Towards SEU Fault Propagation Prediction with Spatio-temporal Graph Convolutional Networks

Li Lu*†, Junchao Chen*, Markus Ulbricht*, and Milos Krstic*†
*IHP-Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
†University of Potsdam, Potsdam, Germany
Emails: {lu, chen, ulbricht, krstic}@ihp-microelectronics.com

***Abstract*—Assessing Single Event Upset (SEU) sensitivity in complex circuits is increasingly important but challenging. This paper proposes an efficient approach using Spatio-temporal Graph Convolutional Networks (STGCN) to predict the results of SEU simulation-based fault injection. Representing circuit structures as graphs and integrating temporal data from the workload's waveform into these graphs, STGCN achieves a 94-96% prediction accuracy on four test circuits.**

***Index Terms*—Simulation-based fault injection, SEU faults, Machine Learning, Graph Neural Networks**

I. Introduction

Soft errors induced by high-energy particles, notably Single Event Upset (SEU), pose increasing concerns in semiconductor devices due to growing complexity and reduced dimensions. SEU, a prevalent soft error, particularly affects space applications, causing temporary bit slips in storage components like flip-flops. Despite being non-permanent, SEU faults can lead to unforeseen data corruption or system failures. Simulation-based fault injection, a widely used technique for circuit reliability analysis in the early stages of the design flow, can be excessively time-consuming, especially with complex circuits. The time needed for fault simulation may exponentially increase as circuit complexity grows, making exhaustive SEU fault injections on large circuits practically unfeasible.

Various approaches exist to tackle this issue, but they come with limitations. For instance, Statistical fault injection (SFI) [1] reduces simulation time by sampling faults, but its efficacy on complex circuits is uncertain due to potential inadequacies in fault sample representation. Improper sampling methods may lead to significant disparities between simulation and actual outcomes [2]. Machine learning (ML) technologies are increasingly applied across diverse fields to automate processes and reduce human resource demands. This prompts our exploration of ML for analyzing SEU faults.

This study delves into the potential of employing Spatio-temporal Graph Convolutional Networks (STGCN) [3], a subset of Graph Neural Networks (GNNs) [4], designed explicitly for graph data, to analyze SEU fault propagation in circuits. In simulation-based fault injection, the reliability assessment is primarily influenced by two factors: the circuit's architecture, representing spatial dependency, and the specific workload or test used in the simulation, introducing temporal dependency. These dependencies can be modeled as spatio-temporal graphs. We present an approach to model a circuit into spatio-temporal graphs containing the structural feature extracted from the circuit's netlist and sequential features generated from the Value Change Dump (VCD) waveform of the workload. STGCN can learn from these graphs to get the capacity to predict the SEU fault propagations. The proposed method is validated on four open-source circuits with different functions and complexities.

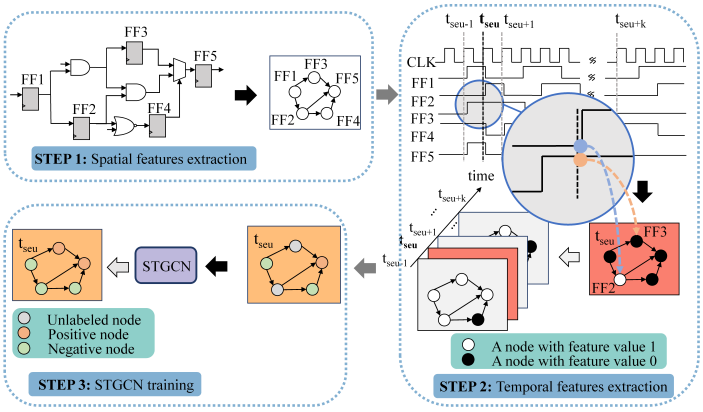

Fig. 1. The process of predicting the results of simulation-based SEU fault injection with STGCN.

II. Methodology

In this study, we utilize STGCN to predict simulation-based SEU fault injection outcomes in circuits, specifically focusing on whether a fault in a flip-flop can propagate to observation points. STGCN is tasked with outputting labels for SEU faults at specific time points, determining whether they can be detected at observation points. Positive nodes represent detectable faults, while negative nodes indicate undetectable faults. Figure 1 depicts the process of creating a dataset into spatio-temporal graphs and training a STGCN. The procedure comprises three steps. In the initial step, spatial features are extracted from the target circuit, forming the basis for constructing spatio-temporal graphs. The circuit is transformed into a graph, where each node corresponds to a flip-flop. In the second step, temporal information from the VCD waveform of the simulation test is integrated into the graph for a specific SEU injection time t_{seu}, generating a sequence of graphs based on a sampling window whose size is recorded as $time\_win\_size$ ($time\_win\_size = k+2$ in the example). These graphs, along with a partially labeled graph, serve as input for STGCN in the final step. The graph annotates a random selection of flip-flops with simulation-based SEU fault injection outcomes. STGCN is then trained on these inputs to predict labels for unannotated

