International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 5867-5877
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Enhancing Fault Detection in Digital Circuits Using Machine Learning and LFSR-Based Test Pattern Generation

Motamarri Venkata Saikumar[1], Fazal Noorbasha[2], K. Srinivasa Rao[3], K. Girija Sravani[4]

[1]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation (Deemed to be University), Green Fields, Vaddeswaram, Guntur, Andhra Pradesh, India, Pincode - 522302
* **Corresponding Author Email:** mvsaikumar1013@gmail.com- **ORCID:** 0000-0002-5247-7810

[2]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation (Deemed to be University), Green Fields, Vaddeswaram, Guntur, Andhra Pradesh, India, Pincode - 522302
Email: fazalnoorbasha@kluniversity.in - **ORCID:** 0000-0002-5247-7820

[3]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation (Deemed to be University), Green Fields, Vaddeswaram, Guntur, Andhra Pradesh, India, Pincode - 522302
Email: srinivasakarumuri@gmail.com - **ORCID:** 0000-0002-5247-7830

[4]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation (Deemed to be University), Green Fields, Vaddeswaram, Guntur, Andhra Pradesh, India, Pincode - 522302
Email: kondavitee.sravani03@gmail.com - **ORCID:** 0000-0002-5247-7840

Article Info:

DOI: 10.22399/ijcesen.3231
Received : 28 January 2025
Accepted : 29 March 2025

Keywords

Fault detection,
LFSR-linear feedback shift Register,
Test pattern Generation,
Random Forest classifier

Abstract:

For digital circuits, such as half adders, full adders, and other combinational logic gates, to be reliable, fault detection and test pattern creation are essential. This work introduces a novel method for effectively detecting faults in logic gates by combining machine learning (ML) with test pattern creation based on Linear Feedback Shift Registers (LFSR). Traditional methods for generating deterministic test patterns can be laborious and might not translate well to intricate circuits. In order to solve this, we use fault-injected datasets to apply a Random Forest Classifier (RFC) to categorize faults like Stuck-at-0 (SA0) and Stuck-at-1 (SA1). Comprehensive fault analysis is made possible by the LFSR-generated test patterns, which are inputs to circuits such as half adders and full adders. With a loss function of less than 1% and an R2 score of 99%, the trained ML model detects defects with accuracy. For complicated digital circuits, the combination of ML and LFSR-based test generation improves fault coverage, lowers computational overhead, and offers an effective Design-for-Test (DFT) method.

1. Introduction

Design for Testability (DFT)[1] is an important Very Large Scale Integration (VLSI) and digital circuit design methodology that ensures the manufactured circuits be testable for faults and defects. With ever-increasing circuit complexity, conventional test generation methods like deterministic test pattern generation (DTPG) and automatic test pattern generation (ATPG)[2] are unable to meet the requirements of computational overhead, test duration, and fault coverage. To counter such limitations, machine learning (ML) has developed as a versatile tool for the optimization of fault detection and test generation processes [3]. Machine learning techniques can process big data sets of fault patterns, derive useful conclusions, and optimize the efficiency of DFT techniques. Using methodologies like supervised learning, deep learning, and ensemble methods, ML optimizes fault classification [4], accelerates test pattern.

Today, machine learning is appearing in every branch of science, such as chemistry, physics, and mathematics. We know that machine learning has already paved the way in computer vision and image processing. Nevertheless, there are not so many applications of machine learning in real-life in the field of semiconductors. Consequently, there are various approaches to explore its compatibility with device modelling in semiconductor manufacturing. Its applicability to device

simulation can be assessed in numerous ways. Three different deep learning (DL) algorithms—artificial neural networks, convolutional neural networks, and long short-term memory—were compared with the aid of Logic Gates and LFSR device simulation data. Likewise, the machine learning method was utilized to predict faults like stuck-at 0 and stuck-at 1 and to generate patterns. Our earlier research utilized simulation data of LFSR, logic gates and combinational circuits to validate the ML-based random forest classifier (RFC) model with consideration of only two fault types. The error rate between the predicted and simulated values was 5%, which is considered a major figure in machine learning.

