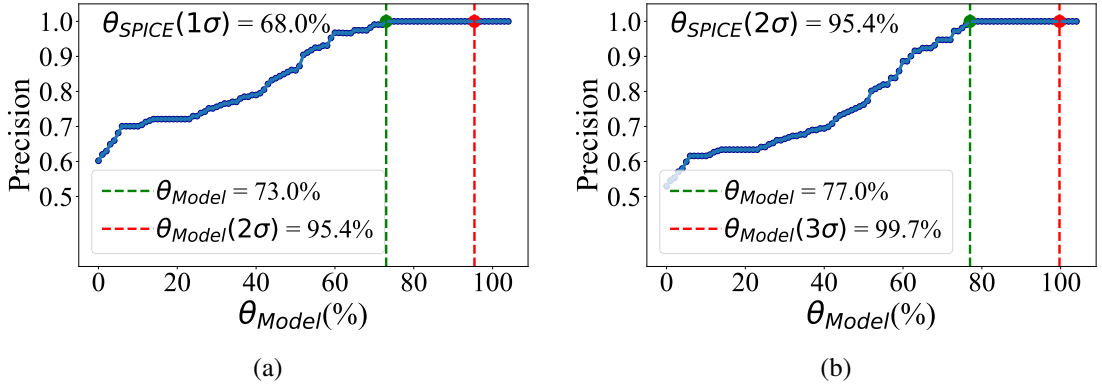


Figure 4.6: Precision vs. Threshold Plot

come when the actual outcome is *Pass* is considered acceptable. However, predicting a *Pass* outcome when the actual outcome is *Fail* must be strictly avoided. To satisfy this requirement, precision is targeted to be improved from 97% to 100% by appropriately adjusting the threshold  $\theta_{Model}$ , while ensuring that the resulting decision boundary accounts for process-induced variations.

Depending on the application and the desired level of confidence, latching probabilities corresponding to  $1\sigma$  (68%),  $2\sigma$  (95.4%),  $3\sigma$  (99.7%), and higher confidence levels can be considered. Figure 4.6a illustrates the variation in precision for  $\theta_{SPICE} = 1\sigma$  as  $\theta_{Model}$  is varied between 0% and 100%. It is observed that 100% precision is achieved when  $\theta_{Model} \geq 73\%$ . Similarly, Fig. 4.6b shows the variation in precision for  $\theta_{SPICE} = 2\sigma$ , where 100% precision is attained for  $\theta_{Model} \geq 77\%$ . It is observed that, with increasing confidence levels, achieving 100% precision requires a higher value of  $\theta_{Model}$ .

To achieve 100% precision for a given latching probability of  $X\%$ ,  $\theta_{Model}$  must be chosen such that  $\theta_{Model} > X\%$ , thereby accounting for statistical errors inherent in ML-driven predictions. A heuristic approach can be adopted to determine the minimum required value of  $\theta_{Model}$ . Specifically, to achieve a latching probability corresponding to  $N\sigma$ ,  $\theta_{Model}$  can be set to  $(N+1)\sigma$ . For example, to achieve a latching probability of 68% ( $1\sigma$ ),  $\theta_{Model}$  can be set to 95.4% ( $2\sigma$ ), and to achieve a latching probability of 95.4% ( $2\sigma$ ),  $\theta_{Model}$  can be set to 99.7% ( $3\sigma$ ). Experimental results confirm that this approach consistently achieves 100% precision. Lower values of  $\theta_{Model}$  (e.g., 50% or  $1\sigma$ ) are suitable for early-stage analysis, whereas higher values (e.g.,  $3\sigma$ ) are more appropriate for final sign-off to ensure robustness. This tunability enables the proposed framework to adapt to different stages of the design flow. The impact of  $\theta_{Model}$  on violation waivers

is analyzed in detail in the following sections.

#### 4.4.2 C2QPredict

*C2QPredict* is defined as a regression model for predicting the clock-to-Q (C2Q) delay ( $\tau_{C2Q}$ ) of a flip-flop based on a set of input features. Through experimental analysis, the following circuit parameters are identified as having a significant impact on  $\tau_{C2Q}$ :

$$\tau_{C2Q} = G(t_d, t_c, \tau_s, \tau_h, C_L) \quad (4.2)$$

where  $t_d$  denotes the data slew,  $t_c$  denotes the clock slew,  $\tau_s$  denotes the setup skew,  $\tau_h$  denotes the hold skew, and  $C_L$  denotes the output load. In this formulation,  $\tau_{C2Q}$  is treated as the label, while  $t_d$ ,  $t_c$ ,  $\tau_s$ ,  $\tau_h$ , and  $C_L$  serve as the input features. Using these features, SPICE simulations are subsequently performed to generate the dataset required for training the model.

---

**Algorithm 2** Dataset generation algorithm for *C2QPredict*

---

**Require:** Input parameters:  $\tau_h$ ,  $\tau_{s,start}$ ,  $\tau_{s,end}$ ,  $t_c$ ,  $t_d$ , and  $C_L$

**Ensure:** The  $\tau_{C2Q}^{curr}$  is computed and stored for every unique combination of  $t_c$ ,  $t_d$ ,  $\tau_h$ ,  $\tau_s$ , and  $C_L$

Initialize:  $count = 0$ ,  $\tau_s = \tau_{s,start}$ ,  $\Delta = 10p$ ,  $\tau_{C2Q}^{prev} = 0$ , and convergence threshold  $\epsilon = 0.1p$

**for** each combination of  $t_c, t_d, \tau_h, C_L$  and  $\tau_s$  where  $\tau_s \leq \tau_{s,end}$  **do**

**while**  $count \leq 3$  **do**

        Perform simulation to calculate  $\tau_{C2Q}^{curr}$ , update  $\tau_s = \tau_s + \Delta$ , compute  $Diff = \tau_{C2Q}^{curr} - \tau_{C2Q}^{prev}$

**if**  $|Diff| \leq \epsilon$  **then**

            Increment counter:  $count = count + 1$  to indicate convergence progress

**else**

            Reset counter:  $count = 0$  to restart the convergence process

**end if**

        Update the previous value:  $\tau_{C2Q}^{prev} = \tau_{C2Q}^{curr}$  to prepare for the next iteration

**end while**

**end for**

**Output:** Store the  $\tau_{C2Q}^{curr}$  value for the current combination of  $t_c$ ,  $t_d$ ,  $\tau_h$ ,  $\tau_s$ , and  $C_L$

---

Algo. 2 is used to systematically construct the dataset by iterating over different combinations of timing parameters ( $t_d, t_c, \tau_s, \tau_h, C_L$ ). Key algorithm parameters are initialized, including the initial  $\tau_s$ , the step size  $\Delta$ , the previous C2Q delay value  $\tau_{C2Q}^{prev}$ , and a convergence threshold  $\epsilon$ . For each combination of input parameters,  $\tau_s$  is incremented by  $\Delta$  at each step, and a SPICE simulation is executed to compute the current



$\tau_{C2Q}$ . The difference between the current and previous  $\tau_{C2Q}$  values is then evaluated to monitor convergence. If the absolute difference  $|\tau_{C2Q}^{\text{curr}} - \tau_{C2Q}^{\text{prev}}|$  stays within  $\epsilon$  for four consecutive steps, the SPICE simulations for that specific set of  $t_d, t_c, \tau_h$ , and  $C_L$  are terminated. This condition ensures that the process continues as long as variations in  $\tau_{C2Q}$  are observed and is halted once no significant variation is detected over consecutive steps, indicating convergence. This approach enables efficient dataset generation with accurate  $\tau_{C2Q}$  values across varying parameter combinations while avoiding unnecessary computation. The resulting dataset is subsequently used to train the ML model for predicting  $\tau_{C2Q}$  in flip-flops.

The dataset generated using the above algorithm is preprocessed prior to being used for model training and evaluation. Here,  $X_f^{C2Q} \in R^{n \times m}$  represents the feature matrix, where  $n = 5000$  denotes the number of samples and  $m = 5$  denotes the number of features, and  $y_t^{C2Q} \in R^n$  represents the corresponding target labels. Similar to the *Latch-Predict* model, the features are subjected to min–max normalization to scale all features within the range  $[0, 1]$ . The dataset is then partitioned into a training set  $(X_{f,\text{train}}^{C2Q}, y_{t,\text{train}}^{C2Q})$  and a testing set  $(X_{f,\text{test}}^{C2Q}, y_{t,\text{test}}^{C2Q})$ . The model is trained using  $(X_{f,\text{train}}^{C2Q}, y_{t,\text{train}}^{C2Q})$ , and its performance is evaluated on  $(X_{f,\text{test}}^{C2Q}, y_{t,\text{test}}^{C2Q})$ .

After the training and testing datasets are generated, multiple ML models with varying complexities and data representation requirements are developed and evaluated. The training process is carefully managed to avoid overfitting. A comparative analysis of eight ML models, based on their NRMSE values, is presented in Table 4.4. To achieve optimal performance, extensive hyperparameter tuning is conducted for all models, and after evaluating multiple configurations, the best-performing model is selected. Among all models, the ANN demonstrates the highest accuracy in both training and testing, achieving the lowest test NRMSE. Table 4.4 summarizes the NRMSE values for both phases across different models. The ANN achieves the lowest NRMSE in training (0.0130) and testing (0.0133), outperforming the other models. Gradient Boosting and XGBoost follow closely with comparable NRMSE values, while Decision Tree and Random Forest exhibit slightly higher errors, indicating lower prediction accuracy relative to the ANN.

Table 4.5 details the hyperparameter tuning results for the ANN, including configurations for hidden layers, activation function (ReLU), solver (Adam), alpha (0.01),

Table 4.4: Model Performance Comparison

| Model                            | Train NRMSE   | Test NRMSE    |
|----------------------------------|---------------|---------------|
| Decision Tree                    | 0.0241        | 0.0253        |
| Random Forest                    | 0.0182        | 0.0196        |
| Gradient Boosting                | 0.0163        | 0.0164        |
| XGBoost                          | 0.0162        | 0.0165        |
| LightGBM                         | 0.0150        | 0.0158        |
| <b>Artificial Neural Network</b> | <b>0.0130</b> | <b>0.0133</b> |

Table 4.5: Hyperparameter Tuning Results (Stacked)

| Hidden Layers                    | Train NRMSE   | Test NRMSE    |
|----------------------------------|---------------|---------------|
| 10, 10                           | 0.0291        | 0.0309        |
| 10, 10, 10                       | 0.0227        | 0.0230        |
| 10, 10, 10, 10                   | 0.0179        | 0.0182        |
| 10, 10, 10, 10, 10               | 0.0223        | 0.0225        |
| 50, 50                           | 0.0181        | 0.0184        |
| 50, 50, 50                       | 0.0165        | 0.0173        |
| 50, 50, 50, 50                   | 0.0159        | 0.0164        |
| 50, 50, 50, 50, 50               | 0.0167        | 0.0173        |
| 100, 50, 10                      | 0.0146        | 0.0147        |
| 100, 100                         | 0.0158        | 0.0162        |
| 100, 100, 100                    | 0.0149        | 0.0153        |
| 100, 100, 100, 100               | 0.0135        | 0.0139        |
| 100, 100, 100, 100, 100          | 0.0151        | 0.0159        |
| 150, 150                         | 0.0141        | 0.0147        |
| 150, 150, 150                    | 0.0136        | 0.0144        |
| 150, 150, 150, 150               | 0.0133        | 0.0139        |
| 150, 150, 150, 150, 150          | 0.0137        | 0.0145        |
| 150, 100, 50, 10                 | 0.0143        | 0.0149        |
| 250, 250                         | 0.0145        | 0.0152        |
| 250, 250, 250                    | 0.0139        | 0.0145        |
| 250, 250, 250, 250               | 0.0138        | 0.0146        |
| 250, 250, 250, 250, 250          | 0.0148        | 0.0156        |
| <b>500, 500</b>                  | <b>0.0130</b> | <b>0.0133</b> |
| 500, 500, 500                    | 0.0142        | 0.0149        |
| 1000, 1000                       | 0.0158        | 0.0161        |
| 1000, 1000, 1000                 | 0.0141        | 0.0145        |
| 1000, 500, 250, 150, 100, 50, 10 | 0.0186        | 0.0190        |

and learning rate (0.001). It outlines various hidden layer combinations and their corresponding NRMSE values for training and testing. The optimal configuration, highlighted in bold, consists of two hidden layers with 500 nodes each, yielding the best performance with a training NRMSE of 0.0130 and a testing NRMSE of 0.0133. Interestingly, increasing the complexity of the architecture-such as adding more neurons (1000, 1000), or more hidden layers (500, 500, 500)-did not result in improved NRMSE. This suggests that simpler models are sufficient to capture the complex relationships in the data. Overall, the (500, 500) configuration provides the best trade-off between accuracy, generalization, and efficiency, making it the most suitable choice.

## 4.5 Comparison with Prior Work

Recent works [60]-[66] that leverage the interdependence between setup and hold times for circuit optimization have predominantly followed deterministic approaches, wherein process-induced variations affecting the probability of correct data capture are neglected. As a result, these works differ fundamentally from the approach adopted in this chapter, which is based on a probabilistic formulation and inherently accounts for such variations, thereby providing a more realistic characterization of timing behaviour. In this section, the differences between the two approaches are highlighted, and the accuracy advantages of the proposed model are demonstrated.

In previous works, the interdependence of various timing attributes (ST, HT, and C2Q delays) has been modelled using linear, polynomial, or piece-wise linear functions. More recently, ML-driven models have been employed in [19, 66] to capture the highly nonlinear behaviour of flip-flops. In these works, it is assumed that correct data latching occurs if the predicted delay  $\tau_{C2Q}$  is less than a bound on the nominal delay, i.e.,  $\tau_{C2Q} \leq \alpha \times \tau_{C2Q}^{Nom}$ , where  $\tau_{C2Q}^{Nom}$  denotes the nominal delay under large setup and hold skews, and  $\alpha > 1.0$  represents a delay threshold factor. An  $\alpha$  value of 1.1 was proposed in [19], while a more relaxed value of  $\alpha = 1.5$  was suggested in [66] for operation in the near-threshold voltage (NTV) domain.

In contrast, the proposed *LatchPredict* model introduces a probabilistic framework in which the likelihood of correct data latching ( $P_L$ ) is evaluated under varying circuit conditions. Rather than relying on fixed delay margins, the decision threshold  $\theta_{Model}$

is adaptively adjusted based on the desired confidence levels ( $1\sigma$ ,  $2\sigma$ ,  $3\sigma$ ), thereby enabling flexible precision–recall trade-offs. To assess the generalization capability and robustness of the proposed probabilistic model, a new test dataset comprising approximately 800 unseen samples is generated, spanning a wide range of  $t_c$ ,  $t_d$ ,  $\tau_h$ ,  $\tau_s$ , and  $C_L$ . It is noted that a direct one-to-one comparison with [19, 66] is not feasible, as those approaches employ deterministic delay-based waiver schemes, whereas the proposed methodology relies on a probabilistic framework that explicitly captures statistical variations and latching probability. Nevertheless, the limitations of the prior works that are addressed by the proposed approach are highlighted in the following paragraphs..

#### 4.5.1 Case-Level Comparison

First, a comparison is presented using selected examples from the 800 unseen samples for which discrepancies arise between deterministic methods and SPICE simulations with respect to successful data latching, and it is illustrated how such cases are effectively handled by the proposed method. Table 4.6 presents a comparison between the golden SPICE results and the results obtained using the proposed model. Each test case corresponds to a unique combination of circuit parameters ( $t_c, t_d, \tau_h, \tau_s, C_L$ ). The model-computed delay ( $\tau_{C2Q}^{Pred}$ ) and the predicted latching probability (Pred  $P_L$ ) are observed to match reasonably well with the SPICE-computed delay ( $\tau_{C2Q}^{Spice}$ ) and the corresponding latching probability (Actual  $P_L$ ). The “Actual (P/F)” column denotes the true latching outcome obtained from Monte Carlo simulations based on the actual latching probability (Actual  $P_L < 50\%$  indicates failure), where the letters P and F denote pass and fail conditions, respectively. For brevity, only samples in which flip-flops fail to latch data due to process-induced variations are shown; consequently, all entries are marked as F in the final column.

Table 4.7 compares the results obtained using deterministic methods from prior works with those produced by the probabilistic approach proposed in this work. The ANN-based approaches suggested in [19, 66] are observed to exhibit false-pass (FP) instances, in which failing samples (“F”) are incorrectly classified as passing (“P”). These misclassifications arise from the use of fixed deterministic thresholds (e.g.,  $\tau_{C2Q}^{Max} = 1.1 \times \tau_{C2Q}^{Nom}$  or  $1.5 \times \tau_{C2Q}^{Nom}$ ), which are unable to capture the nonlinear interdependence between setup and hold skews in the presence of process-induced delay variations. To

Table 4.6: Input and Measured/Predicted  $\tau_{C2Q}$  and  $P_L$  Values

| FF   | Input Parameters |       |          |          |       | Measured Values    |                      |              | Predicted Values    |            | Actual (P/F) |
|------|------------------|-------|----------|----------|-------|--------------------|----------------------|--------------|---------------------|------------|--------------|
|      | $t_c$            | $t_d$ | $\tau_h$ | $\tau_s$ | $C_L$ | $\tau_{C2Q}^{Nom}$ | $\tau_{C2Q}^{Spice}$ | Actual $P_L$ | $\tau_{C2Q}^{Pred}$ | Pred $P_L$ |              |
|      | (ps)             | (ps)  | (ps)     | (ps)     | (fF)  | (ps)               | (ps)                 | (%)          | (ps)                | (%)        |              |
| FF1  | 80.0             | 40.0  | 56.0     | 17.2     | 43.4  | 240.0              | 257.0                | 37.1         | 251.0               | 42.2       | F            |
| FF2  | 10.0             | 10.0  | 6.0      | 24.2     | 38.9  | 202.0              | 226.0                | 37.1         | 220.0               | 34.2       | F            |
| FF3  | 20.0             | 10.0  | 6.0      | 26.6     | 49.2  | 246.0              | 283.0                | 49.4         | 274.0               | 51.8       | F            |
| FF4  | 20.0             | 40.0  | 28.0     | 23.8     | 35.1  | 190.0              | 280.0                | 35.1         | 275.0               | 44.1       | F            |
| FF5  | 40.0             | 20.0  | 18.0     | 17.2     | 54.8  | 275.0              | 289.0                | 31.3         | 288.0               | 40.0       | F            |
| FF6  | 20.0             | 40.0  | 28.0     | 23.8     | 19.0  | 126.0              | 163.0                | 35.1         | 168.0               | 44.1       | F            |
| FF7  | 20.0             | 80.0  | 112.0    | 32.0     | 45.8  | 233.0              | 289.0                | 35.5         | 278.0               | 40.4       | F            |
| FF8  | 40.0             | 80.0  | 28.0     | 28.9     | 35.7  | 198.0              | 228.0                | 34.5         | 215.0               | 39.8       | F            |
| FF9  | 20.0             | 40.0  | 28.0     | 23.8     | 29.4  | 167.0              | 212.0                | 35.0         | 207.0               | 38.2       | F            |
| FF10 | 20.0             | 80.0  | 112.0    | 32.0     | 45.4  | 231.0              | 256.0                | 35.5         | 251.0               | 40.4       | F            |
| FF11 | 20.0             | 40.0  | 28.0     | 23.8     | 56.4  | 275.0              | 320.0                | 35.0         | 301.0               | 38.2       | F            |