979-8-3503-4859-0/DATE24/© 2024 EDAA

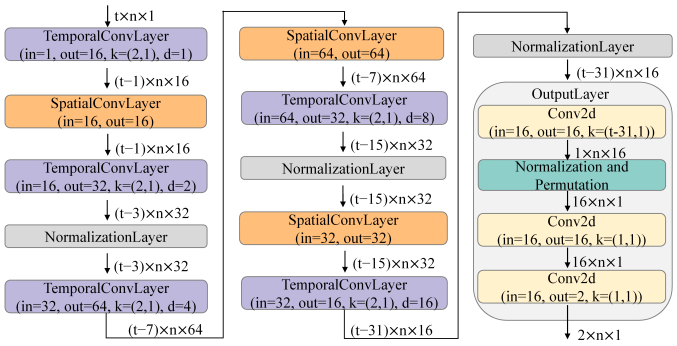

Fig. 2. The architecture of STGCN. The input of each layer indicates the dimension of the input feature tensor X. In each layer, in is the input feature size (corresponding to the 3^{rd} dimension of X), out is the output feature size, k is the kernel size, and d is the dilation rate.

nodes. The result is a fully annotated graph with prediction labels for previously unannotated nodes, indicating SEU fault simulation outcomes on all flip-flops at the specified time.

STGCN was initially designed for predicting traffic patterns, detailed in [3]. Our adapted STGCN, illustrated in Fig. 2, is derived from this and modified to suit our application. The input feature tensor, representing SEU fault simulation features at a time point, has dimensions $t \times n \times 1$, where t is $time\_win\_size$ and n is the number of flip-flops in the targeted circuit. The output tensor has dimensions $2 \times n \times 1$, reflecting the label (0 or 1) of each SEU fault in our datasets. Our STGCN consists of four types of layers. The temporal convolutional layer captures dynamic behaviors over time, employing 2-dimensional atrous convolution [5]. The spatial convolutional layer uses the convolution of GCN [6] to explore the structural features of the circuit. Following a structure similar to [3], a spatial layer bridges two temporal layers, concluding with a normalization layer to form the spatio-temporal convolutional block. The output layer transforms the generated features into final predictions.

III. Validation Results

We chose four circuits with varying functions and complexities to evaluate our method's effectiveness across diverse circuits. Three circuits (I2C, SPI, and XGE-MAC) are from OpenCores [7], and Ibex [8] is an open-source 32-bit RISC-V core. Reference simulations are performed using the Xcelium fault simulator from Cadence. For I2C and SPI (with 153 and 131 flip-flops, respectively), SEU faults are injected at 50 time points in each circuit. For XGE-MAC (1358 flip-flops) and Ibex (2126 flip-flops), 36 and 30 time points are randomly selected for SEU fault injection. The number of samples in Table I is calculated as the product of the number of flip-flops and injection time points. We allocate 80% of samples for training and the rest for validation. Our STGCN is built using the Deep Graph Library [9] based on PyTorch. Cross Entropy is the loss function during training, and all experiments run on a single Intel Xeon processor E5-4627V2.

We evaluate our model's performance using standard metrics, including precision, recall, and accuracy. The test circuits exhibit varying percentages of positive samples (refer to Table I). We experimentally search the space from 10 to 200 for the hyperparameter $time\_win\_size$, determining the best time window size for various circuits, with optimal values being 60, 40, 20, and 20 for I2C, SPI, XGE-MAC, and Ibex, respectively. Table I summarizes the best performance achieved for each circuit. Precision, recall, and accuracy on the relatively balanced datasets (I2C, SPI, and XGE-MAC) are all above 96%. Ibex's precision and recall are around 80%, but the accuracy is nearly 94%, demonstrating STGCN's competence on a complex circuit with an imbalanced dataset (Ibex).

TABLE I
Performance of Circuits on Test Circuits.

Circuit	Number of Samples	Percentage of Positive Samples (%)	Precision (%)	Recall (%)	Accuracy (%)
I2C	7650	70.56	97.25	97.71	96.52
SPI	6550	62.60	96.19	97.63	96.23
XGE-MAC	48888	52.12	96.54	96.33	96.26
Ibex	63780	16.51	83.32	78.55	93.96

IV. Discussion

SEU fault injection is time-intensive. For instance, simulating a test case ($3.6ms$) with 30 SEU injection time points on 2126 flip-flops takes approximately 45 days on Ibex. One approach to apply the proposed method to accelerate the process is to divide the circuit's flip-flops into three selections. Fault simulation is conducted on two selections (training and validation datasets), and a trained STGCN predicts fault simulation results on the third selection (test dataset) to reduce the overall required time.

Acknowledgment

The project underlying this report is funded by the German Federal Ministry of Education and Research under grant number 16ME0134 and 16KIS1339K. The responsibility for the content of this publication lies with the authors.

References

[1] P. Ramachandran *et al.*, "Statistical fault injection," in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. IEEE, 2008, pp. 122–127.

[2] C. Zhao *et al.*, "Fault samples simulation based on monte carlo method in testability virtual test," in *The Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety*. IEEE, 2011, pp. 358–362.

[3] B. Yu *et al.*, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.

[4] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[5] F. Yu *et al.*, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[6] T. N. Kipf *et al.*, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[7] OpenCores.org. (1999-2022). [Online]. Available: https://opencores.org/

[8] P. D. Schiavone *et al.*, "Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 2017, pp. 1–8.

[9] M. Wang *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.