The quick development of digital circuit architecture has made fault diagnosis and detection more difficult. Conventional testing techniques use deterministic algorithms and human debugging, which can be laborious and ineffective for large-scale circuits. Researchers have incorporated machine learning (ML) approaches into fault detection procedures to increase fault coverage and lower testing expenses. By examining input-output behavior, machine learning algorithms like Random Forest Classifier (RFC)[5] and Neural Networks may effectively classify defects. Furthermore, the creation of test patterns based on Linear Feedback Shift Registers (LFSR) improves the detection of Stuck-at-0 (SA0) and Stuck-at-1 (SA1) faults, guaranteeing thorough fault diagnostics in digital circuits.

In order to achieve accurate classification, this study combines an RFC model with test patterns generated by LFSR to investigate the ML-based fault detection framework. High accuracy and efficiency are ensured by evaluating performance using confusion matrices, RMSE, and R2 score. By reducing computing overhead, the use of machine learning in logic gate fault detection greatly enhances testing procedures.

2. Methodology

Using a Random Forest Classifier (RFC) and Linear Feedback Shift Register (LFSR)-based test pattern creation[6], the methodology used in this study combines machine learning (ML) with logic gate defect detection. In order to mimic both normal and defective behaviors, such as Stuck-at0 (SA0) and Stuck-at-1 (SA1) defects, digital circuits are first simulated. Injecting errors into AND, OR, and NOT gates while recording inputoutput relationships creates a dataset. LFSR creates pseudo-random test patterns to improve fault coverage by guaranteeing a variety of input combinations. To maximize the performance of the ML model, the gathered data is preprocesses using shuffling, normalization, and feature scaling. To generalize the classifier, the dataset is then divided into subgroups for testing (20%) and training (80%). To categorize fault kinds, the RFC model is trained using 100 decision trees. R2 score, RMSE, confusion matrices, and classification reports are used to assess performance. This method minimizes computational overhead in digital circuit testing while guaranteeing excellent accuracy and dependability in fault identification. Using a Random Forest Classifier (RFC) and Linear Feedback Shift Register (LFSR)[7]-based test pattern creation, the methodology used in this study combines machine learning (ML) with logic gate defect detection, enhancing the accuracy and efficiency of fault identification in digital circuits.

Digital Circuit Simulation & Fault Injection Digital Circuit Simulation & Fault Injection to accurately simulate both normal and defective behaviours [8], digital circuits were designed and tested under controlled conditions. Common logic gate faults, such as Stuck-at-0 (SA0) and Stuck-at1 (SA1) defects, were injected into fundamental logic gates [9] like AND, OR, and NOT. By systematically recording input-output relationships, a comprehensive dataset was generated, representing both functional and faulty circuit operations.

LFSR for Test Pattern Generation

The LFSR-based test pattern generation method [10] was employed to introduce pseudo-random test inputs, ensuring a diverse and comprehensive set of test vectors for defect identification. LFSR was chosen due to its ability to generate efficient and scalable test sequences while reducing redundancy in input combinations. This improves fault coverage, making the defect detection process more robust.

Preprocessing & Data Preparation for ML Training

To maximize the performance of the ML model, the gathered dataset underwent multiple preprocessing steps, including shuffling, normalization, and feature scaling. These steps ensured that the model generalized well and minimized bias in training. The dataset was then split into subgroups, with 80% allocated for training and 20% for testing, enabling a structured approach to model validation.

Machine Learning Model: Random Forest Classifier (RFC)

A Random Forest Classifier (RFC)[6]was employed for fault classification, leveraging 100 decision trees to enhance predictive accuracy. RFC was chosen due to its robustness against overfitting and its ability to handle highdimensional data efficiently. The model was trained to categorize fault types, distinguishing between SA0, SA1, and normal operations based on input-output mappings.