Table 4.7: Comparison with Previous and Proposed Models

| FF   | Actual (P/F) | Previous Models |             |              |              | Proposed Models |            |            |            |
|------|--------------|-----------------|-------------|--------------|--------------|-----------------|------------|------------|------------|
|      | MC           | [19] (Pred)     | [66] (Pred) | [19] (Spice) | [66] (Spice) | 50              | 1 $\sigma$ | 2 $\sigma$ | 3 $\sigma$ |
| FF1  | <b>F</b>     | P               | P           | P            | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF2  | <b>F</b>     | P               | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF3  | <b>F</b>     | <b>F</b>        | P           | <b>F</b>     | P            | P               | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF4  | <b>F</b>     | <b>F</b>        | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF5  | <b>F</b>     | P               | P           | P            | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF6  | <b>F</b>     | <b>F</b>        | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF7  | <b>F</b>     | <b>F</b>        | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF8  | <b>F</b>     | P               | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF9  | <b>F</b>     | <b>F</b>        | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF10 | <b>F</b>     | P               | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |
| FF11 | <b>F</b>     | P               | P           | <b>F</b>     | P            | <b>F</b>        | <b>F</b>   | <b>F</b>   | <b>F</b>   |

ensure that this behavior is not an artifact of the ANN-based prediction of  $\tau_{C2Q}^{Pred}$ , a fair comparison is performed using SPICE-derived  $\tau_{C2Q}^{Spice}$  values. Even under this idealized scenario, which assumes a perfect model that exactly matches SPICE delays, several flip-flops that fail in Monte Carlo simulations (“F”) are still classified as passing (“P”) by the deterministic threshold rule. This observation demonstrates that the limitation

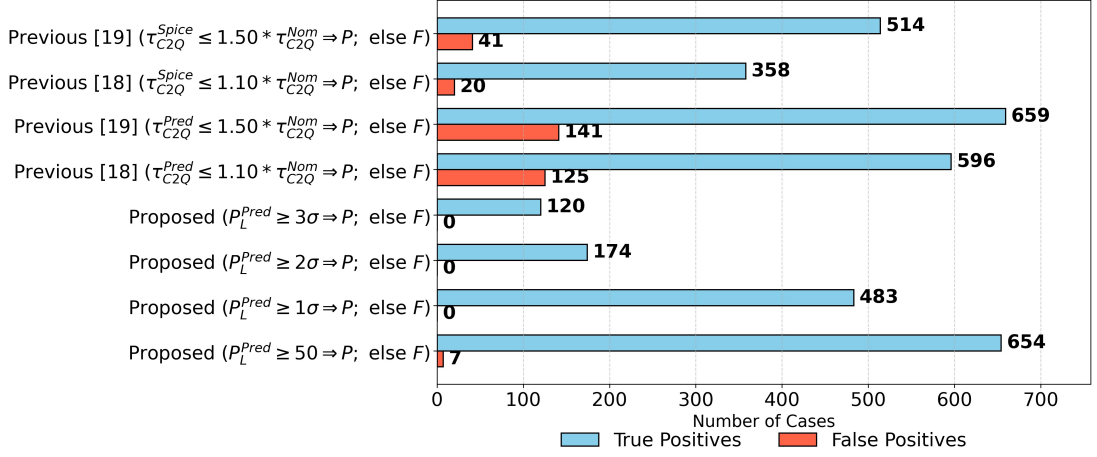


Figure 4.7: Comparison of true positive (TP) and false positive (FP) counts across all 800 test points for the proposed probabilistic LatchPredict and previous ANN-based timing models [19, 66]

originates not from the prediction model itself but from the inherent rigidity of fixed-threshold approaches. Between the two prior methods, the model in [19] exhibits fewer FP cases than that in [66], since [66] employs a more relaxed threshold ( $1.5 \times \tau_{C2Q}^{Nom}$ ), thereby classifying a larger number of failing instances as passing. Nevertheless, both deterministic approaches remain incapable of adapting to variation-induced delay shifts, leading to unreliable waiver decisions under marginal timing conditions. This limitation is expected to persist in other prior approaches that capture setup–hold interdependence using piecewise-linear, analytical, or ML-driven timing models, as these methods rely on SPICE-computed delays under simultaneously small setup and hold skew conditions without explicitly accounting for process-induced variations.

In contrast, the proposed probabilistic model adaptively determines the waiver boundary using the predicted probability of successful latching ( $P_L$ ). By evaluating confidence levels corresponding to  $\theta_{Model} = 50, 1\sigma, 2\sigma$ , and  $3\sigma$ , the P/F decision boundary is dynamically adjusted according to the desired reliability target. As shown in Tables 4.6 and 4.7, no FP cases are observed for confidence thresholds of  $2\sigma$  and  $3\sigma$ , while high overall accuracy is maintained. Even under the moderate confidence setting ( $\theta_{Model} = 50$ ), a precision exceeding 99% is achieved, thereby confirming the robustness and reliability of the proposed model in identifying true timing violations.

### 4.5.2 Complete Test Set Evaluation

Figure 4.7 compares the true-positive (TP) and false-positive (FP) counts for the proposed probabilistic *LatchPredict* framework and prior ANN-based timing models [19, 66], evaluated over all 800 test points. Here, a TP denotes a case in which both the Monte Carlo (MC) simulation and the model correctly predict a pass condition, whereas an FP corresponds to a case in which the model predicts a pass but the actual outcome is a fail. Minimization of FPs is critical during timing signoff, as FP cases can lead to design failure. Consequently, achieving a zero-FP rate is treated as the primary design objective, even if it results in a reduction in the number of TPs. The results reveal a clear TP–FP trade-off. As  $\theta_{model}$  is increased (from 50% to  $3\sigma$ ), FPs are completely eliminated, while the number of TPs is correspondingly reduced. In other words, the proposed model becomes more stringent in labeling a case as “P,” thereby preventing false waiver decisions at the cost of reduced waiver of marginal failures. This trade-off is desirable in VLSI design, where timing closure and signoff prioritize zero FPs (i.e., ensuring that violating flip-flops are correctly marked as failing), even if the total percentage of waived marginal violations is reduced. Hence, configurations such as  $\theta_{model} = 1\sigma$  or  $\theta_{model} = 2\sigma$  are identified as practical operating points, delivering zero FPs while maintaining meaningful waiver of failures.

In contrast, the prior ANN-based models [19, 66] exhibit significantly higher FP rates, 125 for [19] and 141 for [66], despite achieving relatively high TP counts (596 and 659, respectively). When evaluated using SPICE-derived  $\tau_{C2Q}$  values, FP counts of 20 for [19] and 41 for [66] are observed, with corresponding TP counts of 358 and 514, respectively. These results can be attributed to the deterministic nature of these approaches and their reliance on fixed delay threshold factors, which do not explicitly model process-induced variations. As a result, statistical shifts in timing due to random process variations are ignored, causing these deterministic models to waive violations even when MC simulations indicate unsafe operation. Therefore, although the deterministic ANN-based methods [19, 66] appear to waive a larger number of violations (higher TP counts), this is achieved at the cost of false optimism, which is unacceptable for design signoff. In contrast, the probabilistic approach adopted in *LatchPredict* ensures that all predicted violations correspond to FP = 0, thereby providing a safer and more robust prediction mechanism suitable for real-silicon timing validation.

## 4.6 Timing Violation Waiver

Figure 4.8 illustrates the framework of the proposed violation waiver. A failing flip-flop  $F$  is received from an STA tool, and a hierarchical filtering mechanism is applied to determine whether the given violation can be resolved. For this decision, the *LatchPredict* and *C2QPredict* models of the flip-flop are utilized from the augmented timing library. The timing attributes required for these models and filtering mechanisms, such as slack, arrival times, and slews, are obtained from the STA tool.

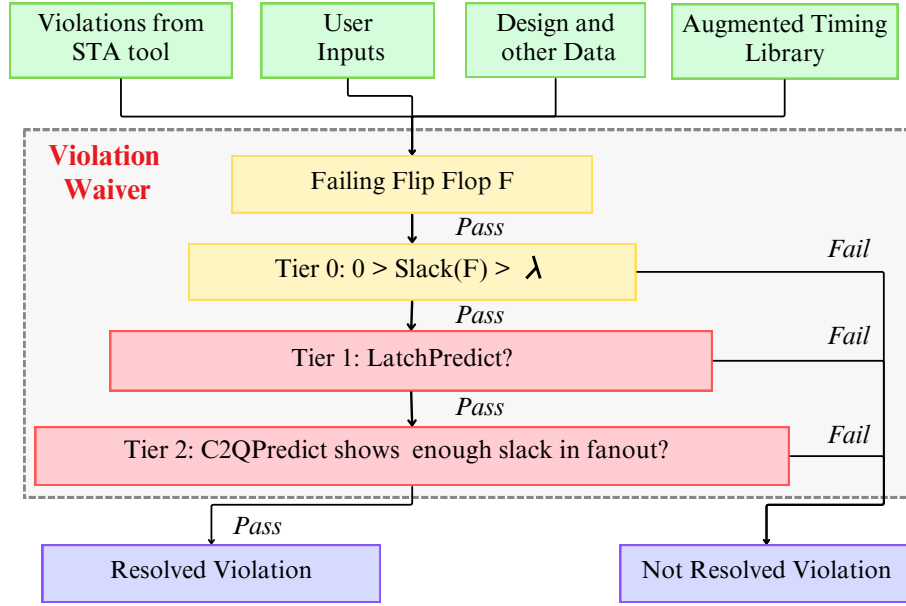


Figure 4.8: Timing violation waiver framework

### 4.6.1 Tier 0: Identifying marginal violations

Removal of pessimism in the flip-flop timing model alone is insufficient to resolve a violation with a large negative slack. Therefore, the violation waiver is applied only to marginally failing flip-flops. A violation is classified as marginal if the slack of the failing flip-flop  $F$  satisfies the following condition:

$$\lambda < \text{Slack}(F) < 0 \quad (4.3)$$

where  $\lambda$  denotes a user-defined negative slack margin. If the slack satisfies the above condition, the checking proceeds to the next tier; otherwise, the violation is declared unresolved.



### 4.6.2 Tier 1: Checking latching probability

In this tier, an evaluation using *LatchPredict* is performed to determine whether the marginally failing flip-flop can still latch data despite the violation. The model utilizes attributes extracted from the STA tool, namely  $\tau_s$ ,  $\tau_h$ ,  $t_d$ , and  $t_c$ , to predict whether the data will be latched successfully. If correct latching is predicted, the checking proceeds to the next tier; otherwise, the violation is declared unresolved.

### 4.6.3 Tier 2: Checking fanout

Although a marginally failing flip-flop  $F$  may latch data correctly, as identified in Tier 1, its C2Q delay may exceed the STA-predicted delay, which relies on a conventional flip-flop model. Consequently, this tier checks whether sufficient surplus slack exists in the fanout of the failing flip-flop to accommodate the increased C2Q delay.

The C2Q delay of the failing flip-flop  $F$  is computed using the *C2QPredict* model with the features  $\tau_s$ ,  $\tau_h$ ,  $t_d$ ,  $t_c$ , and  $C_L$  extracted from the STA tool. The increase in C2Q delay is computed as

$$\Delta = C2Q_{Model} - C2Q_{NOM}, \quad (4.4)$$

where  $C2Q_{Model}$  denotes the C2Q delay computed using the *C2QPredict* model, and  $C2Q_{NOM}$  denotes the nominal C2Q delay obtained from the STA tool using the conventional technology library. Subsequently, the flip-flop  $F_{fo}$  in the fanout of the failing flip-flop  $F$  with the minimum slack is identified using the STA tool. If the slack of  $F_{fo}$  exceeds  $\Delta$ , the increased C2Q delay of  $F$  does not introduce a violation in the fanout, and the violation is considered to have passed Tier 2.

In a special scenario, a failing flip-flop  $F$  may still be considered for waiver even when the slack of its fanout flip-flop  $F_{fo}$  is less than  $\Delta$ . If the critical path forms a cycle such that the begin and end points correspond to the same flip-flop, the Tier 1 latching probability check is repeated, since the increased C2Q delay directly impacts  $\tau_h$  and  $\tau_s$ . Furthermore, an additional condition is introduced in Tier 2 for such cases: if the slack of the same flip-flop is sufficient to absorb the  $\Delta$  (increase in C2Q delay) computed using the *C2QPredict* model, the violation may be waived.

A violation is declared resolved if all checks in the hierarchy are satisfied. Otherwise, the violation remains unresolved, requiring further analysis and potential design modifications.

#### 4.6.4 Integration of Violation Waiver into STA Flow

Once the ML models *LatchPredict* and *C2QPredict* are trained, the conventional timing library is augmented by embedding references to the *LatchPredict* and *C2QPredict* models within it. In a standard Liberty (.lib) file, each flip-flop contains timing arcs that describe setup, hold, and clock-to-Q characteristics. The existing libraries are augmented by introducing user-defined named attributes that store references to the trained ML-driven predictors.

For example, within the *timing()* group corresponding to a clock-to-Q arc of a flip-flop, new attributes are introduced in addition to the existing C2Q tables, as shown below:

```
timing() {
    related_pin : "CLK";
    timing_sense : positive_unate;

    /* Nominal tables already existing in the library */
    ...

    /* Augmented ML-driven models */
    latch_predict: "LatchPredict.json";
    c2q_predict: "C2QPredict.json";
}
```

Here,

- *latch\_predict* points to the ML model trained to predict flip-flop pass or fail behavior under marginal violations,
- *c2q\_predict* points to the ML model trained to predict the clock-to-Q delay under marginal violations.

During STA, the augmented Liberty file is imported by the STA tool in the same

manner as a conventional technology library. When a violation is reported, the ML-driven models can be additionally queried by the tool:

- At Tier 0, the slack is checked against the user-defined margin,
- At Tier 1, *LatchPredict* is invoked to evaluate whether the data can be latched correctly despite a marginal violation,
- At Tier 2, *C2QPredict* is invoked to compute the actual C2Q delay. This predicted delay is compared with the nominal C2Q delay obtained from the conventional table, and the fanout slack is evaluated accordingly.

Thus, the augmentation transforms the flip-flop timing specification from a table-based model into a hybrid representation that links conventional timing data with ML-driven predictors. The STA tool can import and utilize these augmented libraries to make violation waiver decisions within the STA flow.

## 4.7 Results

The proposed approach is validated on benchmark circuits from TAU CONTEST’19 [67], implemented using 45 nm cell libraries and SPICE models. SPICE simulations are performed in Virtuoso [24], while STA is conducted in Tempus [6] in graph-based mode. First, the *LatchPredict* and *C2QPredict* flip-flop timing models are developed for these libraries, which constitutes a one-time activity. Subsequently, the violation-waiving methodology described in the previous section is applied to various circuits. Detailed results obtained using the proposed technique on individual paths of these circuits are first presented, followed by the overall violation waiver results for the entire circuit.

### 4.7.1 Waiving Violation

A portion of the circuit extracted from the *leon2\_iccad* design is illustrated in Fig. 4.9. The flip-flop FF2 exhibits a timing violation with a slack value of  $-6$  ps. With a user-defined slack threshold of  $\lambda = -10$  ps, this violation is classified as marginal, thereby

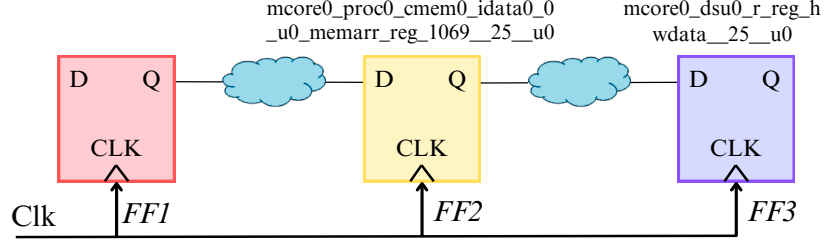


Figure 4.9: A portion of the circuit from leon2\_iccad.

Table 4.8: Waiving Violation ( $\lambda = -10$  ps,  $\theta_{\text{Model}} = 99.7\%$  ( $3\sigma$ ))

| $C_L^{\text{FF2}}$              | $t_c^{\text{FF2}}$                | $t_d^{\text{FF2}}$                         | $\tau_s^{\text{FF2}}$       | $\tau_h^{\text{FF2}}$                    |
|---------------------------------|-----------------------------------|--------------------------------------------|-----------------------------|------------------------------------------|
| 30 fF                           | 40 ps                             | 13 ps                                      | 23 ps                       | 19 ps                                    |
| $\text{Slack}^{\text{FF2}}$     | $P_{L,MC}^{\text{FF2}}$           | $\text{Latch}_{\text{SPICE}}^{\text{FF2}}$ | $P_{L,Mod}^{\text{FF2}}$    | $\text{Latch}_{\text{Mod}}^{\text{FF2}}$ |
| -4 ps                           | 100%                              | Pass                                       | 99.98%                      | Pass                                     |
| $C2Q_{\text{Nom}}^{\text{FF2}}$ | $C2Q_{\text{SPICE}}^{\text{FF2}}$ | $C2Q_{\text{Model}}^{\text{FF2}}$          | $\text{Slack}^{\text{FF3}}$ |                                          |
| 165 ps                          | 181 ps                            | 186 ps                                     | 57 ps                       |                                          |

satisfying the Tier 0 condition. To further evaluate the feasibility of waiving the violation, it is assessed whether FF2 can correctly latch data. The *LatchPredict* model is employed using the attributes extracted from STA, as listed in Table 4.8 (first two rows).

The model predicts correct latching with a probability of  $P_{L,Mod}^{\text{FF2}} = 99.98\%$  and a classification of *Pass*. This prediction is corroborated by Monte Carlo simulations, which yield  $P_{L,MC}^{\text{FF2}} = 100\%$  and a *Pass* classification. Consequently, the Tier 1 criterion is satisfied, and the check proceeds to Tier 2. The *C2QPredict* model estimates the C2Q delay of FF2 to be 186 ps using the timing attributes extracted from the STA tool. The nominal C2Q delay obtained from the STA tool is 165 ps, resulting in an increase in delay given by

$$\Delta = C2Q_{\text{Model}}^{\text{FF2}} - C2Q_{\text{Nom}}^{\text{FF2}} = 186 - 165 = 21 \text{ ps}.$$

To ensure that this increased delay does not introduce a violation in the fanout, the slacks of flip-flops capturing data launched by FF2 are examined. The flip-flop FF3 in the fanout of FF2 is found to exhibit the minimum slack, with a value of 57 ps. Since FF3 possesses sufficient surplus slack to accommodate the increased C2Q delay from FF2, the Tier 2 condition is also satisfied, and the timing violation at FF2 is waived. Additionally, the SPICE-computed C2Q delay (181 ps) closely matches the ML-predicted value (186 ps), thereby validating the proposed approach.

