2026 Design, Automation & Test in Europe Conference (DATE 2026)

Towards learning-based gate-level glitch analysis

Anastasis Vagenas *, Dimitrios Garyfallou, and George Stamoulis
Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece
{avagenas, digaryfa, georges}@e-ce.uth.gr

***Abstract*—In advanced technology nodes, accurate glitch modeling is crucial for designing high-performance, energy-efficient, and reliable integrated circuits. In this work, we present a new approach for gate-level glitch propagation modeling, employing efficient Artificial Neural Networks (ANNs) to accurately estimate glitch shape characteristics, propagation delay, and power consumption. Moreover, we propose an iterative workflow that integrates our models into standard cell libraries, exploiting the available accuracy and size trade-off. Experimental results on gates implemented in 7 nm FinFET technology indicate that our ANNs exhibit a strong correlation with SPICE (R^2 over 0.99). Therefore, our approach could enable accurate full-chip glitch analysis and effectively guide glitch reduction techniques.**

I. Introduction

As process geometries shrink, accurate modeling of glitches caused by Multiple Input Switching (MIS) plays a critical role in designing high-performance, energy-efficient, and reliable integrated circuits. During logic transitions, multiple signals can switch simultaneously at the inputs of gates, leading to unwanted complete or partial voltage swings at the outputs, commonly referred to as glitches. Increasing operating frequencies and process-induced variations raise the probability of MIS-induced glitches, especially in dense operation-intensive designs, such as mobile and artificial intelligence applications [1]. Once generated, glitches may propagate through the circuit, violating the gate noise margins [2] and leading to timing and logic errors. Moreover, glitches lead to substantial dynamic power dissipation, contributing up to 70% of total power in arithmetic units, especially in modern FinFET devices with reduced leakage power [3].

SPICE is the golden standard for glitch analysis but suffers from excessive simulation times. Instead, most tools operate at the gate level, employing ad-hoc approaches, such as inertial and degradation [4], involution [5], or derate-based models [6], to approximate glitches and their power consumption. However, they cannot accurately capture the analog shape of glitch waveforms, particularly the timing and location of the peak voltage, with industrial tools resulting in 40% glitch power estimation error [7]. In a recent study, Machine Learning (ML) was used to model the generation of MIS-induced glitches; nevertheless, glitch propagation was neglected [8].

In this paper, we introduce an ML approach for modeling gate-level glitch propagation. We utilize efficient and compact Artificial Neural Networks (ANNs) that can accurately estimate the output glitch shape, propagation delay, and power dissipation. Moreover, we present an iterative workflow that integrates our ANNs into standard cell libraries, exploiting the available trade-off between model size and accuracy. Thus, our ML approach could be combined with glitch generation models [8] to enable accurate full-chip gate-level glitch analysis and effectively guide glitch reduction techniques.

II. Proposed approach

A. Glitch propagation modeling

Fig. 1 depicts our proposed approach for efficiently modeling glitches. Each glitch voltage waveform is characterized by three time instants, T_{start}, T_{end}, and T_{peak}. In particular, T_{start} and T_{end} represent the first and last time instants when the glitch waveform crosses specific VDD thresholds (e.g., 5% for rising glitches and 95% for falling glitches), while T_{peak} corresponds to the time instant of the glitch peak voltage (V_{peak}). Utilizing these time instants, we can derive the primary shape characteristics of each glitch, namely its width (W), height (H), and time-to-peak value (TP), which can be computed as $W = T_{\text{end}} - T_{\text{start}}$, $H = |V_{\text{rail}} - V_{\text{peak}}|$ (where V_{rail} is VDD or VSS), and $TP = T_{\text{peak}} - T_{\text{start}}$, respectively.

Given the primary shape characteristics of the input glitch ($W_{\text{in}}, H_{\text{in}}, TP_{\text{in}}$) and the output load ($C_{\text{L}}$) as input features, our ML models can predict the output glitch shape characteristics ($W_{\text{out}}, H_{\text{out}}, TP_{\text{out}}$), propagation delay ($D_{\text{out}}$), and power ($E_{\text{out}}$). W_{in} and H_{in} are the primary attributes that determine whether the input glitch will propagate through the gate and influence the size of the output glitch ($W_{\text{out}}, H_{\text{out}}$), while TP_{in} predominantly defines the shape of the output glitch (TP_{out}), which is essential for accurate noise and dynamic timing analysis [2], [9]. To determine the location of the output glitch relative to the input glitch, we utilize D_{out}, which is similar to the NLDM delay and is computed as $D_{\text{out}} = T^{\text{out}}_{\text{peak}} - T^{\text{in}}_{\text{peak}}$. Finally, E_{out} is obtained by integrating the VDD power waveform. Note that our method also enables the calculation of the output glitch switching (E_{SW}) and internal (E_{INT}) power, as $E_{\text{SW}} = \frac{VDD \cdot H_{\text{out}} \cdot C_L}{2}$ and $E_{\text{INT}} = E_{\text{out}} - E_{\text{SW}}$ [10].