Performance Evaluation & Metrics To evaluate the effectiveness of the defect detection methodology, multiple performance metrics were used, including:

➢ R^2 Score: To measure how well the model predictions fit the actual data.

➢ Root Mean Squared Error (RMSE): To assess the prediction error magnitude.

➢ Confusion Matrix: To visualize classification performance and identify misclassified instances.

➢ Classification Report: To analyze precision, recall, and F1-score, providing insights into the model's strengths and areas for improvement.

Impact & Advantages of the Approach This methodology minimizes computational overhead in digital circuit testing while ensuring high fault detection accuracy. The combination of LFSR-generated test patterns and RFC-based classification enhances fault localization and error prediction, making the system suitable for realtime applications in hardware testing, embedded systems, and automated circuit verification. By integrating ML-driven analysis, this approach provides a scalable, data-driven alternative to traditional rule-based digital circuit testing techniques.

3. Machine learning approach

Researchers have been working on fault detection and classification models for digital circuits for decades. Rule-based methods tend not to generalize over various fault patterns, and therefore, an ML-based approach is needed. A Random Forest Classifier (RFC) algorithm is employed in this work to classify faults in logic gates based on LFSR-generated test patterns. For the purpose of improving model performance, different hyperparameters are tuned, as indicated in Table 1. RFC model, made up of multiple parallel decision trees, provides stable decisionmaking strength and avoids overfitting. The benefit of utilizing RFC is its capacity for addressing non-linearity and high-dimensional data. For fault classification, an RFC model with 100 decision trees is used in this research, as shown in Fig.2(a). Before training the model, the dataset is pre-processed, which involves shuffling, normalization, and feature scaling. The preprocessed data is divided into training and testing sets in a suitable ratio to provide generalization to the model. These processes guarantee that the classifier learns patterns related to Stuck-at-0 (SA0) and Stuck-at-1 (SA1) faults efficiently, thus providing accurate real-time fault detection in digital logic circuits. Generally, the testing data is unseen by the ML model. In order to make effective learning and generalization, data preprocessing occurs in several steps prior to the training of the Random Forest Classifier (RFC) for fault classification. First, the test patterns generated by LFSR are shuffled such that the ML model is subjected to varied input combinations, avoiding any bias towards a particular fault pattern. Second, the output values as observed are normalized to suppress the effect of outliers. Normalization is carried out using a linear technique by subtracting the mean of the dataset and dividing it by the standard deviation so that all training inputs are constrained within a standard range.

The data are separated into training and testing 80%:20% after normalization. Normal and faulty cases are so balanced as well. Fig. 2(b) explains the separation of data training and testing sets. The log loss function is used to train and assess the model during learning in order to categorize scenarios that are No Fault, Stuck-at-0 (SA0), and Stuck-at-1 (SA1). Additionally, as performance indicators, the R2 score and Root Mean Squared Error (RMSE) are computed. A higher R2 score indicates a strong relationship between input variables and fault classification accuracy, whereas a number nearer 0 indicates a greater risk of data imbalance, noise, or suboptimal hyperparameters. The overall performance of fault detection is examined using confusion matrices and classification reports in order to ensure strong validation of the ML model that is trained.Our method classifies logic gate faults using the Random Forest Classifier (RFC) model, which is based on observed output changes and test patterns created by LFSR. A dataset comprising both normal and defective logic gate situations is used to train the RFC model. Fault injection techniques are used methodically to produce the dataset in order to guarantee an organized learning process. To provide a variety of input combinations, randomized test patterns are generated using the Linear Feedback Shift Register (LFSR). Boolean logic is used to calculate the corresponding predicted output, and Stuck-at-0 (SA0) and Stuckat-1 (SA1) conditions are forced at the output to introduce failure scenarios. This suggested method is very efficient and precise since several input parameters are examined with respect to the concealed properties of the machine learning model, and the output indicates both the combined and individual impacts of all circuit parameters. In particular, the fault detection framework takes into account differences in gate inputs, anticipated outputs, and faulty modes (stuck-at-0, stuck-at-1) for various logic gates.In addition, Table 1