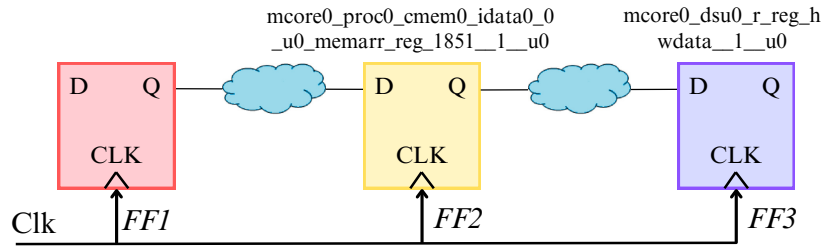


Figure 4.10: A portion of the circuit from leon2\_iccad.

Table 4.9: Detecting capture problem ( $\lambda = -10$  ps,  $\theta_{\text{Model}} = 99.7\%$  ( $3\sigma$ ))

|                   |                     |                       |                   |                     |
|-------------------|---------------------|-----------------------|-------------------|---------------------|
| $C_L^{FF2}$       | $t_c^{FF2}$         | $t_d^{FF2}$           | $\tau_s^{FF2}$    | $\tau_h^{FF2}$      |
| 41 fF             | 40 ps               | 17 ps                 | 20 ps             | -4 ps               |
| $Slack^{FF2}$     | $P_{L,MC}^{FF2}$    | $Latch_{SPICE}^{FF2}$ | $P_{L,Mod}^{FF2}$ | $Latch_{Mod}^{FF2}$ |
| -7 ps             | 0 %                 | Fail                  | 8%                | Fail                |
| $C2Q_{Nom}^{FF2}$ | $C2Q_{SPICE}^{FF2}$ | $C2Q_{Model}^{FF2}$   | $Slack^{FF3}$     |                     |
| 209 ps            | $\infty$            | $\infty$              | 803 ps            |                     |

### 4.7.2 Detecting capture problem

A portion of the circuit extracted from the *leon2\_iccad* design is illustrated in Fig. 4.10. The flip-flop FF2 exhibits a timing violation with a slack value of  $-7$  ps. With a user-defined slack threshold of  $\lambda = -10$  ps, this violation is classified as marginal and satisfies the Tier 0 condition. In Tier 1, using the attributes extracted from STA and shown in Table 4.9, the *LatchPredict* model predicts a latching probability of  $P_{L,Mod}^{FF2} = 8\%$  and a classification of *Fail*. This result is corroborated by Monte Carlo simulations, which yield  $P_{L,MC}^{FF2} = 0\%$  and a *Fail* classification, thereby confirming that the Tier 1 criterion is not met. As a result, the Tier 2 check is not performed.

The difference between the results presented in Table 4.8 (Section 4.7.1) and Table 4.9 (Section 4.7.2) can be explained as follows. In Section 4.7.1, both the setup and hold skews are positive ( $\tau_s^{FF2} = 23\text{ps}$  and  $\tau_h^{FF2} = 19\text{ps}$ ), leading to a latching probability close to 100%. In contrast, in Section 4.7.2, the hold skew becomes negative ( $\tau_h^{FF2} = -4\text{ps}$ ) and the setup skew is tighter (20ps). These adverse conditions result in latching failure and an infinite C2Q delay.

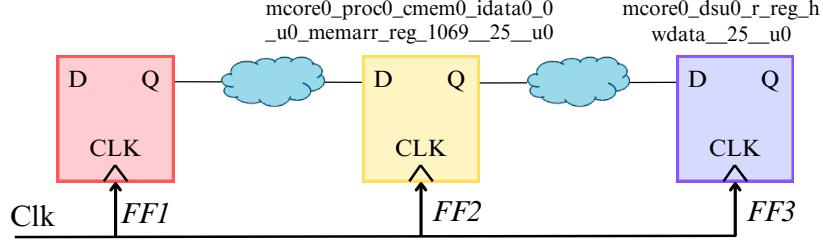


Figure 4.11: A portion of the circuit from leon2\_iccad.

Table 4.10: Detecting violation in the fanout ( $\lambda = -10$  ps,  $\theta_{\text{Model}} = 99.7\%$  ( $3\sigma$ ))

| $C_L^{\text{FF2}}$          | $t_c^{\text{FF2}}$                | $t_d^{\text{FF2}}$                         | $\tau_s^{\text{FF2}}$       | $\tau_h^{\text{FF2}}$             |
|-----------------------------|-----------------------------------|--------------------------------------------|-----------------------------|-----------------------------------|
| 36 fF                       | 40 ps                             | 10 ps                                      | 21 ps                       | 19 ps                             |
| $\text{Slack}^{\text{FF2}}$ | $P_{L,MC}^{\text{FF2}}$           | $\text{Latch}_{\text{SPICE}}^{\text{FF2}}$ | $P_{L,Mod}^{\text{FF2}}$    | $\text{Latch}_{Mod}^{\text{FF2}}$ |
| -6 ps                       | 100%                              | Pass                                       | 99.77%                      | Pass                              |
| $C2Q_{Nom}^{\text{FF2}}$    | $C2Q_{\text{SPICE}}^{\text{FF2}}$ | $C2Q_{Model}^{\text{FF2}}$                 | $\text{Slack}^{\text{FF3}}$ |                                   |
| 185 ps                      | 201 ps                            | 209 ps                                     | 12 ps                       |                                   |

### 4.7.3 Detecting violation in the fanout

A portion of the circuit extracted from the *leon2\_iccad* design is illustrated in Fig. 4.11. The flip-flop FF2 exhibits a timing violation with a slack value of  $-6$  ps. With a user-defined slack threshold of  $\lambda = -10$  ps, this violation is classified as marginal and satisfies the Tier 0 condition. In Tier 1, using information obtained from the STA tool as shown in Table 4.10, the *LatchPredict* model predicts correct latching with a probability of  $P_{L,Mod}^{\text{FF2}} = 99.67\%$  and a *Pass* classification. This prediction is corroborated by Monte Carlo simulations, which yield  $P_{L,MC}^{\text{FF2}} = 100\%$  and a *Pass* classification. Consequently, the Tier 1 criterion is satisfied, and the check proceeds to Tier 2.

The *C2QPredict* model estimates the C2Q delay of FF2 to be 209 ps using the timing attributes extracted from the STA tool, which closely aligns with the SPICE-computed delay of 201 ps. The nominal C2Q delay obtained from the STA tool is 185 ps, resulting in an increase of 24 ps. To ensure that this increased delay does not cause a violation in the fanout, the slacks of flip-flops capturing data launched by FF2 are examined. The flip-flop FF3 in the fanout of FF2 is found to exhibit the minimum slack, with a value of only 12 ps. Since FF3 does not have sufficient surplus slack to absorb the increased C2Q delay of FF2, the Tier 2 condition is not satisfied. Consequently, despite FF2 latching data correctly, the increased C2Q delay can introduce a new violation in an otherwise non-violating flip-flop in its fanout, and the violation at

FF2 is not waived.

Nevertheless, information regarding correct data latching at FF2, along with the potential failure at FF3 due to the increased C2Q delay, is provided to the designer as debug information. These insights can be leveraged during violation fixing. Notably, in this scenario, the issue can be resolved either by increasing the slack at FF2 or by increasing the slack at FF3, thereby offering greater flexibility for design optimization.

#### 4.7.4 Results on benchmark circuits

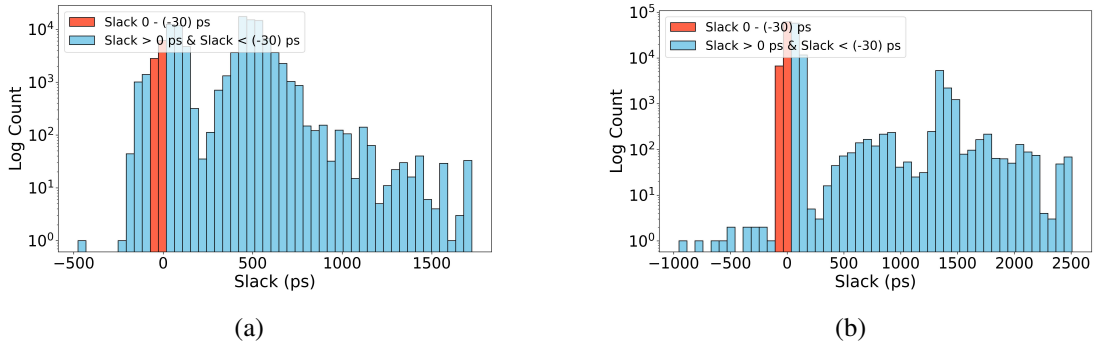


Figure 4.12: (a) Slack Histogram leon3mp\_iccad, (b) Slack Histogram leon2\_iccad.

Next, the results of the proposed methodology are presented on a design taken from the TAU Contest 2019 [67] and mapped to 45 nm cell libraries. First, STA is performed using the conventional technology library, and the results obtained using the marginal violation resolution framework with the augmented libraries containing the *LatchPredict* and *C2QPredict* flip-flop timing models are examined. The filtering percentage for each tier is reported and is defined as

$$\text{Filter \%} = \left( \frac{Y}{X} \right) \times 100 \%, \quad (4.5)$$

where  $X$  denotes the total number of violations provided as input to a given tier, and  $Y$  denotes the total number of violations that *Pass* that tier. The designer-controlled slack margin parameter ( $\lambda$ ) is varied to illustrate its impact on the proposed violation-waiving mechanism. In addition, the classification threshold  $\theta_{Model}$  is varied to examine its impact on precision and filtering percentage.

The slack histograms for *leon3mp\_iccad* (comprising 1.61 million instances and 149 K flip-flops) and *leon2\_iccad* (comprising 1.24 million instances and 108 K flip-

Table 4.11: Effect of  $\lambda$  at  $\theta_{Model}=50\%$  for leon2\_iccad and leon3mp\_iccad

| leon3mp_iccad (#instances: 1.24M, #flip-flops: 108K, #total viol: 8,477) |                        |        |     |    |        |        |      |        |                   |                      |                  |
|--------------------------------------------------------------------------|------------------------|--------|-----|----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                            | Marginal<br>Violations | Tier 1 |     |    |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                          |                        | Pass   | TP  | FP | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                    | 3322                   | 772    | 726 | 46 | 25%    | 0.0149 | 762  | 93%    | 46                | 23%                  | 55               |
| -20ps                                                                    | 2471                   | 772    | 726 | 46 | 33%    | 0.0149 | 762  | 93%    | 46                | 31%                  | 44               |
| -10ps                                                                    | 1454                   | 716    | 670 | 46 | 49%    | 0.0149 | 706  | 96%    | 46                | 49%                  | 30               |
| -5ps                                                                     | 790                    | 429    | 383 | 46 | 60%    | 0.0143 | 421  | 89%    | 46                | 53%                  | 21               |
| -3ps                                                                     | 556                    | 312    | 266 | 46 | 64%    | 0.0143 | 306  | 85%    | 46                | 52%                  | 17               |
| -2ps                                                                     | 434                    | 258    | 218 | 40 | 69%    | 0.0143 | 254  | 85%    | 40                | 59%                  | 16               |

| leon2_iccad (#instances: 1.61M, #flip-flops: 149K, #total viol: 29,309) |                        |        |      |     |        |        |      |        |                   |                      |                  |
|-------------------------------------------------------------------------|------------------------|--------|------|-----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                           | Marginal<br>Violations | Tier 1 |      |     |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                         |                        | Pass   | TP   | FP  | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                   | 16998                  | 3963   | 3837 | 126 | 24%    | 0.0164 | 3886 | 95%    | 126               | 23%                  | 239              |
| -20ps                                                                   | 11160                  | 3963   | 3837 | 126 | 37%    | 0.0164 | 3886 | 95%    | 126               | 35%                  | 160              |
| -10ps                                                                   | 4924                   | 3780   | 3659 | 121 | 79%    | 0.0164 | 3704 | 95%    | 121               | 75%                  | 76               |
| -5ps                                                                    | 2343                   | 1993   | 1916 | 77  | 94%    | 0.0174 | 1954 | 94%    | 77                | 83%                  | 42               |
| -3ps                                                                    | 1342                   | 1116   | 1047 | 69  | 91%    | 0.0181 | 1092 | 92%    | 69                | 81%                  | 28               |
| -2ps                                                                    | 1025                   | 865    | 807  | 58  | 90%    | 0.0187 | 847  | 92%    | 58                | 83%                  | 24               |

flops) are presented in Fig. 4.12. As expected for designs close to the signoff stage, the majority of paths exhibit positive slack; however, a significant concentration of paths is clustered close to zero slack. These near-zero slack paths are particularly sensitive to pessimism in flip-flop timing models. In both designs, many violations are observed within the narrow range of 0 to  $-30$  ps (highlighted in red). To illustrate the trade-off between  $\lambda$  and the percentage of violations filtered,  $\lambda = -30$  ps is considered as a reference point. It should be noted that  $\lambda = -30$  ps is not proposed as a fixed global threshold; rather, it is used to demonstrate the trade-off in the violation-filtering process, as summarized in Tables 4.11–4.13. Based on this value of  $\lambda$ , marginal violations are extracted from the STA reports for further analysis (Tables 4.11–4.13).

The *LatchPredict* model is first applied in Tier 1 to assess whether marginally failing flip-flops can still capture data despite timing violations. Flip-flops that *Pass* Tier 1 are subsequently evaluated in Tier 2 using the *C2QPredict* model, which estimates the C2Q delay more accurately than the conventional STA flip-flop model. At this stage, it is checked whether the available surplus slack in the fanout paths of the failing flip-flop is sufficient to accommodate the potentially increased C2Q delay. The percentage of violations successfully resolved through this process is reported as the waiver percentage.

The results presented in Tables 4.11–4.13 clearly illustrate how the choice of  $\lambda$  and  $\theta_{Model}$  affects the filtering of marginal violations. At  $\theta_{Model} = 50\%$  (Table 4.11), it



Table 4.12: Effect of  $\lambda$  at  $\theta_{Model}=95.4\%$  for *leon2\_iccad* and *leon3mp\_iccad*

| leon3mp_iccad (#instances: 1.24M, #flip-flops: 108K, #total viol: 8,477) |                        |        |     |    |        |        |      |        |                   |                      |                  |
|--------------------------------------------------------------------------|------------------------|--------|-----|----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                            | Marginal<br>Violations | Tier 1 |     |    |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                          |                        | Pass   | TP  | FP | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                    | 3322                   | 417    | 417 | 0  | 13%    | 0.0149 | 412  | 99%    | 0                 | 12%                  | 53               |
| -20ps                                                                    | 2471                   | 417    | 417 | 0  | 17%    | 0.0149 | 412  | 99%    | 0                 | 17%                  | 42               |
| -10ps                                                                    | 1454                   | 417    | 417 | 0  | 29%    | 0.0149 | 412  | 99%    | 0                 | 28%                  | 29               |
| -5ps                                                                     | 790                    | 281    | 281 | 0  | 36%    | 0.0143 | 277  | 99%    | 0                 | 35%                  | 20               |
| -3ps                                                                     | 556                    | 194    | 194 | 0  | 35%    | 0.0143 | 191  | 98%    | 0                 | 34%                  | 16               |
| -2ps                                                                     | 434                    | 159    | 159 | 0  | 37%    | 0.0143 | 157  | 98%    | 0                 | 36%                  | 15               |

| leon2_iccad (#instances: 1.61M, #flip-flops: 149K, #total viol: 29,309) |                        |        |      |    |        |        |      |        |                   |                      |                  |
|-------------------------------------------------------------------------|------------------------|--------|------|----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                           | Marginal<br>Violations | Tier 1 |      |    |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                         |                        | Pass   | TP   | FP | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                   | 16998                  | 1743   | 1743 | 0  | 11%    | 0.0167 | 1713 | 98%    | 0                 | 10%                  | 232              |
| -20ps                                                                   | 11160                  | 1743   | 1743 | 0  | 16%    | 0.0167 | 1713 | 98%    | 0                 | 15%                  | 152              |
| -10ps                                                                   | 4924                   | 1743   | 1743 | 0  | 35%    | 0.0167 | 1713 | 98%    | 0                 | 34%                  | 73               |
| -5ps                                                                    | 2343                   | 1382   | 1382 | 0  | 59%    | 0.0166 | 1357 | 98%    | 0                 | 57%                  | 39               |
| -3ps                                                                    | 1342                   | 791    | 791  | 0  | 59%    | 0.0165 | 775  | 97%    | 0                 | 57%                  | 26               |
| -2ps                                                                    | 1025                   | 619    | 619  | 0  | 60%    | 0.0165 | 607  | 98%    | 0                 | 59%                  | 22               |

Table 4.13: Effect of  $\lambda$  at  $\theta_{Model}=99.7\%$  for *leon2\_iccad* and *leon3mp\_iccad*

| leon3mp_iccad (#instances: 1.24M, #flip-flops: 108K, #total viol: 8,477) |                        |        |     |    |        |        |      |        |                   |                      |                  |
|--------------------------------------------------------------------------|------------------------|--------|-----|----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                            | Marginal<br>Violations | Tier 1 |     |    |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                          |                        | Pass   | TP  | FP | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                    | 3322                   | 320    | 320 | 0  | 10%    | 0.0148 | 318  | 99%    | 0                 | 10%                  | 52               |
| -20ps                                                                    | 2471                   | 320    | 320 | 0  | 13%    | 0.0148 | 318  | 99%    | 0                 | 13%                  | 42               |
| -10ps                                                                    | 1454                   | 320    | 320 | 0  | 22%    | 0.0148 | 318  | 99%    | 0                 | 26%                  | 29               |
| -5ps                                                                     | 790                    | 232    | 232 | 0  | 29%    | 0.0143 | 231  | 99%    | 0                 | 29%                  | 20               |
| -3ps                                                                     | 556                    | 149    | 149 | 0  | 27%    | 0.0145 | 148  | 99%    | 0                 | 27%                  | 16               |
| -2ps                                                                     | 434                    | 120    | 120 | 0  | 27%    | 0.0145 | 119  | 99%    | 0                 | 27%                  | 15               |

| leon2_iccad (#instances: 1.61M, #flip-flops: 149K, #total viol: 29,309) |                        |        |      |    |        |        |      |        |                   |                      |                  |
|-------------------------------------------------------------------------|------------------------|--------|------|----|--------|--------|------|--------|-------------------|----------------------|------------------|
| Slack<br>Margin ( $\lambda$ )                                           | Marginal<br>Violations | Tier 1 |      |    |        | Tier 2 |      |        | SPICE<br>Mismatch | Violations<br>Waived | Runtime<br>(sec) |
|                                                                         |                        | Pass   | TP   | FP | Filter | NRMSE  | Pass | Filter |                   |                      |                  |
| -30ps                                                                   | 16998                  | 1376   | 1376 | 0  | 8%     | 0.0191 | 1356 | 99%    | 0                 | 8%                   | 219              |
| -20ps                                                                   | 11160                  | 1376   | 1376 | 0  | 12%    | 0.0191 | 1356 | 99%    | 0                 | 12%                  | 146              |
| -10ps                                                                   | 4924                   | 1376   | 1376 | 0  | 28%    | 0.0191 | 1356 | 99%    | 0                 | 28%                  | 65               |
| -5ps                                                                    | 2343                   | 1156   | 1156 | 0  | 49%    | 0.0194 | 1139 | 99%    | 0                 | 49%                  | 36               |
| -3ps                                                                    | 1342                   | 635    | 635  | 0  | 47%    | 0.0197 | 627  | 99%    | 0                 | 47%                  | 24               |
| -2ps                                                                    | 1025                   | 487    | 487  | 0  | 47%    | 0.0197 | 482  | 99%    | 0                 | 47%                  | 21               |

is observed for both *leon2\_iccad* and *leon3mp\_iccad* that a large number of violations lie within the  $-30$  ps to  $-2$  ps range. As  $\lambda$  is reduced from  $-2$  ps to  $-30$  ps, the number of marginal violations considered for waiving steadily increases, while the filtering rate decreases significantly. Moreover, it is observed that decreasing  $\lambda$  from  $-20$  ps to  $-30$  ps does not result in any additional marginal violations being waived. For example, in *leon2\_iccad*, the filtering rate is reduced from approximately 83% at  $-2$  ps to approximately 23% at  $-30$  ps, while the NRMSE values remain within a narrow and stable range (0.018–0.016), demonstrating that the ML predictions remain both accu-

rate and robust. A similar trend is observed for *leon3mp\_iccad*, where the filtering rate is reduced from over 62% at  $-2$  ps to approximately 22% at  $-30$  ps. A comparable reduction in filtering rate with decreasing  $\lambda$  is also observed in Tables 4.12 and 4.13. These results confirm that  $\lambda = -20$  ps represents a reasonable pessimism bound, below which no additional marginal violations are waived, and that almost all marginal violations that can be waived are captured within this margin. A further reduction in  $\lambda$  would therefore incur additional computational effort without any corresponding benefit in violation waiving.