B. Workflow for ANN-based glitch propagation modeling

To effectively model glitch propagation across multiple cells in a standard cell library, we adopted a fine-grained approach. Each input-to-output timing arc of a cell susceptible to glitch propagation (i.e., glitch arc) is modeled using five separate ANN models, each predicting the corresponding output glitch parameter. These models are trained on datasets generated

* The research was conducted within the operating framework of the University of Thessaly Innovation, Technology Transfer Unit and Entrepreneurship Center "One Planet Thessaly", under the "Scholarship Grants to University of Thessaly Doctoral Candidates", and was funded by the Special Account of Research Grants of the University of Thessaly.

978-3-9826741-1-7/DATE26/© 2026 EDAA

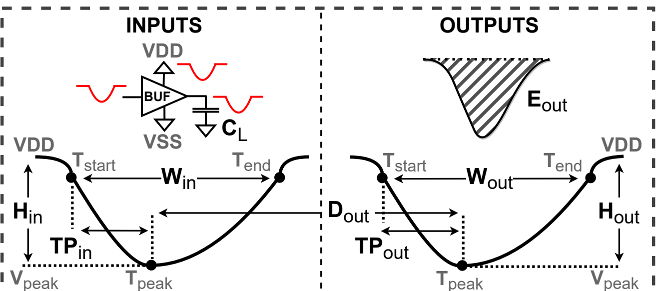

Fig. 1: Features and outputs of our glitch propagation models.

through SPICE simulations of input glitches with varying features (i.e., W_{in}, H_{in}, TP_{in}), reserving 80% of the data for training and 20% for testing. Finally, model training was performed using the Adam optimizer [11], the Mean Squared Error loss function, and ReLU activation for the hidden layers.

Fig. 2 provides an overview of our iterative model generation methodology, outlining the inputs and primary steps of the workflow. Our framework requires three inputs, namely, the SPICE models and standard cell library of the gates to be characterized, and a user-defined configuration file. The corresponding library is used to extract the cells' logic function and capacitance ranges, while the configuration file specifies the candidate ANN architectures to be employed during training, along with essential workflow parameters, such as the initial number of data points and the model accuracy/size constraints.

Initially, the engine parses the input files and analyzes the cells for all glitch arcs, generating the SPICE decks and running simulations to produce the golden results. Then, for each glitch arc and output parameter (e.g., W_{out}), an ANN architecture is selected based on the user-provided configuration, and a model is trained using the respective dataset. In case the model fails to meet the target accuracy, we select a more complex architecture. In the absence of other available architectures, we augment the dataset with additional data points and restart the entire process. Once finalized, the ANN models are stored in the standard cell library or as separate binary files, including their architecture and parameters (weights and biases).

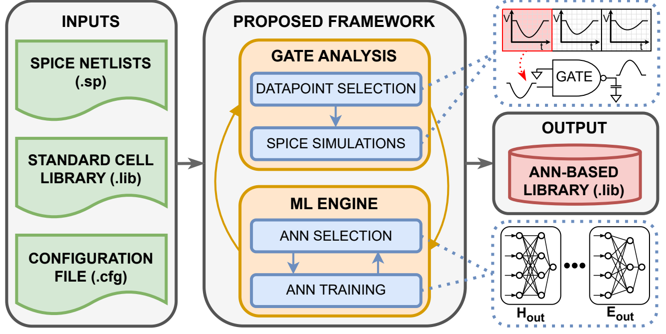

Fig. 2: Workflow for ANN-based glitch propagation modeling.

III. Experimental Results

To evaluate our glitch propagation modeling approach, we utilized the proposed framework on a subset of combinational gates from the ASU ASAP 7 nm PDK [12]. The golden SPICE results were obtained using Xyce [13] and model accuracy was assessed using the R^2 score. The SPICE data generation and model training typically take 30-120 s per gate, depending on the number of inputs and glitch arcs. For the framework configuration, we set the target accuracy threshold to 0.99 (R^2). All experiments were run on a workstation with a 3.70 GHz 12-thread CPU and 16 GB RAM.

Table I presents the mean R^2 score for each gate and output parameter across all glitch arcs, along with the combined memory footprint of the ANN models (when stored in binary format). As can be seen, the proposed ML models achieve exceptional correlation with SPICE across all gates, with every output meeting the target R^2 score of 0.99. The best predictions are made for W_{out} and E_{out} (0.998 and 0.999, respectively), while D_{out} exhibits the worst performance for most gates. Regarding memory, the disk storage requirements of the models range from 21.12 to 117.6 KB, with their size scaling linearly with the number of inputs and glitch arcs. Lastly, Fig. 3 illustrates the predictions of our ANN models for a rising glitch arc of a BUF gate, which further highlights their resistance to outliers and strong correlation with SPICE.