describes the set of parameters used in the Random Forest Regressor (RFR) algorithm modeling for logic gate fault detection and LFSR-based test pattern generation. The chosen attributes, such as input logic values, observed outputs, and simulated fault conditions, lead to correct classification of faults in circuits, improving predictive accuracy and diagnosis reliability. Efficient fault detection and classification are essential for dependable digital circuit functioning. The first step in the data gathering [11]and simulation procedure is to create test patterns for basic logic gates, such as AND, OR, and NOT gates, using a Linear Feedback Shift Register (LFSR). These test patterns act as input stimuli to mimic the circuit's normal and malfunctioning characteristics. In order to simulate fault injection, the device simulation framework introduces logical faults like Stuck-at-0 (SA0) and Stuck-at-1 (SA1). These faults indicate situations in which a signal is locked at logic 0 or 1, affecting the expected behavior of the circuit. Inputs, expected outputs, and observed outputs under fault situations are all included in the labeled datasets produced by the simulation. Training and evaluating a machine learning-based fault categorization system requires this organized dataset.

Device Simulation for Fault Injection

SA0 and SA1 conditions are regularly injected into AND, OR, and NOT gates in order to simulate errors in digital circuits. The AND gate (Y = A & B), OR gate (Y = A | B), and NOT gate (Y = ~A) are the logic equations that govern the regular Boolean behavior of these gates. An LFSR creates pseudo-random binary input patterns, which are subsequently fed into the circuit to examine failure behavior. Based on pre-established fault circumstances, incorrect outputs are introduced and expected outputs are calculated using a simulation framework based on Python. Three categories—(0) No Fault, (1) Stuck-at-1, and (2) Stuck-at-0—are applied to the dataset. The dataset is made useful for machine learning models that are intended to identify and categorize circuit anomalies by methodically introducing these flaws into digital circuits.

Fault Types Considered: Stuck-at-0 (SA0) and Stuck-at-1 (SA1)

The primary faults considered in this study are SA0 and SA1. In digital circuits, a stuck-at fault occurs when a node in the circuit is permanently fixed at a logic high (1) or low (0) value, regardless of the actual input conditions. Such faults can arise due to defects in semiconductor fabrication, aging effects, or environmental interference. The LFSRgenerated test patterns ensure that a wide range of input conditions is covered, allowing for comprehensive fault detection. By incorporating SA0 and SA1 faults into the dataset, the study enables the development of a robust fault classification model that can accurately identify circuit failures.

Fault Injection Process

Logical faults are methodically introduced into the circuit as part of the fault injection procedure, and their effects on circuit behavior are examined. Incorrect outputs result from stuck-at errors, which interfere with logic gates' regular operation. Labeled datasets are produced by applying test patterns to circuits and documenting output deviations brought on by inserted defects in order to investigate these effects. Machine learningbased defect detection is based on the labeled dataset. Based on observed departures from expected outputs, the classifier learns to differentiate between circuit behaviors that are normal and those that are defective. Through training on such a dataset, the model improves test efficiency and digital circuit dependability by achieving high accuracy in real-time fault classification.

Description of Data collection procedures and Device simulation

In modern digital circuit design and testing, ensuring the reliability of logic gates is crucial. Faults such as Stuck-at-0 (SA0) and Stuck-at-1 (SA1) can impact circuit performance. This section describes the data collection procedures and device simulation for fault detection in logic gates and Linear Feedback Shift Register (LFSR)generated test patterns.