At  $\theta_{Model} = 94.5\%$  ( $2\sigma$ , Table 4.12), the classifier is observed to become more conservative. All false positives (FPs) are eliminated across both benchmarks, ensuring that no marginal violations are mistakenly waived. However, this stricter threshold reduces the waiver rates compared to the 50% case. At  $\theta_{Model} = 99.7\%$  ( $3\sigma$ , Table 4.13), the methodology becomes highly conservative, prioritizing correctness over filtering aggressiveness. While FPs remain completely eliminated, the waiver rates are further reduced. For instance, in *leon2\_iccad* with  $\lambda = -30$  ps, the number of violations waived is reduced from 3886 at  $\theta_{Model} = 50\%$  to 1713 at  $\theta_{Model} = 94.5\%$ , and further to 1356 at  $\theta_{Model} = 99.7\%$ . This trade-off demonstrates that very high values of  $\theta_{Model}$  provide maximum confidence in the predictions at the expense of reduced filtering capability.

In summary, Tables 4.11–4.13 collectively demonstrate that (i)  $\lambda = -20$  ps is sufficient to account for pessimism arising from flip-flop timing models, as further reduction does not waive additional marginal violations, (ii) lower values of  $\theta_{Model}$  (50%) enable aggressive filtering while allowing a small fraction of FPs, and (iii) higher values of  $\theta_{Model}$  (94.5%, 99.7%) eliminate FPs entirely at the cost of reduced filtering aggressiveness. These parameters thus provide practical design knobs for balancing efficiency and robustness based on application requirements.

It is noted that  $\lambda$  is not universally fixed. Although the proposed models are trained on data spanning a wide range of slews and skews, enabling context-aware analysis, the selection of  $\lambda$  should be guided by the trade-off trends observed in Tables 4.11–4.13. In the presented experiments,  $\lambda = -20$  ps is found to adequately capture the worst-case slack of failing flip-flops whose violations can be waived. However, the optimal value of  $\lambda$  can vary across different designs and technology nodes and should therefore be selected based on the observed trade-off between the number of waived

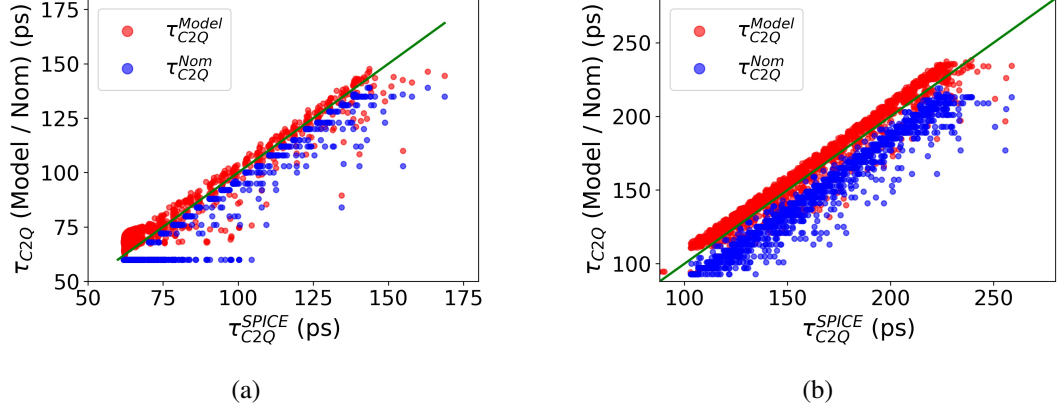


Figure 4.13: Correlation plots for C2Q Delay (a) leon3mp\_iccad, (b) leon2\_iccad.

violations and runtime. As shown in Tables 4.11–4.13, increasing the slack margin from  $-20$  ps to  $-30$  ps increases runtime while not enabling any additional violations to be waived. Consequently,  $\lambda$  should be chosen to balance the trade-off between the number of waived violations and computational efficiency. In other words, a larger slack margin (i.e., a more negative  $\lambda$ ) does not necessarily result in additional violations being waived and may instead introduce unnecessary runtime overhead.

In addition to NRMSE, correlation plots are provided in Fig. 4.13, comparing  $\tau_{C2Q}$  obtained from SPICE, model predictions, and nominal values. These plots serve as a direct accuracy check by illustrating the alignment of each prediction with the  $y = x$  reference line. The results demonstrate that the model predictions closely follow the SPICE values, with most points concentrated around the diagonal, indicating strong agreement. In contrast, the nominal values are observed to systematically underestimate the actual delays, as reflected by their deviation below the reference line. This visual evidence further reinforces that the proposed model not only achieves a low NRMSE but also reliably captures the true delay trends.

### Comparison with the previous work on waiving violations

As discussed in Section 4.5, prior works [19, 66] employed ANN-based models to capture nonlinear flip-flop behavior by predicting  $\tau_{C2Q}$  as a function of  $t_c$ ,  $t_d$ ,  $\tau_h$ ,  $\tau_s$ , and  $C_L$ . It should be noted that a direct one-to-one comparison with [19, 66] is not possible, since both approaches adopt deterministic, delay-based waiver rules, whereas the proposed method generalizes this concept into a probabilistic domain by explicitly capturing process variations. Nevertheless, to demonstrate the performance differences,

Table 4.14: Comparison of Previous [19, 66] and Proposed Methodology ( $\lambda=10$ ps)

| Design        | $\theta_{Model}$ | Marginal Violations | Previous Approach [19]<br>$\tau_{C2Q}^{Max} = 1.10 \times \tau_{C2Q}^{Nom}$ |      |           |        | Proposed Approach |      |     |           |                |        |
|---------------|------------------|---------------------|-----------------------------------------------------------------------------|------|-----------|--------|-------------------|------|-----|-----------|----------------|--------|
|               |                  |                     | TP                                                                          | FP   | Precision | Waived | Tier-1            |      |     |           | Tier-2<br>Pass | Waived |
|               |                  |                     |                                                                             |      |           |        | Pass              | TP   | FP  | Precision |                |        |
| leon3mp_iccad | $3\sigma$        | 1454                | 567                                                                         | 71   | 88%       | 30%    | 320               | 320  | 0   | 100%      | 318            | 22%    |
|               | $2\sigma$        | 1454                | 567                                                                         | 71   | 88%       | 39%    | 417               | 417  | 0   | 100%      | 412            | 28%    |
|               | $1\sigma$        | 1454                | 567                                                                         | 71   | 88%       | 39%    | 684               | 661  | 23  | 97%       | 676            | 46%    |
|               | 50%              | 1454                | 567                                                                         | 71   | 88%       | 39%    | 716               | 670  | 46  | 88%       | 706            | 49%    |
| leon2_iccad   | $3\sigma$        | 4924                | 1431                                                                        | 955  | 60%       | 29%    | 1376              | 1376 | 0   | 100%      | 1356           | 28%    |
|               | $2\sigma$        | 4924                | 1431                                                                        | 955  | 60%       | 29%    | 1743              | 1743 | 0   | 100%      | 1742           | 35%    |
|               | $1\sigma$        | 4924                | 1431                                                                        | 955  | 60%       | 29%    | 3632              | 3585 | 47  | 98%       | 3565           | 72%    |
|               | 50%              | 4924                | 1431                                                                        | 955  | 60%       | 29%    | 3780              | 3659 | 121 | 96%       | 3704           | 75%    |
| Design        | $\theta_{Model}$ | Marginal Violations | Previous Approach [66]<br>$\tau_{C2Q}^{Max} = 1.50 \times \tau_{C2Q}^{Nom}$ |      |           |        | Proposed Approach |      |     |           |                |        |
|               |                  |                     | TP                                                                          | FP   | Precision | Waived | Tier-1            |      |     |           | Tier-2<br>Pass | Waived |
|               |                  |                     |                                                                             |      |           |        | Pass              | TP   | FP  | Precision |                |        |
| leon3mp_iccad | $3\sigma$        | 1454                | 855                                                                         | 358  | 70%       | 59%    | 320               | 320  | 0   | 100%      | 318            | 22%    |
|               | $2\sigma$        | 1454                | 855                                                                         | 358  | 70%       | 59%    | 417               | 417  | 0   | 100%      | 412            | 28%    |
|               | $1\sigma$        | 1454                | 855                                                                         | 358  | 70%       | 59%    | 684               | 661  | 23  | 97%       | 676            | 46%    |
|               | 50%              | 1454                | 855                                                                         | 358  | 70%       | 59%    | 716               | 670  | 46  | 88%       | 706            | 48%    |
| leon2_iccad   | $3\sigma$        | 4924                | 1518                                                                        | 1057 | 59%       | 31%    | 1376              | 1376 | 0   | 100%      | 1356           | 28%    |
|               | $2\sigma$        | 4924                | 1518                                                                        | 1057 | 59%       | 31%    | 1743              | 1743 | 0   | 100%      | 1742           | 35%    |
|               | $1\sigma$        | 4924                | 1518                                                                        | 1057 | 59%       | 31%    | 3632              | 3585 | 47  | 98%       | 3565           | 72%    |
|               | 50%              | 4924                | 1518                                                                        | 1057 | 59%       | 31%    | 3780              | 3659 | 121 | 96%       | 3704           | 75%    |

experiments were conducted on the *leon2\_iccad* and *leon3mp\_iccad* benchmarks using the waiver rule defined in [19, 66], where a violation is considered to pass if  $\tau_{C2Q}^{Pred} \leq \alpha \times \tau_{C2Q}^{Nom}$ , and to fail otherwise.

It is emphasized that the contribution of this work is not intended as a replacement for deterministic C2Q characterization, but rather as a probabilistic sign-off augmentation that addresses marginal violations not handled by prior valid-region ANN models. The methodology proposed in this work advances beyond [19, 66] through the introduction of a two-tier, ML-driven hierarchical violation resolution framework that unifies deterministic accuracy with probabilistic safety. In Tier 1, the *LatchPredict* model is used to estimate the probability of correct data latching for the given setup and hold skews. In Tier 2, a refined C2Q delay is computed and it is verified that sufficient surplus slack exists in the fanout before the violation is waived.

Table 4.14 summarizes the comparison between the prior deterministic approaches [19, 66] and the proposed probabilistic two-tier methodology under different values of  $t_{C2Q}^{Max} = \alpha \times \tau_{C2Q}^{Nom}$  and  $\theta_{Model}$ . At the tighter bound of  $\alpha = 1.10$ , only a small fraction of marginal violations satisfy  $\tau_{C2Q}^{Pred} < t_{C2Q}^{Max}$ , resulting in a limited number of true positives (TPs) and consequently lower waiver percentages (39%, 29%). As the bound is

Table 4.15: Evaluation of Runtime Performance

| Library Creation              |                           |                  |                               |                           |                  |
|-------------------------------|---------------------------|------------------|-------------------------------|---------------------------|------------------|
| LatchPredict                  |                           |                  | C2QPredict                    |                           |                  |
| # Data                        | # Train (80%)             | # Val (20%)      | # Data                        | # Train (80%)             | # Val (20%)      |
| 1230                          | 984                       | 246              | 4907                          | 3923                      | 984              |
| Data Collection               | Model Training            | Model Validation | Data Collection               | Model Training            | Model Validation |
| 24 hr                         | 27 sec                    | 13 sec           | 1.52 hr                       | 62 sec                    | 16 sec           |
| Inference                     |                           |                  |                               |                           |                  |
| LatchPredict                  |                           |                  | C2QPredict                    |                           |                  |
| # Test Data (leon2 + leon3mp) | Runtime (leon2 + leon3mp) |                  | # Test Data (leon2 + leon3mp) | Runtime (leon2 + leon3mp) |                  |
| 20,310                        | 0.64 sec                  |                  | 5,436                         | 0.15 sec                  |                  |

relaxed to  $\alpha = 1.50$ , a larger number of marginal violations fall within the waiver window, increasing both TP and false positives (FP), and yielding higher waiver percentages (59%, 31%). However, the corresponding precision drops to (70%, 59%) from (88%, 60%) due to the increase in FP.

In contrast, the proposed methodology consistently delivers high precision ( $\geq 96\%$ ) while maintaining comparable waiver percentages across both benchmarks. For instance, in the *leon2\_iccad* design, the method in [19] identified 1431 TP but also produced 955 FP, limiting the precision to 60%. In comparison, the proposed Tier 1 framework achieved 1376 TP with zero FP, corresponding to a precision of 100%. After Tier 2 refinement, 1356 cases were confirmed as safe waivers, representing 28% of marginal violations. Similarly, for *leon3mp\_iccad*, the method in [19] achieved 88% precision with 567 TP and 71 FP, whereas the proposed Tier 1 framework achieved 100% precision with 320 TP and no FP. Tier 2 subsequently confirmed 318 waivers, corresponding to 22% of marginal violations. As  $\theta_{Model}$  is relaxed (e.g., from  $3\sigma$  to  $1\sigma$  or 50%), a small number of FP are observed in the proposed approach (e.g., 46 in *leon3mp\_iccad* and 121 in *leon2\_iccad* at 50%). Nevertheless, even under these relaxed conditions, the FP rate remains substantially lower than that of [19], while maintaining very high precision in the range of 96–97%. This tunable control of  $\theta_{Model}$  enables a practical balance between conservativeness and the number of violations waived, allowing designers to select an optimal operating point based on sign-off safety requirements.

### 4.7.5 Evaluation of Runtime Performance

The runtime performance of the proposed framework was evaluated on a Linux system equipped with a 1.8 GHz quad-core processor and 4 GB RAM. The results are summarized in Table 4.15, which is divided into two main phases: library creation and inference.

#### Library Creation

This phase involves the generation of training data, training of the ML models, and their validation. For *LatchPredict*, a total of 1,230 data points were prepared, with 984 used for training and 246 for validation. Data collection using SPICE Monte Carlo simulations required approximately 24 hours, followed by model training and validation times of 27 s and 13 s, respectively. For *C2QPredict*, 4,907 data points were collected, of which 3,923 were used for training and 984 for validation, requiring a total data generation time of 1.52 hours. Model training and validation required 62 s and 16 s, respectively. Since the timing library is generated only once and reused across multiple chip designs, the one-time cost of data collection and training is amortized over subsequent uses. Furthermore, the data collection time can be further reduced by parallelizing SPICE simulations on modern hardware platforms.

#### Inference

Once the libraries are created, inference is performed with negligible runtime overhead. Using *LatchPredict*, 20,310 test queries (from the *leon2\_iccad* and *leon3mp\_iccad* designs) were processed in only 0.64 s. Similarly, *C2QPredict* processed 5,436 test queries in 0.15 s.

These results demonstrate the feasibility of integrating the proposed ML-driven models into conventional timing libraries for STA, enabling efficient resolution of marginal timing violations. Although the library creation phase incurs a relatively high one-time cost, primarily due to Monte Carlo simulations, the rapid inference speed significantly outweighs this overhead, making the proposed approach practical for resolving marginal timing violations in large-scale designs.

#### 4.7.6 Using 7nm Libraries (ASAP 7nm PDK)

To demonstrate the applicability of the proposed framework in advanced technology nodes, experiments were conducted using 7 nm libraries based on the ASAP PDK [27], and the results are presented in this section. Since native Monte Carlo models are not available in the 7 nm ASAP PDK, Monte Carlo simulations were performed using the methodology presented in [71]. Local process variations were modeled using Gaussian distributions for four key device parameters-channel length, fin width, oxide thickness, and work function-where nominal library values were used as the mean, and standard deviations of 4%, 4%, 1.33%, and 1% were assumed, respectively [71]. These variation magnitudes are further supported by [72], which employed work-function fluctuation variations in the 1–5% range for ASAP7 7 nm FinFET circuits.

For various combinations of  $\tau_s$  and  $\tau_h$ , the probability of data latching  $P_L$  and the corresponding  $\tau_{C2Q}$  values were observed, as shown in Fig. 4.14. Figure 4.14a illustrates the impact of  $\tau_s$  on  $P_L$  under different  $\tau_h$  values. When  $\tau_s$  is sufficiently large,  $P_L = 100\%$ , indicating reliable data latching. However, as  $\tau_s$  decreases,  $P_L$  rapidly falls to 0%, signifying failure to latch the data. A threshold  $\theta_{SPICE} = 0.5$  is defined such that  $P_L > \theta_{SPICE}$  implies that the flip-flop is more likely to latch correctly. The setup skew corresponding to this threshold is denoted as  $L_{setup}$ . From Fig. 4.14b, it is evident that  $L_{setup}$  decreases with increasing  $\tau_h$ . Furthermore, Fig. 4.14c depicts the dependence of  $\tau_{C2Q}$  on both  $\tau_s$  and  $\tau_h$ . It is observed that reducing either  $\tau_s$  or  $\tau_h$  leads to an increase in  $\tau_{C2Q}$ . These results exhibit trends consistent with those observed for the 45 nm technology node.