TABLE I: Accuracy and memory demands of our ANN models

Gate	R^2 score against SPICE					Model size (KB)
	H_{out}	W_{out}	D_{out}	TP_{out}	E_{out}	
INV	0.996	0.998	0.997	0.997	0.999	21.69
BUF	0.996	0.998	0.994	0.998	0.999	21.12
NAND2	0.997	0.998	0.991	0.995	0.999	47.12
AND2	0.995	0.998	0.995	0.998	0.999	45.08
NOR2	0.997	0.998	0.996	0.997	0.999	44.17
OR2	0.995	0.998	0.990	0.998	0.999	45.32
XNOR2	0.996	0.998	0.995	0.998	0.999	87.53
XOR2	0.996	0.998	0.994	0.998	0.999	86.07
NAND3	0.997	0.997	0.989	0.995	0.999	65.54
NOR3	0.997	0.997	0.990	0.995	0.999	70.91
AOI21	0.996	0.998	0.990	0.996	0.999	117.6
OAI21	0.996	0.998	0.991	0.996	0.999	116.3

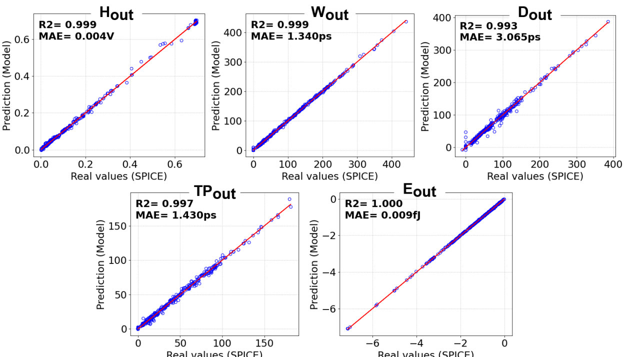

Fig. 3: Predictions of the proposed ANNs for a BUF gate.

IV. Conclusions and Future Work

This work presented a new ML-based method for efficient gate-level glitch propagation modeling. In the future, we plan to improve the performance of our approach by exploring alternative ML models, refining hyperparameters, and incorporating realistic input glitch waveforms during characterization. In addition, we intend to validate our framework across different technology nodes and operating conditions and utilize our ANNs for circuit-level glitch estimation.

References

[1] Synopsys - What is Glitch Power? Accessed: Jan. 19, 2026. [Online]. Available: https://www.synopsys.com/ai/what-is-glitch-power.html

[2] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer Science & Business Media, 2009.

[3] M. Meixner and T. G. Noll, "Accurate estimation of CMOS power consumption considering glitches by using waveform lookup," *IEEE Trans. on Circuits and Systems II Express Briefs*, vol. 64-II, no. 7, pp. 787–791, 2017.

[4] P. Ruiz de Clavijo Vazquez, J. Juan-Chico, M. Bellido, A. Acosta, and M. Valencia, "HALOTIS: High Accuracy Logic TIming simulator with inertial and degradation delay model," in *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 467–471, 2001.

[5] D. Öhlinger *et al.*, "The Involution Tool for Accurate Digital Timing and Power Analysis," *Integration*, vol. 76, pp. 87–98, 2021.

[6] Synopsys - PrimePower. Accessed: Jan. 19, 2026. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primepower.html

[7] M. Meixner and T. G. Noll, "Limits of gate-level power estimation considering real delay effects and glitches," in *Proc. of the International Symposium on System-on-Chip (SoC)*, 2014.

[8] A. Vagenas, D. Garyfallou, N. Evmorfopoulos, and G. Stamoulis, "Advanced gate-level glitch modeling using ANNs," in *Proc. of the 61st ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2024.

[9] G.-I. Paliaroutis *et al.*, "A Gate-Level SER Estimation Tool With Event-Driven Dynamic Timing and SET Height Consideration," *IEEE Trans. on Device and Materials Reliability*, vol. 25, no. 1, pp. 17–26, 2025.

[10] D. Rabe and W. Nebel, "New approach in gate-level glitch modelling," in *Proc. of the European Design Automation Conference (EURO-DAC)*, pp. 66–71, 1996.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] L. T. Clark *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, 2016.

[13] E. Keiter *et al.*, "Xyce Parallel Electronic Simulator Users' Guide Version 7.6." Sandia National Lab. (SNL-NM), Tech. Rep., 2022.