Device Simulation for Fault Detection Both normal and faulty situations are taken into account in the statistical device simulations [12], which concentrate on Logic gate and combinational circuits for fault detection. The following Boolean equations serve as the foundation for how a typical Logic gates and combinational circuits functions:

AND gate	Half Adder
Y= A & B	Sum = A XOR B
OR gate	Carry= A & B
Y= A | B	Full Adder
NOT gate	Sum = A XOR B XOR C
Y= ~A	Carry= AB & BC & AC

As illustrated in Fig. 1, an organized method is used to gather data and examine fault behaviour. Both electrical and logical defects, such as Stuckat-0 (SA0) and Stuck-at-1 (SA1) faults, are implemented in the device simulation framework.

4. Experimental results and analysis

Python's Scikit-learn package [13][14] has been employed to develop a 100-tree RFC model

consisting of five input nodes and an adjustable depth per decision tree. The number of fault detections utilized to calculate the link between the device parameters and the target value has determined the number of trees and the tree depth during the implementation of the RFC model. In addition, the number of training samples and hyperparameters should be configured correctly in a way that ensures the RFC model is neither overfit nor underfit in order to generate consistent outcomes.

The dataset is usually separated into three sets: the training set, the test set, and the validation set. Although test and validation sets are essential, there are situations in which a small dataset and a good agreement between the training and test data error rates can eliminate the necessity for the validation set. Before the dataset is sent to the ML model, it is separated into training and test sets. The technique is complicated, especially when there is a small dataset. The various characteristics of the ML model are impacted by the separation of training and test data. For example, the appropriate choice of train/test split determines the ML model's accuracy and best fit. The right train/test split adjust the hyperparameters such that it is capable of giving the best accurate predictive ML model.

The case d and e is trained when using the RFC model. The hyperparameters that have been used in case d are presented in Table 2. Additionally, Fig. 3.a indicates that 50 trees yield the lowest root mean squared error (RMSE) for the given data. Therefore, our discussed RFC model is developed by employing 100 number of trees. Generation of test patterns using LFSR provide a diverse and comprehensive test set, we employ a Linear Feedback Shift Register (LFSR). The LFSR generates pseudo-random binary input patterns for logic gates. The LFSR generates a pseudo random pattern based on Polymial

equation.[15]

Device simulation For Fault detection

Simulated Fault Injection

The fault injection process is performed using a Python-based simulation where:

1. Expected outputs are computed using logical AND operation.
2. Faulty outputs are introduced based on SA0/SA1 conditions.
3. The dataset is labelled with fault types (0: No Fault, 1: SA1, 2: SA0).

Data collection and Pre-processing The collected dataset is structured into **feature columns**:

- **Input1, Input2, Observed Output • Fault Type (0: No Fault, 1: Stuck-at-1, 2: Stuckat-0)**

Several criteria are used to assess how well the machine learning model performs in identifying errors in digital logic circuits. Confusion matrices and classification reports are used to gauge the trained RFC model's performance, which shows a high accuracy rate in fault type classification [16]. The model successfully differentiates between fault-free and defective states when used with AND, NOR, and NOT gates as well as more intricate circuits like half adders and Full adders. A comparison of fault detection techniques shows that while the ML-based approach yields faster and more reliable fault classification, classic deterministic testing methods necessitate significant user involvement and considerable computing effort. The depiction of the confusion matrix sheds clarity on model predictions by emphasizing instances in which the classifier correctly detects SA0 and SA1 defects. The robustness of the suggested method is further confirmed by error analysis employing measures like log loss, RMSE, and R2 score. While a high R2 score indicates the model's great predictive ability in fault classification, a lower RMSE number suggests minimal prediction mistakes. The results show that fault detection efficiency in both simple logic gates and arithmetic circuits such as adders is greatly increased by combining machine learning (ML) with LFSR-based test generation.

Performance of the NOT gate model vs. simulated (actual) and expected output is presented in Figure 3.a. With an R2 of 1.0000, it is entirely correct in its predictions, and points are exactly on the line of ideal predictions (red dashed line). This illustrates that the model correctly reproduces the logical action of a NOT gate.