#### LatchPredict (ASAP 7 nm PDK)

The same data generation, preprocessing, model selection, and evaluation methodology used for developing the *LatchPredict* model for the 45 nm technology was followed for the 7 nm library. The dataset was generated using Algorithm 1 and comprised  $n = 1813$  samples with  $m = 4$  features, which were normalized using min-max scaling. Several ML models were evaluated, and an ensemble of GBR, CatBoost, and ANN with weights 0.2/0.2/0.6 was found to provide the best performance, achieving an NRMSE of 0.9430 for training and 0.1005 for testing. After regression, a classifier was developed to predict  $P_L$  using thresholds  $\theta_{SPICE}$  (SPICE) and  $\theta_{Model}$  (ML). When  $\theta_{SPICE} = 50\%$  and

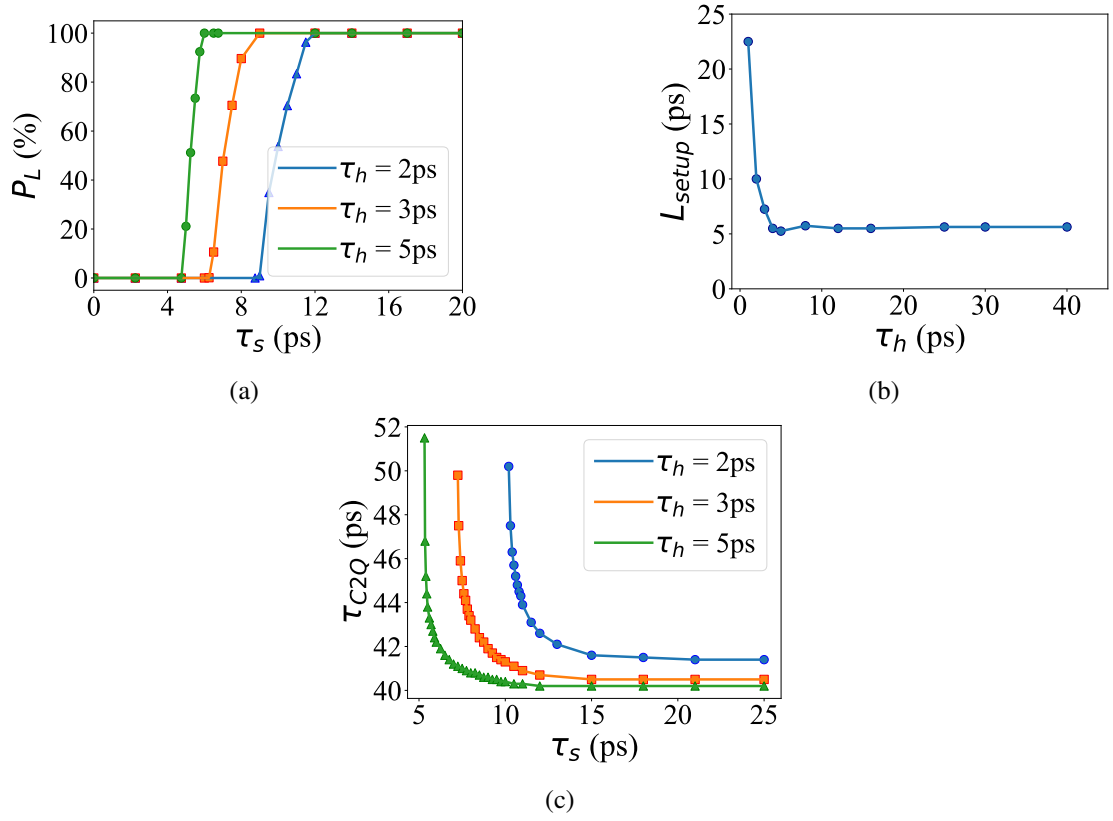


Figure 4.14: (a) Setup skew ( $\tau_s$ ) vs. probability of latching ( $P_L$ ), (b) latching threshold for the setup skew  $L_{setup}$  vs. hold skew ( $\tau_h$ ), (c) C2Q delay ( $\tau_{C2Q}$ ) vs. setup skew ( $\tau_s$ ).

$\theta_{\text{Model}} = 50\%$ , the classifier achieved an accuracy of 0.92, precision of 0.96, sensitivity of 0.88, and specificity of 0.96.

Precision was treated as the primary metric, since the objective is to filter marginal violations without waiving true violations. To evaluate robustness,  $\theta_{\text{SPICE}}$  was varied across  $1\sigma$  (68%),  $2\sigma$  (95.4%), and  $3\sigma$  (99.7%). Figures 4.15a–4.15b show that 100% precision is achieved when  $\theta_{\text{Model}} > \theta_{\text{SPICE}}$  (e.g.,  $\geq 89\%$  for  $1\sigma$  and  $\geq 90\%$  for  $2\sigma$ ). The heuristic approach proposed for 45nm, in which a latching probability corresponding to  $N\sigma$  is achieved by setting  $\theta_{\text{Model}} = (N + 1)\sigma$ , is found to be effective for the 7nm technology also.

### C2QPredict (ASAP 7 nm PDK)

The same methodology for data generation, preprocessing, model selection, and evaluation was adopted for developing the *C2QPredict* model for the 7 nm technology. The dataset was generated using Algorithm 2 and consisted of  $n = 4306$  samples with  $m = 5$  features, which were normalized using min–max scaling. After splitting the dataset into training and testing sets, several ML models with varying complexities and



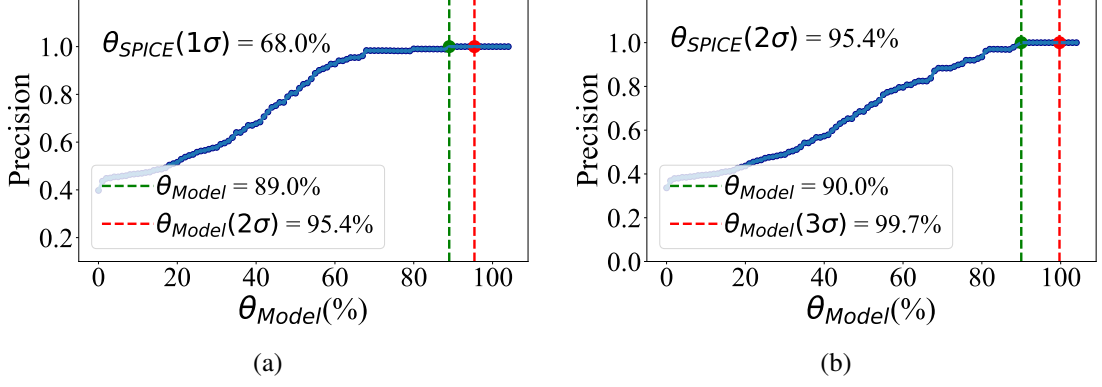


Figure 4.15: Precision vs. Threshold Plot (ASAP 7nm PDK)

data representation requirements were developed and evaluated. The training process was carefully managed to avoid overfitting. Among all evaluated models, the ANN achieved the best performance, yielding the lowest NRMSE values in both training (0.0180) and testing (0.0185). The optimized ANN hyperparameters included the number of hidden layers, ReLU activation function, Adam solver, a regularization parameter of  $\alpha = 0.01$ , and a learning rate of 0.001. Overall, these results exhibit trends consistent with those observed at 45 nm, demonstrating the robustness of the proposed approach across different technology nodes.

## 4.8 Conclusions

In this work, a novel methodology is demonstrated in which ML models are used as a classifier to distinguish between violating and non-violating flip-flops. This approach extends the usable timing space of a flip-flop beyond that provided by conventional flip-flop models. In addition, an auxiliary ML model is employed to capture the C2Q delay within the extended timing space, ensuring that this extension does not introduce timing issues for flip-flops that were already operating safely under the constraints of the conventional model. Design optimization can therefore be performed more effectively within the extended timing space, as evidenced by the resolution of marginal failures through waivers rather than through design modifications that may adversely affect other performance metrics.

Furthermore, the proposed methodology inherently accounts for local process-induced variations. Although global variations are not explicitly addressed, conventional corner-based approaches can be employed to generate augmented timing libraries, wherein a

separate augmented library is created for each corner of the conventional library. It is also noted that this chapter focuses exclusively on waiving setup/late timing violations. A similar methodology can be developed to resolve marginal hold/early timing violations and potentially to mitigate SI and IR-drop failures.

As part of future work, the framework can be extended to learn and model signal transition in addition to delay. Future work can also focus on improving the explainability of the proposed ML models to increase confidence in their predictions for previously unseen circuits. In addition, revisiting classical probabilistic timing analysis frameworks, such as [73], in conjunction with ML-based characterization may provide valuable theoretical insights for developing more robust, interpretable, and generalizable timing models.

Finally, characterizing an advanced PDK with a large number of flip-flop options incurs substantial computational overhead, primarily due to the extensive Monte Carlo simulations required. While the feasibility and benefits of such characterization are demonstrated in this work, future efforts can explore faster and more scalable alternatives to direct Monte Carlo simulations.

# CHAPTER 5

## Prediction and Alleviation of Congestion during Placement

This chapter presents an ML-driven framework for predicting and alleviating routing congestion during placement. It describes the process of feature extraction from layout data, the development of ML models, and their application in guiding congestion-aware placement. The proposed approach aims to reduce design iterations and improve overall physical design quality.

The organization of this chapter is as follows. Section 5.1 presents the background and related work. A top-level overview of the proposed framework is provided in Section 5.2. Section 5.3 details the model development methodology. Section 5.4 presents the experimental results and analysis. Finally, Section 5.5 concludes the chapter.

The work presented in this chapter has been published in [74].

### 5.1 Background and Related Work

In the preceding chapters, ML-based techniques were developed to improve the accuracy and efficiency of timing analysis by addressing pessimism/optimism in combinational and sequential timing models. These techniques were shown to reduce unnecessary design iterations during timing closure. However, timing closure represents only one dimension of the overall design closure for a modern IC. As technology nodes continue to scale, physical design challenges increasingly dominate late-stage design iterations, with routing congestion emerging as a major bottleneck that can significantly prolong closure time or, in extreme cases, lead to routing failure. Similar to pessimism in timing models, inaccurate early estimation of routing congestion during placement often results in suboptimal design decisions, necessitating repeated placement and routing iterations. Consequently, improving the accuracy of early congestion estimation is essential for accelerating physical design closure and reducing overall turnaround time.

One of the most widely adopted methods for alleviating routing congestion is im-

proving placement quality through congestion-aware placement techniques [75, 3]. In industrial placement and routing (PnR) tools, a “light-mode” global router (GR), referred to as an early global router (eGR), is used to guide the placement algorithm toward a congestion-aware solution. However, local routing resources are difficult to accurately account for in eGR, as they vary with factors such as GR layout cell size and pin density [76]. As a result, large miscorrelations are often observed between GR and eGR congestion maps. Furthermore, refining the eGR solution incurs unfavorable run-time trade-offs, and routing improvements driven by eGR during placement are often associated with limited quality [77].

In recent years, ML methods have been increasingly explored for routability improvement. For instance, routing optimization has been formulated as ML-driven regression and classification problems to predict design rule check (DRC) violations using GR and placement features [78, 79, 80, 81, 82]. The task of predicting global routing congestion from placement features has also been investigated in [83, 84, 85]. However, the impact of macros on congestion prediction accuracy is not considered in [83, 84], despite their significant influence in practical designs. In [85], robustness is evaluated using 100 diverse placements per design with varying overflow and related parameters. Nevertheless, the reliance on synthetic layouts can reduce real-world applicability and potentially limit the generalization capability of the predictor.

## 5.2 Proposed Work and Framework

In this work, a fast and accurate framework is proposed for identifying routing congestion without performing early global routing (eGR). Furthermore, a methodology is presented for modifying placement to mitigate congestion using predicted congestion information.

The problem of global routing congestion estimation is formulated as follows. The entire layout is partitioned into two-dimensional rectangular grids, where each grid cell is referred to as a window  $w^{(i,j)}$ . The indices  $i$  and  $j$  denote the window locations along the X- and Y-directions, respectively. Such a formulation enables localized characterization of routing demand and capacity, which is essential for learning-based congestion prediction.

The congestion of a window  $w^{(i,j)}$ , observed after performing global routing (GR), is denoted as  $y_w$ . It is defined as the ratio of the number of used routing tracks to the number of available routing tracks within the window  $w^{(i,j)}$ . The value  $y_w$  is treated as the golden congestion metric and is used as the label for training and testing the ML model. For each window  $w^{(i,j)}$ , a feature vector  $X_w$  is extracted from the layout, capturing the design characteristics that influence routing congestion. Using this feature representation, the ML model predicts the congestion value  $\hat{y}_w$ , thereby enabling congestion estimation without explicitly invoking eGR.

The prediction error is quantified using the root mean square error (RMSE), which is defined as

$$RMSE = \sqrt{\frac{1}{N} \sum_N (y_w - \hat{y}_w)^2},$$

where  $N$  denotes the number of samples used for error computation. The model is trained with the objective of minimizing RMSE, ensuring accurate approximation of GR-observed congestion across diverse layouts.

The proposed congestion prediction and alleviation framework is divided into two major phases, as illustrated in Fig. 5.1. In the first phase, an artificial neural network (ANN)-based model is developed by constructing a dataset, performing feature engineering, and supervised training using GR-derived congestion labels. In the second phase, the trained model is integrated into the placement flow to predict congestion and guide placement modifications to alleviate it early in the design process.

All experiments presented in this chapter are conducted using the ISPD 2014 and ISPD 2015 benchmark suites, as listed in Table 5.1, implemented in 65 nm technology [87, 88]. An industrial placement and routing (PnR) tool, Innovus, is employed to perform placement and routing in all experiments [89], ensuring that the proposed framework is evaluated under realistic design conditions.

## 5.3 Model Development

Model development consists of three primary tasks: dataset creation, model training, and model testing. Datasets are constructed by collecting relevant information from the design, including GR congestion values, which serve as labels for supervised learning.

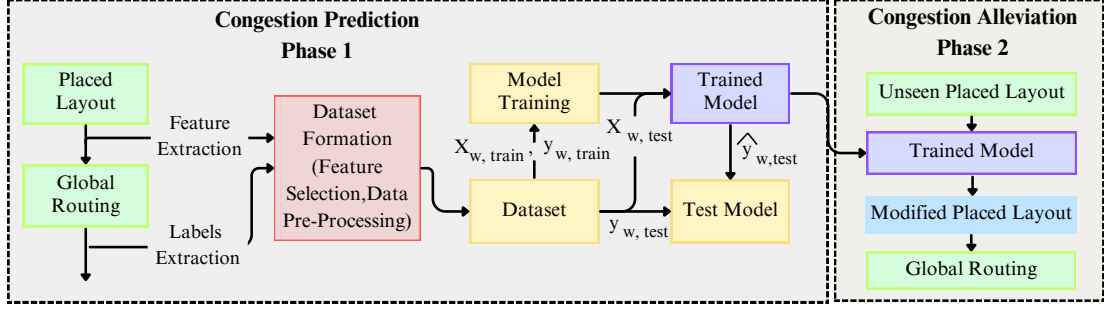


Figure 5.1: Proposed Methodology: Top level view of Congestion Prediction and Alleviation

Table 5.1: Attributes of the ISPD14 and ISPD15 Designs used in this work

| ISPD14             |       |         |       |        | ISPD15             |       |         |       |        |
|--------------------|-------|---------|-------|--------|--------------------|-------|---------|-------|--------|
| Design             | Inst. | Windows | Util. | Macros | Design             | Inst. | Windows | Util. | Macros |
| mgc_des_perf_1     | 112K  | 21K     | 0.9   | 0      | mgc_des_perf_a     | 108K  | 90K     | 0.71  | 4      |
| mgc_des_perf_2     | 112K  | 23K     | 0.85  | 0      | mgc_des_perf_b     | 112K  | 40K     | 0.49  | 0      |
| mgc_edit_dist_1    | 130K  | 58K     | 0.4   | 0      | mgc_fft_a          | 30K   | 71K     | 0.74  | 6      |
| mgc_edit_dist_2    | 130K  | 53K     | 0.43  | 0      | mgc_fft_b          | 30K   | 71K     | 0.74  | 6      |
| mgc_fft            | 32K   | 7.7K    | 0.83  | 0      | mgc_matrix_mult_a  | 149K  | 250K    | 0.76  | 5      |
| mgc_matrix_mult    | 155K  | 33K     | 0.8   | 0      | mgc_matrix_mult_b  | 146K  | 250K    | 0.72  | 7      |
| mgc_pci_bridge32_1 | 30K   | 67K     | 0.84  | 0      | mgc_pci_bridge32_a | 29K   | 17K     | 0.40  | 4      |
| mgc_pci_bridge32_2 | 30K   | 6.5K    | 0.85  | 0      | mgc_pci_bridge32_b | 28K   | 71K     | 0.50  | 6      |

Table 5.2: Summary of Design Features that Impact Global Routing Congestion

| S.No. | Name and Symbol                                     | Definition and Explanation                                                                        |
|-------|-----------------------------------------------------|---------------------------------------------------------------------------------------------------|
| 1     | Macro Region ( $M_w$ )                              | Represents presence of a macro in $w^{(i,j)}$ . $M_w = 1$ if a macro is present, else $M_w = 0$ . |
| 2     | Number of Macros ( $N_{\text{macros}}$ )            | Total number of macros in the design.                                                             |
| 3     | Max Input Pins ( $MI_{\text{pin},w}$ )              | Count of input pins of the instance with the maximum number of input pins in $w^{(i,j)}$ .        |
| 4     | Max Output Pins ( $MO_{\text{pin},w}$ )             | Count of output pins of the instance with the maximum number of output pins in $w^{(i,j)}$ .      |
| 5     | Number of Instances ( $N_{\text{inst},w}$ )         | Total number of instances within $w^{(i,j)}$ .                                                    |
| 6     | Number of Pins ( $N_{\text{pin},w}$ )               | Total number of pins within $w^{(i,j)}$ of all instances.                                         |
| 7     | Total Instance Area ( $I_{\text{area},w}$ )         | Sum of areas of all instances in $w^{(i,j)}$ .                                                    |
| 8     | No. of Metal Layers ( $M_{\text{lay},w}$ )          | Maximum number of metal layers used by instances in $w^{(i,j)}$ .                                 |
| 9     | $\rho_{\text{place},w}$ ( $\rho_{\text{place},w}$ ) | Ratio of $I_{\text{area},w}$ to free space in $w^{(i,j)}$ .                                       |
| 10    | Pin Density ( $\rho_{\text{pin},w}$ )               | Ratio of $N_{\text{pin},w}$ to $I_{\text{area},w}$ in $w^{(i,j)}$ .                               |
| 11    | No. of Special Nets ( $NS_{\text{net},w}$ )         | Number of Vdd and Gnd nets within $w^{(i,j)}$ .                                                   |
| 12    | Neighbourhood ( $N_{\text{bh},w}$ )                 | Congestion tendency based on congestion in neighboring $w^{(i,j)}$ .                              |

Feature selection is a critical aspect of dataset creation, as the model's effectiveness strongly depends on the chosen design attributes. In this work, the design features listed in Table 5.2 are considered for model development. These features are identified

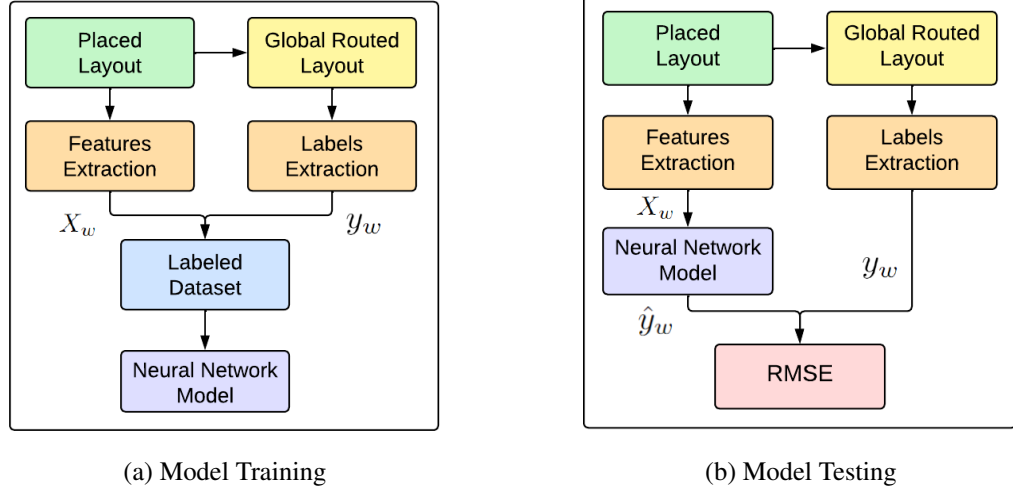


Figure 5.2: Model development framework

through extensive experimentation to determine their impact on routing congestion. To the best of our knowledge, the features number of macros ( $N_{\text{macros}}$ ), maximum input pins ( $MI_{\text{pin},w}$ ), number of metal layers ( $M_{\text{lay},w}$ ), and number of special nets ( $NS_{\text{net},w}$ ) are employed for global routing congestion prediction for the first time in this work.

After raw data are collected from multiple designs, several preprocessing steps are performed, including format transformation, inconsistency removal, and feature scaling. Multicollinearity analysis is carried out to identify highly correlated features that may adversely affect model interpretability and performance. Feature selection is then applied to retain the most informative attributes, thereby reducing model complexity and improving prediction accuracy. The model is subsequently trained and evaluated using the selected features. If the achieved performance is found to be inadequate, further feature exploration and refinement are conducted. This iterative process continues until satisfactory performance is achieved.

Finally, the constructed dataset is partitioned into training and testing subsets. The overall training and testing flow for model development is illustrated in Fig. 5.2. Multiple ML algorithms are explored using a grid-search-based optimization strategy, as described in Sec. 5.3.3. The trained model is evaluated on previously unseen designs, and the RMSE is computed to assess prediction quality. The trained model is integrated into the placement flow to reduce routing congestion.

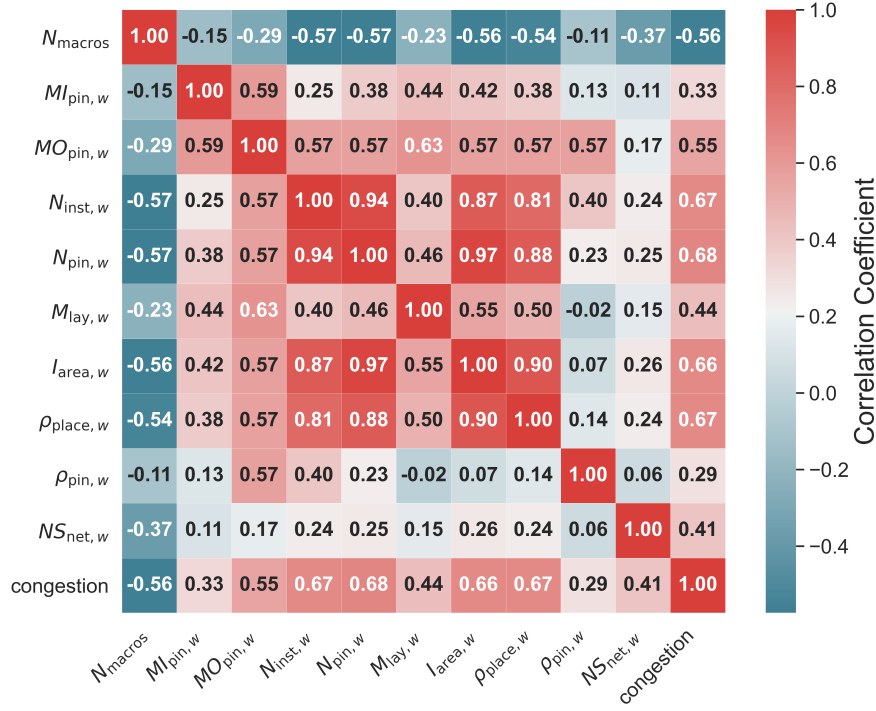


Figure 5.3: Correlation between features and target variable

### 5.3.1 Feature Selection

Using the Spearman correlation matrix, the relationships between the features and the target variable are examined [90]. The Spearman correlation coefficient, which ranges from  $-1$  to  $1$ , is selected since it is capable of evaluating non-linear relationships. However, for windows consisting entirely of macros, the feature values remain constant and the corresponding target variable is zero. As a result, these windows do not contribute meaningful information to the correlation analysis and can introduce artificially strong correlations with the target variable. Therefore, data points with zero target values are removed during the correlation analysis.

From Fig. 5.3, it can be observed that each feature exhibits some degree of correlation with the target variable. In addition, strong correlations among the features themselves are observed, indicating multicollinearity. Multicollinearity suggests that certain variables provide redundant or highly similar information. For instance, the Spearman correlation matrix reveals a significant correlation of  $0.97$  between the pin count ( $N_{\text{pin},w}$ ) and the instance area ( $I_{\text{area},w}$ ) within a window. Multicollinearity is generally assumed to occur when the correlation coefficient approaches  $0.80$ . Despite correlations among features, they may still provide independent information useful for model training [91]. Consequently, a Variance Inflation Factor (VIF) analysis is conducted



Table 5.3: Results of VIF analysis

| Feature                 | Feature Set 1 | Feature Set 2  | Feature Set 3  |
|-------------------------|---------------|----------------|----------------|
| $N_{\text{macros}}$     | 2.18          | 2.05           | 1.99           |
| $MI_{\text{pin},w}$     | 7.45          | 5.37           | 4.83           |
| $MO_{\text{pin},w}$     | 269.07        | <i>Removed</i> | <i>Removed</i> |
| $N_{\text{inst},w}$     | 94.17         | <i>Removed</i> | <i>Removed</i> |
| $N_{\text{pin},w}$      | 173.89        | <i>Removed</i> | <i>Removed</i> |
| $M_{\text{lay},w}$      | 14.3          | 8.07           | 7.46           |
| $I_{\text{area},w}$     | 98.14         | <i>Removed</i> | <i>Removed</i> |
| $\rho_{\text{place},w}$ | 27.33         | 6.57           | 5.9            |
| $\rho_{\text{pin},w}$   | 189.83        | 9.98           | <i>Removed</i> |
| $NS_{\text{net},w}$     | 5.78          | 5.72           | 5.16           |

to comprehensively investigate feature correlations. The VIF quantifies the extent to which the variance (or standard error) of an estimated regression coefficient is inflated due to collinearity. In general, VIF values greater than 10 are considered indicative of problematic multicollinearity [92].

First, a VIF analysis is conducted using all the aforementioned features, and the results are reported in Table 5.3 as Feature Set 1. Several features, including  $MO_{\text{pin},w}$ ,  $N_{\text{inst},w}$ ,  $N_{\text{pin},w}$ ,  $M_{\text{lay},w}$ ,  $I_{\text{area},w}$ ,  $\rho_{\text{place},w}$ , and  $\rho_{\text{pin},w}$ , are observed to exhibit high VIF coefficients, indicating strong correlations among them.

To address this issue, an iterative feature elimination approach is adopted, in which the feature with the highest VIF coefficient is removed in each iteration and the VIF values are recomputed. This process is repeated until the VIF coefficients of all remaining features are reduced to values close to 10. As a result, a reduced feature set with lower collinearity is obtained, as shown in Table 5.3 as Feature Set 2.

In addition, a further experiment is performed by eliminating one more feature,  $\rho_{\text{pin},w}$ , and the resulting feature configuration is reported as Feature Set 3 in Table 5.3.

After the feature sets described above are created, the models are trained and tested separately using each feature set, and the corresponding  $RMSE$  values are reported in Fig. 5.4. The designs highlighted in gray in Table 5.1 are used for training, while the remaining designs are used for testing. All features are normalized using a min-max scaler to ensure fairness across the feature set.

It is observed that, despite containing a larger number of features, Feature Set 1 yields a higher  $RMSE$  than Feature Set 2. This behavior is attributed to multicollinear-

ity among the features in Feature Set 1, which adversely affects the model’s predictive accuracy. Furthermore, Feature Set 3, obtained by removing additional features and further reducing VIF, produces slightly inferior results compared to Feature Set 2. This degradation in performance is attributed to the loss of important information as more features are removed, leading to a less comprehensive representation of the relationships between the features and the target variable.

Therefore, Feature Set 2, which exhibits reduced multi-collinearity while retaining relevant information and achieving a lower  $RMSE$ , is selected for use in all subsequent experiments.

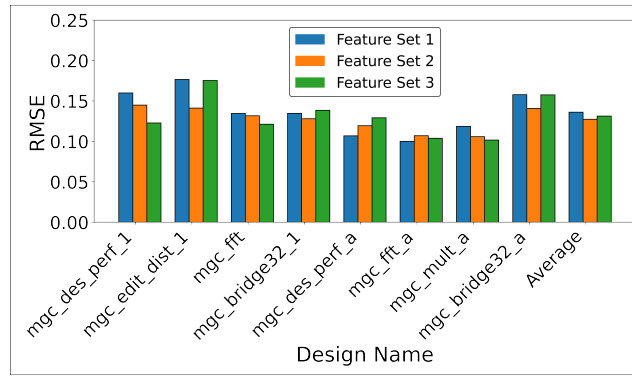


Figure 5.4:  $RMSE$  obtained for different feature sets

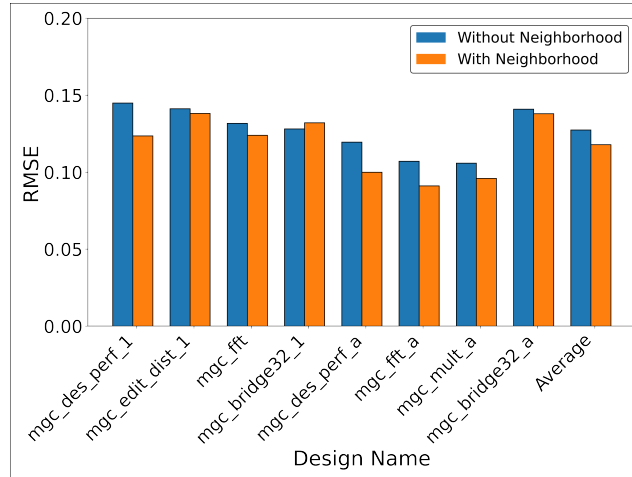


Figure 5.5: Impact of neighborhood on  $RMSE$

### 5.3.2 Impact of neighbourhood

Routing congestion is typically observed in clustered regions on a layout. Hence, intuitively, the probability of congestion in a given window is increased when congestion is present in neighboring windows. To account for this effect, features from neighbor-

Table 5.4: Comparison of ML Models and Attributes of ANN

| ML Models Comparison |      | ANN Attributes      |               |
|----------------------|------|---------------------|---------------|
| ML Models            | RMSE | Attribute           | Specification |
| Linear               | 0.12 | Hidden Layers       | 3             |
| Lasso                | 0.13 | Neurons per Layer   | 100           |
| Decision Tree        | 0.12 | Activation Function | ReLU          |
| ANN                  | 0.10 | Regularization      | 0.1           |

ing windows are also included when predicting the congestion of a given window. The “neighboring windows,” are those windows that share at least one common corner with the given window.

The impact of incorporating neighborhood features on the  $RMSE$  is demonstrated in Fig. 5.5. The results indicate that including neighborhood features enhances the predictive capability of the congestion model.

### 5.3.3 Model Selection and Fine Tuning

The designs highlighted in gray in Table 5.1 are used for training and testing the ML models. Subsequently, the dataset is divided into a 70:30 split. Four types of ML models, namely Linear Regression, Lasso, Decision Tree, and ANN, are trained and tested on the respective splits, and the results are presented in Table 5.4. It can be observed that the ANN model outperforms the other three models.

An ANN model is composed of multiple interconnected layers of neurons, which enable nonlinear relationships to be learned and represented more accurately. The flexibility of the ANN model allows intricate patterns in the dataset to be captured, thereby improving performance compared to the other models.

The hyper-parameters of the ANN model are tuned to enhance its effectiveness. A grid search is performed to explore different configurations of the ANN model, including variations in the number of hidden layers, the number of neurons, the activation function, the solver, and the regularization parameters. This exhaustive search for optimal hyper-parameters results in the final ANN model attributes shown in Table 5.4. The results obtained using this model are presented in the next section.

## 5.4 Results

First, the results of congestion prediction are presented, followed by the results of congestion alleviation.

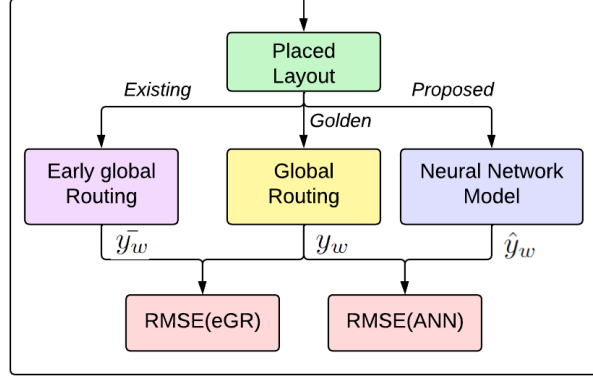


Figure 5.6: Framework for benchmarking prediction accuracy

### 5.4.1 Congestion Prediction

To fairly assess the effectiveness of the ANN model for routing congestion prediction, unseen designs are used for evaluation. The designs highlighted in gray in Table 5.1 are used exclusively for training, while the remaining designs are reserved for testing. The experimental setup shown in Fig. 5.6 is employed to evaluate the effectiveness of the proposed ANN model. For each window on the layout, three congestion values are obtained as follows:

- Golden congestion values ( $y_w$ ): obtained by performing GR using the industrial PnR tool and computing the actual congestion.
- eGR-predicted congestion values ( $\bar{y}_w$ ): obtained using the industrial PnR tool on the placed (but not yet routed) layout, where the tool internally performs eGR.
- ANN-predicted congestion values ( $\hat{y}_w$ ): obtained using the proposed ANN model with features extracted from the placed (but not yet routed) layout.

Prior to quantitative analysis, the congestion maps generated by the three methods are compared visually in Fig. 5.7 and 5.8 for two representative designs: *mgc\_pci\_bridge32\_1* (without macros) and *mgc\_ft\_a* (with macros).

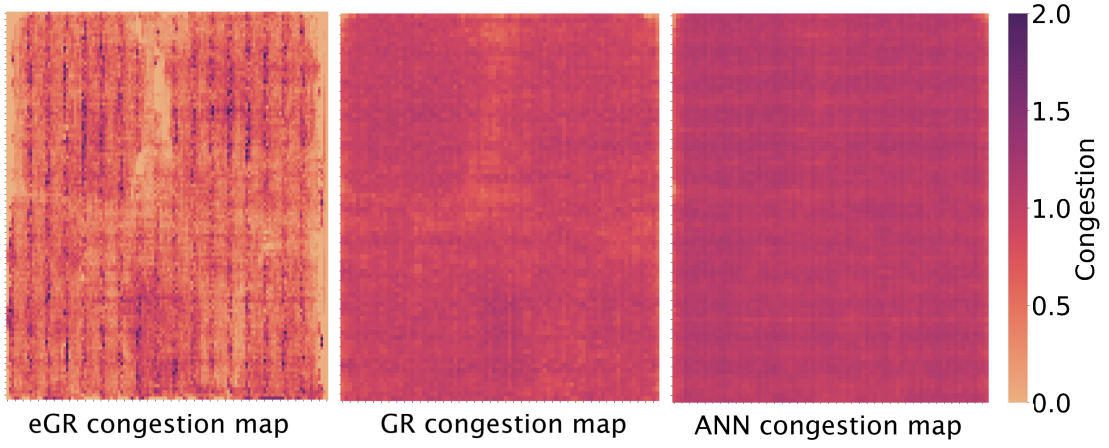


Figure 5.7: Congestion maps for *mgc\_pci\_bridge32\_1*

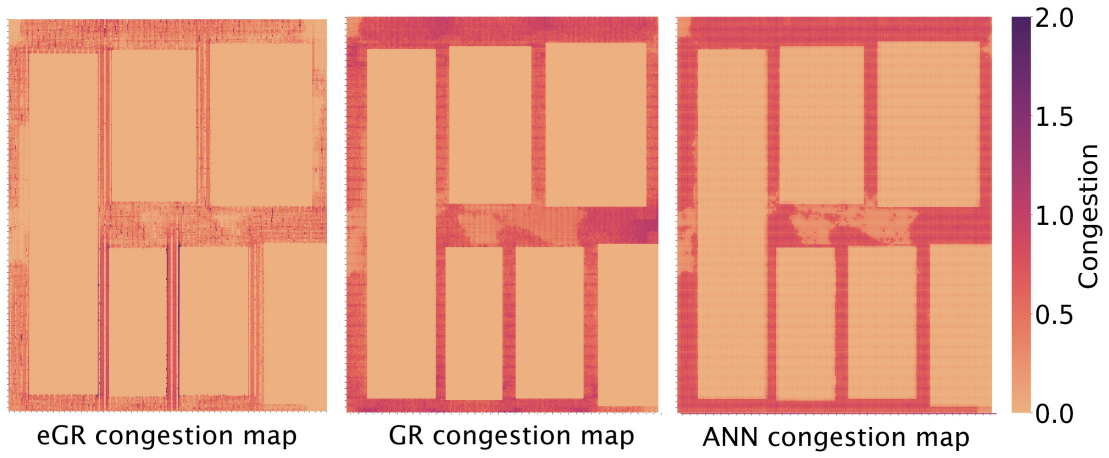


Figure 5.8: Congestion maps for *mgc\_fft\_a*

It can be observed that the congestion maps produced by the ANN closely resemble the golden congestion maps. To quantify prediction accuracy, the following RMSE values are computed and reported in Table 5.5:

$$RMSE(eGR) = \sqrt{\frac{1}{N} \sum_N (y_w - \bar{y}_w)^2},$$

$$RMSE(ANN) = \sqrt{\frac{1}{N} \sum_N (y_w - \hat{y}_w)^2},$$

where  $N$  denotes the total number of samples. The metric  $RMSE(eGR)$  captures the error between the golden congestion and the eGR-predicted values, while  $RMSE(ANN)$  measures the error between the golden congestion and the ANN-predicted values. It is observed that  $RMSE(ANN)$  is, on average, 68% lower than  $RMSE(eGR)$ . These results demonstrate that the proposed ANN-based model predicts congestion more accurately than the existing eGR-based approach.