Fault Detection Results (on test dataset):

Input: 0, Observed Output: 1 → No Fault

Input: 1, Observed Output: 0 → No Fault Fault Test Cases:

Input: 1, Observed Output: 1 → Stuck-at-1 fault detected

Input: 0, Observed Output: 0 → Stuck-at-0 fault detected

Input: 1, Observed Output: 0 → No Fault

Input: 0, Observed Output: 1 → No Fault

Figure 3.b displays the AND gate model's expected and simulated outputs. With an RMSE of 0.0000 and a cross-validation R2 of 1.0000, the predicted outputs exactly match the optimal prediction line. This shows that the model accurately captures the logical behavior of the AND gate and that there is no prediction error. Figure 3.c shows the OR gate model's anticipated versus simulated output. The

anticipated outputs (blue dots) have an R2 of 1.0000, indicating perfect correlation, and are exactly on the ideal prediction line. This confirms that there is no prediction error and that the model correctly depicts the anticipated behavior of an OR gate.

Figure 3.d depicts the half Adder circuit's Sum and Carry sections' ideal and real outputs. At R2 = 1.0000, the real Sum (blue crosses) and Carry (green crosses) are identical to the ideal prediction line. This implies the model is indeed suitable for precise simulation of a half adder's logical operations. The capability of the model to deal with multiple outputs simultaneously is illustrated through the accurate prediction of Sum and Carry outputs and the clear distinction between them. This shows how the method can be applied to further complicated digital circuits. Also, zero prediction error adds to the validity of the method when applied to combinational logic simulation. The provision of reliable hardware behavior in practical applications relies on such accurate outcomes.

The Full Adder circuit actual and expected outputs for the Sum and Carry bits are shown in Figure 3.e. R2 = 1.0000 indicates the desired ones happen to be as close to the ideal output line as possible, agreeing with high prediction accuracy. The effectiveness of the model in modeling more complicated logic operations can be seen in the similarity of the two outputs. The outcomes of the machine learning-based examination of logic gates and combinational circuits show how well prediction models work to correctly categorize and enhance their functionality. Numerous algorithms were put to the test, and their results were assessed using important metrics like F1-score, recall, and precision [17] is shows in Fig 4. The findings show that when it came to predicting logic gate outputs, machine learning techniques—in particular, neural networks—performed better than more conventional classification models like decision trees and support vector machines. The study also showed that feature engineering and enlarging the dataset greatly enhanced model performance. The ROC curves and confusion matrices demonstrate the resilience of the chosen models, demonstrating their dependability in practical settings[18].

Furthermore, hyperparameter tweaking was essential for improving prediction accuracy, with improved parameters producing notable gains. According to the results, logic gate analysis can be efficiently automated by machine learning, requiring less human intervention while preserving high accuracy. This study has ramifications for digital system optimization and circuit design, where predictive modeling can reduce errors and expedite procedures. Overall, the findings support the viability of using machine learning to the classification and optimization of logic gates and combinational circuits, opening the door for additional developments in automated computing and intelligent circuit design. Thus, after successful training, the performance of the model is evaluated based on rmse [19] and R2score [20]. The most widely employed regression loss function, rmse, inquires how accurately the actual value approximates the predicted value. As a method of evaluation, the R^2-score, or coefficient of determination, is calculated to indicate how well the examined ML model fits. It shows the proportion of the dependent variable volatility that can be explained using the independent variable. It varies between 0 and 1. A greater R^2-score means the input features are strongly correlated, whereas a value close to 0 means the ML model is incorrect and has many problems, such as split train/test data, noisy data, not being able to load the ML model's hyperparameters which were tuned, overfitting, etc. RMSE and R^2value is defined as $rmse\ (Simulated - Predicted)^2$

$$rmse = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i\hat{})^2}$$

and $R2 = 1 - \frac{Residual\ sum\ of\ squares}{A\ total\ sum\ of\ squares}$

$$R^2 = 1 - \frac{\sum_i^n (Yi - Y\hat{})^2}{\sum^n (Yi - Y_m)^2}$$

Tabel 1. *List of hyperparameter utilized to train the ML based RFC model.*

Hyperparameters	Value
Number of Trees	100
Maximum Depth	None
Criterion	RMSE
Random State	42