Table 5.5: Results of Predicting Congestion ( $Nw \equiv$  Number of Windows)

| Design          | $Nw$       | RMSE (eGR)    | RMSE (ANN)    | Improvement |
|-----------------|------------|---------------|---------------|-------------|
| mgc_des_perf_1  | 22K        | 0.3210        | 0.0989        | 69%         |
| mgc_edit_dist_1 | 58K        | 0.2986        | 0.1053        | 64%         |
| mgc_fft         | 8K         | 0.3192        | 0.0986        | 69%         |
| mgc_bridge32_1  | 7K         | 0.3953        | 0.1060        | 73%         |
| mgc_des_perf_a  | 90K        | 0.1939        | 0.0734        | 62%         |
| mgc_fft_a       | 71K        | 0.2345        | 0.0768        | 67%         |
| mgc_mult_a      | 251K       | 0.2058        | 0.0643        | 68%         |
| mgc_bridge32_a  | 18K        | 0.3122        | 0.1014        | 68%         |
| <b>Average</b>  | <b>65K</b> | <b>0.2850</b> | <b>0.0905</b> | <b>68%</b>  |

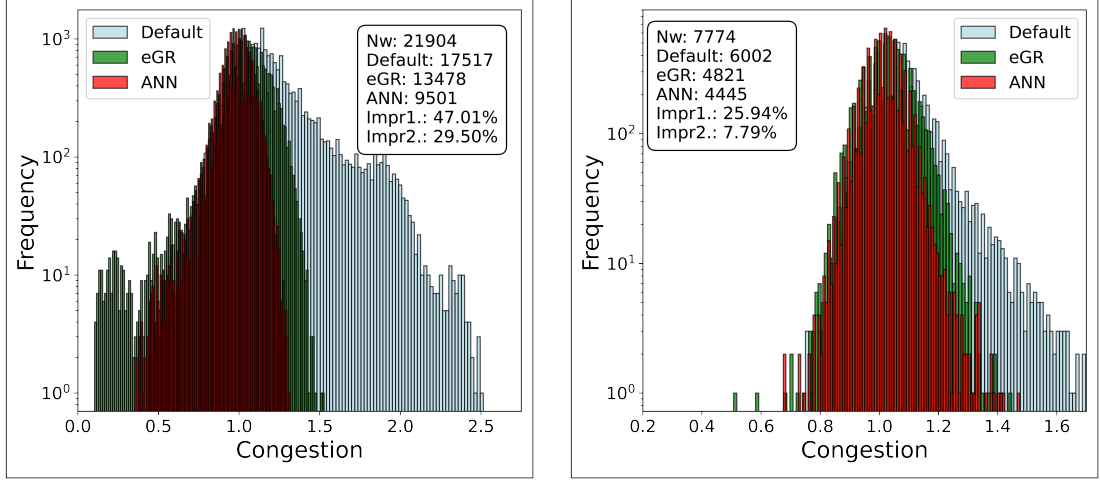
Furthermore, congestion prediction using the ANN requires fewer arithmetic operations, resulting in faster runtime compared to eGR. Although ANN training incurs additional computational cost, it is a one-time process. Consequently, the proposed model is well suited for integration into the placement flow to alleviate congestion, which is examined in the following section.

### 5.4.2 Congestion Alleviation

The efficacy of congestion mitigation techniques during placement is influenced by the following factors:

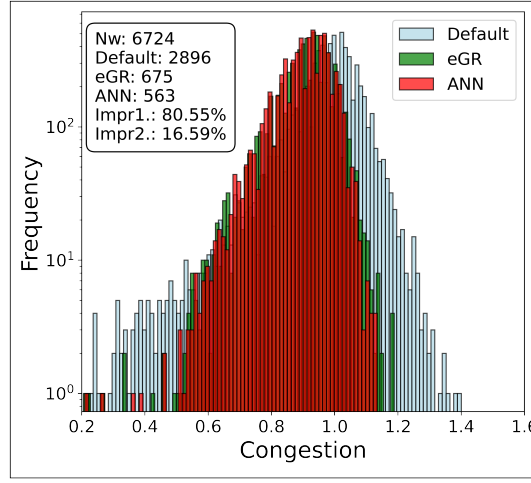
- Congestion prediction: accurate prediction enables optimization to be applied in appropriate layout regions.
- Incremental placement: after the identification of regions requiring congestion reduction, effective algorithms can efficiently refine placement solutions to mitigate congestion.
- Design attributes: congestion alleviation techniques can reduce congestion only if physical constraints permit routing resource redistribution.

In this work, the focus is placed on demonstrating how the improved predictability of the ANN model can be employed for congestion mitigation. It is worth pointing out that the objective of this work is not to propose an improved incremental placer. Therefore, as a demonstration of congestion alleviation, the incremental placement capability of the commercial PnR tool is directly employed to refine placement using the approach described below.



(a) mgc\_des\_perf\_1

(b) mgc\_fft



(c) mgc\_bridge32\_1

Impr1. = Improvement (%) of ANN over Default, Impr2. = Improvement (%) of ANN over eGR,  $C_T = 1$ ,  $RF = 0.95$

Figure 5.9: Results obtained using three congestion alleviation techniques

A window is identified as potentially congested if the predicted congestion exceeds a user-defined threshold  $C_T$ . Subsequently, the targeted placement density ( $\rho_{T,W}$ ) for a potentially congested window is specified as

$$\rho_{T,W} = \rho_{C,W} \times RF,$$

where  $\rho_{C,W}$  denotes the current placement density and  $RF$  is a user-defined reduction factor ( $RF < 1$  for congestion reduction). Thereafter, the incremental placer of the commercial PnR tool is employed to move cells out of potentially congested windows such that the targeted  $\rho_{T,W}$  can be achieved and congestion can be reduced.

To illustrate the benefits of the improved predictability of the ANN model on con-

gestion alleviation, the following three experiments are conducted:

1. Default: congestion is predicted using eGR by the commercial PnR tool, and the default congestion reduction techniques available in the tool are applied.
2. eGR: potentially congested windows are predicted using eGR, after which placement density is modified using the incremental placer as described above.
3. ANN: potentially congested windows are predicted using the proposed ANN model, after which placement density is modified using the incremental placer as described above (identical to the eGR experiment).

The results of the above experiments are reported in Fig. 5.9 for designs exhibiting significant congestion issues (i.e., congestion present in more than 5% of the windows). In Fig. 5.9(a), the design `mgc_des_perf_1` contains a total of 21,904 windows, of which 17,517 exhibit congestion under the default flow, 13,478 under the eGR-based flow, and 9,501 under the ANN-based flow. This corresponds to an improvement of 47.01% of ANN over the default flow and 29.50% over the eGR-based flow. In Fig. 5.9(b), the design `mgc_fft` has 7,774 windows, with 6,002 congested under the default flow, 4,821 under the eGR-based flow, and 4,445 under the ANN-based flow. The ANN-based method achieves a 25.94% reduction compared to the default flow and a 7.79% reduction compared to the eGR-based flow. In Fig. 5.9(c), the design `mgc_bridge32_1` includes 6,724 windows, among which 2,896 are congested under the default flow, 675 under the eGR-based flow, and 563 under the ANN-based flow. This results in an 80.55% improvement over the default flow and a 16.59% improvement over the eGR-based flow. Overall, the ANN-based reduction technique significantly decreases the number of windows exhibiting post-global-routing congestion compared to the default flow (by 26–81%). This improvement is attributed to more accurate congestion prediction and the additional optimization enabled by the incremental placer. Furthermore, the ANN-based approach also reduces congestion compared to the eGR-based flow (by 8–30%). Since the incremental placer-based optimization remains identical in both flows, the observed gains are solely due to the higher accuracy of the ANN-based prediction, which enables more realistic identification of optimization targets with respect to global routing congestion.

Moreover, Fig. 5.9 shows a general leftward shift of the histogram for the ANN-



based reduction technique, indicating overall congestion alleviation. A reduced area under the curve is also observed for the ANN-based technique, which can be attributed to more equitable utilization of routing resources across the layout. These results demonstrate the effectiveness of the proposed ANN-based congestion prediction approach and its applicability for congestion reduction during placement.

## **5.5 Conclusion**

In this work, an ANN-based framework is proposed for more accurate prediction of global routing congestion and leveraging this prediction to mitigate congestion. In addition, the results of the proposed technique are presented on benchmark designs and are compared with existing methods. The reasonably good results obtained using the ANN-based prediction model highlight the effectiveness of the proposed approach.

In future work, advanced neural network architectures, such as convolutional neural networks (CNNs), along with enhanced feature engineering-particularly features related to interconnects-can be explored to further improve the results. Additionally, the ANN-based prediction model can be more tightly coupled with the incremental placer, and the incremental placement algorithm can be further refined to fully exploit the improved prediction capability, not only for congestion alleviation but also for enhancing other design figures of merit.

# CHAPTER 6

## Conclusion and Future Work

### 6.1 Conclusion

This thesis has explored the use of ML to address accuracy, scalability, and runtime challenges across multiple stages of the EDA flow. With continued technology scaling, increasing circuit complexity and the growing pessimism of conventional models necessitate the development of fast, accurate, and adaptive modeling techniques. The work presented in this thesis demonstrates that data-driven approaches can effectively complement traditional EDA methodologies, enabling improved design insight and optimization without incurring prohibitive computational cost.

ML has been leveraged to efficiently model MIS effects in logic gates, glitch behavior in combinational circuits, timing behavior of flip-flops near marginal operating regions, and global routing congestion during physical design. Across these diverse problem domains, ML-driven models were shown to significantly reduce reliance on exhaustive SPICE simulations or expensive analysis steps, while maintaining high prediction accuracy and offering substantial runtime speedups.

The proposed frameworks demonstrate that intelligent dataset generation, feature engineering, and carefully designed learning architectures are key enablers for practical adoption. By selectively focusing on the most critical operating regions, such as MIS-induced speed-up effects, glitch-prone transitions, marginal flip-flop timing windows, and congestion-sensitive layout regions, the presented approaches achieve a favorable balance between accuracy and efficiency. Furthermore, the results consistently show that ML-driven predictions can be directly leveraged to guide design optimization, whether by improving timing analysis fidelity, mitigating glitches, resolving marginal timing violations, or alleviating routing congestion during placement.

Another important contribution of this thesis is the demonstration of how ML models can be integrated into existing design flows with minimal disruption. This thesis adopts a hybrid design philosophy in which ML complements rather than replaces con-

ventional symbolic EDA algorithms. Classical EDA techniques continue to provide the underlying optimization and verification framework, while ML models are employed to accelerate computationally intensive tasks and improve prediction accuracy. Such a complementary approach combines the robustness and interpretability of conventional algorithms with the efficiency of data-driven models, making it more suitable for practical industrial deployment.

Despite these encouraging results, several limitations of the present work should be acknowledged. The predictive performance of the proposed ML models depends on the quality, diversity, and representativeness of the training datasets. Although the datasets were carefully generated for the targeted applications, their coverage of rare operating conditions and previously unseen design scenarios may affect model generalization. In addition, the experimental validation presented in this thesis has primarily been performed using a 7nm & 45 nm technology node. Consequently, additional validation across different technology nodes, standard-cell libraries, and foundries is required before the proposed methodologies can be considered fully technology independent.

In summary, this thesis establishes that ML provides a powerful and flexible framework for addressing several long-standing challenges in modern EDA flow. By improving modeling accuracy, reducing computational overhead, and enabling earlier and more effective optimization, the techniques presented in this work contribute toward more reliable, efficient, and scalable digital design flows, and lay a strong foundation for future research and industrial adoption of ML-driven EDA methodologies.

## **6.2 Future Work**

Several opportunities exist to extend the research presented in this thesis. One promising direction is the investigation of more advanced machine learning architectures, including convolutional neural networks (CNNs), graph neural networks (GNNs), transformer-based models, and physics-informed neural networks (PINNs), which are better suited to capturing the spatial, topological, and physical characteristics of integrated circuit designs. These architectures have the potential to further improve prediction accuracy and generalization across diverse EDA applications.

Another important direction is the evaluation of the proposed methodologies across

advanced technology nodes, such as 5 nm, 3 nm, and beyond, together with different standard-cell libraries, semiconductor foundries, and large-scale industrial benchmark circuits. Expanding the training datasets to include a wider range of operating conditions, process variations, and rare corner cases would further enhance model generalization, robustness, and portability across different design environments.

The proposed frameworks can also be extended to address additional optimization objectives beyond those considered in this thesis. Future research may investigate joint optimization of timing, routing congestion, power consumption, thermal behaviour, manufacturability, signal integrity, power integrity, and reliability, thereby enabling comprehensive multi-objective optimization within the EDA flow. Furthermore, incorporating the effects of voltage and temperature variations, aging, manufacturing variability, IR drop, and crosstalk into the learning process would improve the robustness of the proposed models under realistic operating conditions.

Another promising research direction is the tighter integration of the proposed ML models with commercial EDA tools for timing analysis, placement, routing, and signal integrity verification. Such integration would enable ML predictions to be utilized more effectively during design optimization while allowing a quantitative assessment of their impact on complete chip implementation flows. In particular, future work should evaluate how the proposed characterization methodologies influence downstream EDA tasks, including full-chip static timing analysis, timing closure, placement quality, routing convergence, and overall design performance.

Similarly, the runtime overhead associated with ML models training, retraining, hyperparameter optimization and inference during complete chip implementation should be systematically analysed to quantify the trade-off between improved prediction accuracy and additional computational cost. Such end-to-end studies would provide designers with a clearer understanding of the practical benefits, deployment costs, and industrial applicability of ML-assisted EDA methodologies.

Finally, future work can investigate model interpretability, prediction uncertainty, and failure analysis to improve confidence in ML-assisted design automation. Incorporating explainable ML techniques and systematically analysing prediction uncertainty, failure cases, boundary conditions, and worst-case scenarios would increase the transparency, reliability, and trustworthiness of the proposed methodologies, thereby facili-

tating their adoption in industrial EDA environments.

## Bibliography

- [1] P. Beniwal, S. Saurabh, N. V. Chowdary, S. Skariah, A. Mandal, and R. Venkattraman, “LiMo: A Framework Leveraging Machine Learning for Multi-Input Switching Timing Models of Complex Logic Gates,” *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 72, no. 12, pp. 7979–7992, Dec. 2025. doi: 10.1109/TCSI.2025.3579073.
- [2] Synopsys, “Liberty library modeling,” Jan. 29, 2025. [Online]. Available: <https://www.synopsys.com/community/interoperability-programs/tap-in.html>
- [3] S. Saurabh, *Introduction to VLSI Design Flow*. Cambridge, U.K.: Cambridge Univ. Press, 2023.
- [4] C. Lutkemeyer, “A practical model to reduce margin pessimism for multi-input switching in static timing analysis of digital CMOS circuits,” in *Proc. TAU Workshop*, 2015, pp. 10–20.
- [5] D. Sinha, V. Rao, C. Peddawad, M. Wood, J. Hemmett, S. Skariah, and P. Williams, “Statistical timing analysis considering multiple-input switching,” in *Proc. 57th ACM/IEEE Design Automation Conf. (DAC)*, San Francisco, CA, USA, 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218601.
- [6] Cadence Design Systems, “Tempus timing signoff solution,” Jan. 29, 2025. [Online]. Available: [https://www.cadence.com/en\\_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html)
- [7] R. Tayade, S. Nassif, and J. Abraham, “Analytical model for the impact of multiple input switching noise on timing,” in *Proc. Asia South Pacific Design Automation Conf. (ASP-DAC)*, 2008, pp. 514–517, doi: 10.1109/ASPDAC.2008.4484005.
- [8] Y. Jun, K. Jun, and S. Park, “An accurate and efficient delay time modeling for MOS logic circuits using polynomial approximation,” *IEEE Trans. Comput.-*

- Aided Design Integr. Circuits Syst.*, vol. 8, no. 9, pp. 1027–1032, Sep. 1989, doi: 10.1109/43.35557.
- [9] T. Fukuoka, A. Tsuchiya, and H. Onodera, “Statistical gate delay model for multiple input switching,” in *Proc. ASP-DAC*, 2008, pp. 286–291, doi: 10.1109/ASP-DAC.2008.4483959.
  - [10] J. Shin, J. Kim, N. Jang, E. Park, and Y. Choi, “A gate delay model considering temporal proximity of multiple input switching,” in *Proc. Int. SoC Design Conf. (ISOCDC)*, Nov. 2009, pp. 577–580, doi: 10.1109/SOCDC.2009.5423815.
  - [11] A. R. Subramaniam, J. Roveda, and Y. Cao, “A finite-point method for efficient gate characterization under multiple input switching,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 1, pp. 1–25, Dec. 2015, doi: 10.1145/2778970.
  - [12] A. Ferdowsi, J. Maier, D. Öhlinger, and U. Schmid, “A simple hybrid model for accurate delay modeling of a multi-input gate,” in *Proc. Design, Automation and Test in Europe (DATE)*, Antwerp, Belgium, 2022, pp. 1461–1466, doi: 10.23919/DAT54114.2022.9774547.
  - [13] 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 Anal.: Hybrid Syst.*, vol. 56, 2025, doi: 10.1016/j.nahs.2024.101572.
  - [14] W. Raslan and Y. Ismail, “Deep-learning cell-delay modeling for static timing analysis,” *Ain Shams Eng. J.*, vol. 14, no. 1, 2023, doi: 10.1016/j.asej.2022.101828.
  - [15] F. Klemme, Y. Chauhan, J. Henkel, and H. Amrouch, “Cell library characterization using machine learning for design technology co-optimization,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2020, pp. 1–9, doi: 10.1145/3400302.3415713.
  - [16] 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,” in *Proc. ICCAD*, San Diego, CA, USA, 2020, pp. 1–9.

- [17] P. d'Hondt, A. Ladhar, P. Girard, and A. Virazel, "A learning-based methodology for accelerating cell-aware model generation," in *Proc. DATE*, Grenoble, France, 2021, pp. 1580–1585, doi: 10.23919/DATEDATE51398.2021.9474227.
- [18] E. Naswali, N. Kim, and P. Chandran, "Fast and accurate library generation leveraging deep learning for OCV modelling," in *Proc. IEEE Int. Symp. Quality Electron. Design (ISQED)*, 2021, pp. 349–354, doi: 10.1109/ISQED51717.2021.9424316.
- [19] M. Agarwal and S. Saurabh, "An efficient timing model of flip-flops based on artificial neural network," in *Proc. ACM/IEEE 3rd Workshop Mach. Learn. CAD (MLCAD)*, Raleigh, NC, USA, 2021, pp. 1–6, doi: 10.1109/MLCAD52597.2021.9531284.
- [20] O. V. S. Shashank Ram and S. Saurabh, "Modeling multiple-input switching in timing analysis using machine learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 723–734, Apr. 2021, doi: 10.1109/T-CAD.2020.3009624.
- [21] H. Jiang, X. Cheng, and P. Cao, "Multiple-input switching modeling with graph neural network," in *Proc. Int. Symp. Electron. Design Autom. (ISED)*, Nanjing, China, 2023, pp. 428–432, doi: 10.1109/ISED59274.2023.10218616.
- [22] S. Vollala, "A graph-based approach for computation reduction multi-input switching," in *Proc. Int. Workshop Timing Issues Specif. Synthesis Digit. Syst.*, 2019.
- [23] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Theory and Applications of Satisfiability Testing*, LNCS, vol. 2919 (Berlin, Germany: Springer), 2003, pp. 502–518, doi: 10.1007/978-3-540-24605-3\_37.
- [24] Cadence Design Systems, "Virtuoso ADE Suite," Jan. 29, 2025. [Online]. Available: [https://www.cadence.com/en\\_US/home/tools/custom-ic-analog-rf-design/circuit-design/virtuoso-ade-suite.html](https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-design/virtuoso-ade-suite.html)
- [25] Scikit-learn, "Cross-validation: Evaluating estimators," [Online]. Avail-



able: [https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

- [26] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Automation of Reasoning*, J. H. Siekmann and G. Wrightson, Eds. Berlin, Germany: Springer, 1983, pp. 466–483, doi: 10.1007/978-3-642-81955-1\_28.
- [27] L. T. Clark *et al.*, "ASAP7: A 7-nm FinFET predictive process design kit," *Microelectron. J.*, vol. 53, pp. 105–115, 2016, doi: 10.1016/j.mejo.2016.04.006.
- [28] ISCAS Circuits, "ISCAS benchmark circuits." [Online]. Available: <http://www.pld.ttu.ee/maksim/benchmarks/>
- [29] P. Beniwal, A. Mandal, R. Venkatraman, S. Skariah, and S. Saurabh, "Modeling Glitches due to Multiple Input Switching using Machine Learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Mar. 2026. doi: 10.1145/3802920.
- [30] Synopsys, "Liberty library modeling," Jan. 29, 2025. [Online]. Available: <https://www.synopsys.com/community/interoperability-programs/tap-in.html>
- [31] V. Chandramouli and K. A. Sakallah, "Modeling the effects of temporal proximity of input transitions on gate propagation delay and transition time," in *Proc. 33rd Design Automation Conf. (DAC)*, 1996, pp. 617–622, doi: 10.1109/DAC.1996.545649.
- [32] A. Agarwal, F. Dartu, and D. Blaauw, "Statistical gate delay model considering multiple input switching," in *Proc. 41st Design Automation Conf. (DAC)*, 2004, pp. 658–663, doi: 10.1145/996566.996746.
- [33] S. H. Choi, K. Kang, F. Dartu, and K. Roy, "Timed input pattern generation for an accurate delay calculation under multiple input switching," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 3, pp. 497–502, Mar. 2010, doi: 10.1109/TCAD.2009.2035482.
- [34] D. Rabe and W. Nebel, "New approach in gate-level glitch modeling," in *Proc. Eur. Design Autom. Conf. (EURO-DAC)*, 1996, pp. 66–71, doi: 10.1109/EUR-DAC.1996.558079.

- [35] P. Israsena and S. Summerfield, “Novel pattern-based power estimation tool with accurate glitch modeling,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2000, pp. 721–724, doi: 10.1109/ISCAS.2000.858853.
- [36] D. Khalil, D. Sinha, H. Zhou, and Y. Ismail, “A timing-dependent power estimation framework considering coupling,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 6, pp. 843–847, Jun. 2009, doi: 10.1109/TVLSI.2008.2008739.
- [37] M. Meixner and T. G. Noll, “Limits of gate-level power estimation considering real delay effects and glitches,” in *Proc. Int. Symp. System-on-Chip (SoC)*, 2014, pp. 1–7, doi: 10.1109/ISSOC.2014.6972437.
- [38] M. Meixner and T. G. Noll, “Accurate estimation of CMOS power consumption considering glitches by using waveform lookup,” *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 64, no. 7, pp. 787–791, Jul. 2017, doi: 10.1109/TC-SII.2016.2642179.
- [39] M. Ebrahimi, R. Seyyedi, L. Chen, and M. B. Tahoori, “Event-driven transient error propagation: A scalable and accurate soft error rate estimation approach,” in *Proc. 20th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2015, pp. 743–748, doi: 10.1109/ASPDAC.2015.7059099.
- [40] G.-I. Paliaroutis, P. Tsoumanis, D. Garyfallou, A. Vagenas, N. Evmorfopoulos, and G. Stamoulis, “Accurate soft error rate evaluation using event-driven dynamic timing analysis,” in *Proc. Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, 2023, pp. 1–6, doi: 10.1109/DFT59622.2023.10313573.
- [41] G.-I. Paliaroutis, P. Tsoumanis, D. Garyfallou, A. Vagenas, N. Evmorfopoulos, and G. Stamoulis, “A gate-level SER estimation tool with event-driven dynamic timing and SET height consideration,” *IEEE Trans. Device Mater. Rel.*, vol. 25, no. 1, pp. 17–26, 2025, doi: 10.1109/TDMR.2024.3508696.
- [42] A. Vagenas, D. Garyfallou, N. Evmorfopoulos, and G. Stamoulis, “Advanced gate-level glitch modeling using ANNs,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2024, pp. 1–6, doi: 10.1145/3649329.3657351.

- [43] P. Beniwal, S. Saurabh, N. V. Chowdary, S. Skariah, A. Mandal, and R. Venkatraman, "LiMo: A framework leveraging machine learning for multi-input switching timing models of complex logic gates," *IEEE Trans. Circuits Syst. I: Reg. Papers*, 2025, doi: 10.1109/TCSI.2025.3579073.
- [44] M. Hashimoto, H. Onodera, and K. Tamaru, "A power optimization method considering glitch reduction by gate sizing," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 1998, pp. 221–226, doi: 10.1145/280756.280907.
- [45] E. T. A. F. Jacobs and M. R. C. M. Berkelaar, "Gate sizing using a statistical delay model," in *Proc. Design Autom. Test Europe Conf. (DATE)*, 2000, pp. 283–290, doi: 10.1109/DATE.2000.840285.
- [46] T. Raja, V. D. Agrawal, and M. L. Bushnell, "Variable input delay CMOS logic for low power design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 10, pp. 1534–1545, Oct. 2009, doi: 10.1109/TVLSI.2009.2027567.
- [47] S. Bathla, R. M. Rao, and N. Chandrachoodan, "A simulation-based metric to guide glitch power reduction in digital circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 2, pp. 376–386, Feb. 2019, doi: 10.1109/TVLSI.2018.2882994.
- [48] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992, doi: 10.1109/4.126534.
- [49] P. Beniwal, S. Saurabh, A. Mandal, S. Skariah, and R. Venkatraman, "Machine Learning-Driven Flip-Flop Timing Model and Its Application in Resolving Marginal Timing Violations," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Dec. 2025. doi: 10.1145/3785294.
- [50] R. W. Phelps, "Advanced library characterization for high-performance ASIC," in *Proc. 4th Annu. IEEE Int. ASIC Conf. Exhibit*, 1991, pp. P15-3/1, doi: 10.1109/ASIC.1991.242888.
- [51] M. A. Cirit, "Characterizing a VLSI standard cell library," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, 1991, pp. 25.7/1–25.7/4, doi: 10.1109/CICC.1991.164076.

- [52] S. Saurabh, H. Shah, and S. Singh, “Timing closure problem: Review of challenges at advanced process nodes and solutions,” *IETE Tech. Rev.*, pp. 1–14, 2018, doi: 10.1080/02564602.2018.1531733.
- [53] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E. G. Friedman, “Pessimism reduction in static timing analysis using interdependent setup and hold times,” in *Proc. 7th Int. Symp. Quality Electron. Design (ISQED)*, 2006, pp. 164–169, doi: 10.1109/ISQED.2006.100.
- [54] S. Srivastava and J. Roychowdhury, “Interdependent latch setup/hold time characterization via Euler–Newton curve tracing on state-transition equations,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2007, pp. 136–141.
- [55] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E. G. Friedman, “Exploiting setup-hold-time interdependence in static timing analysis,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 6, pp. 1114–1125, Jun. 2007, doi: 10.1109/TCAD.2006.885834.
- [56] S. Srivastava and J. Roychowdhury, “Independent and interdependent latch setup/hold time characterization via Newton–Raphson solution and Euler curve tracking of state-transition equations,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 5, pp. 817–830, May 2008, doi: 10.1109/TCAD.2008.917595.
- [57] S. Hatami, H. Abrishami, and M. Pedram, “Statistical timing analysis of flip-flops considering codependent setup and hold times,” in *Proc. ACM Great Lakes Symp. VLSI (GLSVLSI)*, New York, NY, USA, 2008, pp. 101–106, doi: 10.1145/1366110.1366135.
- [58] E. Salman and E. G. Friedman, “Utilizing interdependent timing constraints to enhance robustness in synchronous circuits,” *Microelectron. J.*, vol. 43, no. 2, pp. 119–127, Feb. 2012, doi: 10.1016/j.mejo.2011.11.005.
- [59] N. Chen, B. Li, and U. Schlichtmann, “Iterative timing analysis based on non-linear and interdependent flip-flop modelling,” *IET Circuits Devices Syst.*, vol. 6, no. 5, pp. 330–337, 2012, doi: 10.1049/iet-cds.2011.0347.

- [60] A. B. Kahng and H. Lee, “Timing margin recovery with flexible flip-flop timing model,” in *Proc. IEEE Int. Symp. Quality Electron. Design (ISQED)*, Santa Clara, CA, USA, 2014, pp. 496–503, doi: 10.1109/ISQED.2014.6783367.
- [61] H. Seo, J. Heo, and T. Kim, “Clock skew optimization for maximizing time margin by utilizing flexible flip-flop timing,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Santa Clara, CA, USA, 2015, pp. 35–39.
- [62] J. Heo and T. Kim, “Timing analysis and optimization based on flexible flip-flop timing model,” in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Pittsburgh, PA, USA, 2016, pp. 42–46, doi: 10.1109/ISVLSI.2016.41.
- [63] G. L. Zhang, B. Li, and U. Schlichtmann, “PieceTimer: A holistic timing analysis framework considering setup/hold time interdependency using a piecewise model,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2016, pp. 1–8, doi: 10.1145/2966986.2967064.
- [64] Y. M. Yang, K. H. Tam, and I. H. R. Jiang, “Criticality-dependency-aware timing characterization and analysis,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6, doi: 10.1145/2744769.2744812.
- [65] H. A. Balef, H. Jiao, J. P. de Gyvez, and K. Goossens, “An analytical model for interdependent setup/hold-time characterization of flip-flops,” in *Proc. 18th Int. Symp. Quality Electron. Design (ISQED)*, Santa Clara, CA, USA, 2017, pp. 209–214, doi: 10.1109/ISQED.2017.7918318.
- [66] P. Cao, Y. Qin, and H. Jiang, “Timing analysis and optimization method with interdependent flip-flop timing model for near-threshold design,” *Electronics*, vol. 11, p. 3670, 2022, doi: 10.3390/electronics11223670.
- [67] TAU Contest, “TAU contest resources,” Jan. 29, 2025. [Online]. Available: <https://sites.google.com/view/tau-contest-2019/home/resources>
- [68] A. B. Kahng, “New directions for learning-based IC design tools and methodologies,” in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jeju, South Korea, 2018, pp. 405–410, doi: 10.1109/ASPDAC.2018.8297357.

- [69] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. [Online]. Available: <https://www.tensorflow.org>. doi: 10.48550/arXiv.1603.04467.
- [70] O. V. S. Shashank Ram and S. Saurabh, “Modeling multiple input switching in timing analysis using machine learning,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 723–734, Apr. 2021, doi: 10.1109/T-CAD.2020.3009624.
- [71] S. Masoumian *et al.*, “Reliability analysis of FinFET-based SRAM PUFs for 16 nm, 14 nm, and 7 nm technology nodes,” in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, Antwerp, Belgium, 2022, pp. 1189–1192, doi: 10.23919/DAT54114.2022.9774735.
- [72] L. B. Moraes, A. L. Zimpeck, C. Meinhardt, and R. Reis, “FinFET inverter designs: Behavior and challenges of process variability,” *J. Integr. Circuits Syst.*, pp. 1–13, 2021, doi: 10.29292/jics.v16i2.222.
- [73] K. P. Parker and E. J. McCluskey, “Probabilistic Treatment of General Combinational Networks,” in *IEEE Transactions on Computers*, vol. C-24, no. 6, pp. 668–670, June 1975, doi: 10.1109/T-C.1975.224279.
- [74] P. Beniwal and S. Saurabh, “Artificial Neural Network-based Prediction and Alleviation of Congestion during Placement,” *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSI-D)*, Kolkata, India, 2024, pp. 300–305. doi: 10.1109/VLSID60093.2024.00056.
- [75] M. Pan and C. Chu, “FastRoute: A step to integrate global routing into placement,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2006, pp. 464–471.
- [76] Y. Wei *et al.*, “GLARE: Global and local wiring aware routability evaluation,” in *Proc. Design Autom. Conf. (DAC)*, 2012, pp. 768–773.
- [77] H. J. L. Westra, “Congestion analysis and management,” Ph.D. dissertation, Dept. Elect. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2009.

- [78] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi, “BEOL stack-aware routability prediction from placement using data mining techniques,” in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Scottsdale, AZ, USA, 2016, pp. 41–48.
- [79] W.-T. J. Chan, P. Ho, A. B. Kahng, and P. Saxena, “Routability optimization for industrial designs at sub-14 nm process nodes using machine learning,” in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2017.
- [80] L.-C. Chen, C.-C. Huang, Y.-L. Chang, and H.-M. Chen, “A learning-based methodology for routability prediction in placement,” in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, 2018, pp. 1–4.
- [81] A. F. Tabrizi, N. K. Darav, L. Rakai, I. Bustany, A. Kennings, and L. Behjat, “Eh?Predictor: A deep learning framework to identify detailed routing short violations from a placed netlist,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 6, pp. 1177–1190, Jun. 2020.
- [82] Y.-Y. Huang, C.-T. Lin, W.-L. Liang, and H.-M. Chen, “Learning-based placement refinement to reduce DRC short violations,” in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, 2021, pp. 1–4.
- [83] Z. Zhou, S. Chahal, T.-Y. Ho, and A. Ivanov, “Supervised-learning congestion predictor for routability-driven global routing,” in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, 2019, pp. 1–4.
- [84] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Y. Young, “PROS: A plugin for routability optimization applied in state-of-the-art commercial EDA tools using deep learning,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–8.
- [85] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, “Global placement with deep learning-enabled explicit routability optimization,” in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, 2021, pp. 1821–1824.
- [86] A. B. Kahng and S. Reda, “Digital layout-placement,” in *EDA for IC Implementation, Circuit Design, and Process Technology*. Boca Raton, FL, USA: CRC Press, 2006.

- [87] V. Yutsis, I. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, “ISPD 2014 benchmarks with sub-45 nm technology rules for detailed-routing-driven placement,” in *Proc. Int. Symp. Phys. Design (ISPD)*, Mar. 2014, pp. 161–168.
- [88] I. Bustany, D. G. Chinnery, J. R. Shinnerl, and V. Yutsis, “ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement,” in *Proc. Int. Symp. Phys. Design (ISPD)*, 2015.
- [89] Cadence Design Systems, “Innovus implementation system.” [Online]. Available: [https://www.cadence.com/en\\_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html)
- [90] C. Spearman, “The proof and measurement of association between two things,” *Amer. J. Psychol.*, vol. 15, no. 1, pp. 72–101, 1904.
- [91] N. Shrestha, “Detecting multicollinearity in regression analysis,” *Amer. J. Appl. Math. Stat.*, Jun. 2020.
- [92] J. H. Kim, “Multicollinearity and misleading statistical results,” *Korean J. Anesthesiol.*, 2019.



## LIST OF PAPERS BASED ON THESIS

1. **P. Beniwal**, S. Saurabh, N. V. Chowdary, S. Skariah, A. Mandal, and R. Venkatraman, "LiMo: A Framework Leveraging Machine Learning for Multi-Input Switching Timing Models of Complex Logic Gates," *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 72, no. 12, pp. 7979–7992, Dec. 2025. [10.1109/TCSI.2025.3579073](https://doi.org/10.1109/TCSI.2025.3579073).
2. **P. Beniwal**, A. Mandal, R. Venkatraman, S. Skariah, and S. Saurabh, "Modeling Glitches due to Multiple Input Switching using Machine Learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Mar. 2026. doi: [10.1145/3802920](https://doi.org/10.1145/3802920).
3. **P. Beniwal**, S. Saurabh, A. Mandal, S. Skariah, and R. Venkatraman, "Machine Learning-Driven Flip-Flop Timing Model and Its Application in Resolving Marginal Timing Violations," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Dec. 2025. doi: [10.1145/3785294](https://doi.org/10.1145/3785294).
4. **P. Beniwal** and S. Saurabh, "Artificial Neural Network-based Prediction and Alleviation of Congestion during Placement," *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, Kolkata, India, 2024, pp. 300–305. [10.1109/VLSID60093.2024.00056](https://doi.org/10.1109/VLSID60093.2024.00056).

## LIST OF OTHER PAPERS

1. D. Yadav, P. Bhagat, **P. Beniwal**, and S. Saurabh, “Codriver: A Tool for Coverage-Driven Functional Verification of RISC-V Processors,” in *VLSI for Embedded Intelligence*, A. Gupta, J. G. Pandey, N. Chaturvedi, and D. Dwivedi, Eds., Lecture Notes in Electrical Engineering, vol. 1210, Springer, Cham, pp. 157–169, 2025. DOI: [10.1007/978-981-97-3756-7\\_13](https://doi.org/10.1007/978-981-97-3756-7_13).
2. U. Singadiya, **P. Beniwal**, and S. Saurabh, “HAWK: Machine Learning-Driven Waiver of Marginal Hold-Time Violations with Process-Variation Awareness,” accepted for publication in the *2026 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA)*, 2026 (to appear). **Recipient of the Best Paper Award.**
3. U. Singadiya, **P. Beniwal**, S. Saurabh, and A. Mandal, “Hitch: Leveraging Machine Learning for Hold Violation Waiver Through Internal Glitch Height Estimation,” accepted for publication in the *IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2026)*, 2026 (to appear). **Recipient of the Best Poster Award.**

## **BRIEF BIODATA OF THE AUTHOR**

Pooja Beniwal (Graduate Student Member, IEEE) received the B.Tech. degree in Electronics and Communication Engineering and the M.Tech. degree in VLSI Design. She has qualified GATE and UGC-NET (JRF). She is currently a Ph.D. candidate in the Department of Electronics and Communication Engineering at Indraprastha Institute of Information Technology Delhi (IIIT Delhi), India.

During her doctoral studies, she led an SRC-funded research project in collaboration with Texas Instruments, IBM, and Siemens. Her research focuses on applying Machine Learning (ML) techniques to Electronic Design Automation (EDA), with particular interests in standard cell characterization, static timing analysis, and physical design. She has contributed to several high-impact publications in IEEE TCAS-I, ACM TODAES, VLSID, VLSI TSA, ISVLSI and VDAT.