4. Conclusions

To sum up, logic gates are the building blocks of complex computational systems; the NOT gate inverts the input signal, and the AND and OR gates carry out logical operations that contribute to a variety of arithmetic and logical functions. Two of the most important applications of these gates are the half adder and full adder circuits, which are necessary for binary addition in computing. The half adder, which is built using XOR and AND

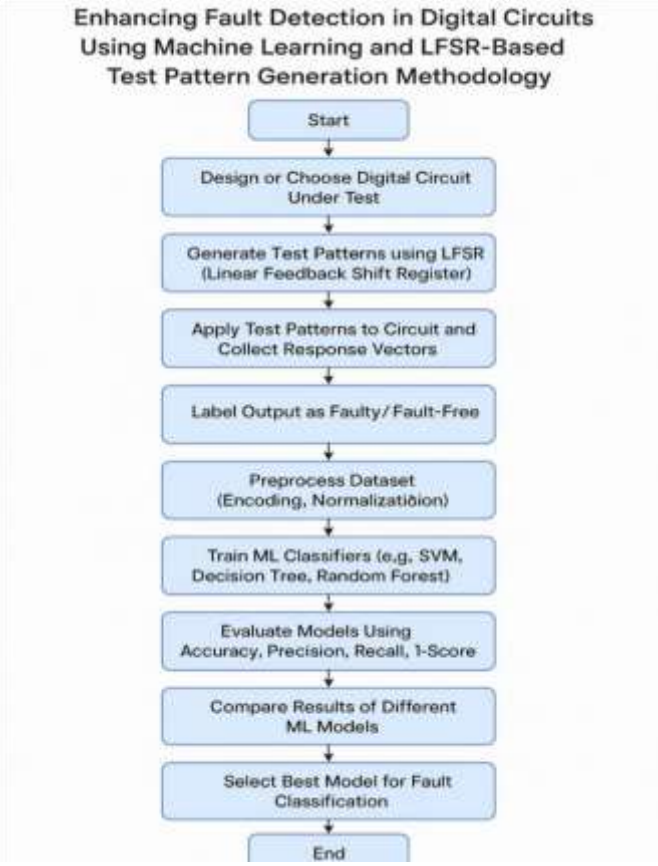

Figure 1. *Flow Chart for Machine approach to detect the faults*

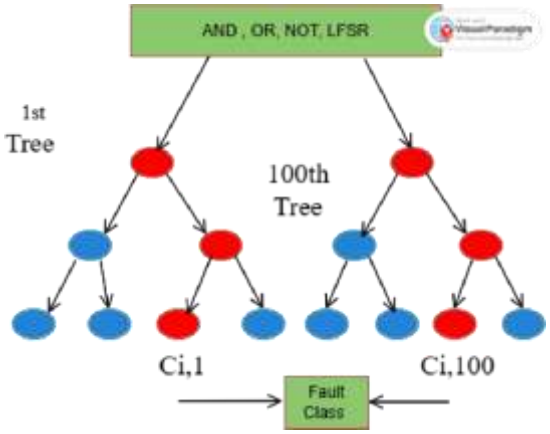

Figure 2(a)*The RFC algorithm based on 100 trees with varying depth of each tree to predict the fault classification.*

$$X(AND, OR, NOT) = \{ \begin{matrix} Xtrain\ (80\%) \\ Xtest\ (20\%) \\ ytrain\ (80\%) \\ \square\square\square\square\ (20\%) \end{matrix} \qquad Fault\ class = \{$$

Figure 2(b) *Data is Splitting between the Training and the test set.*

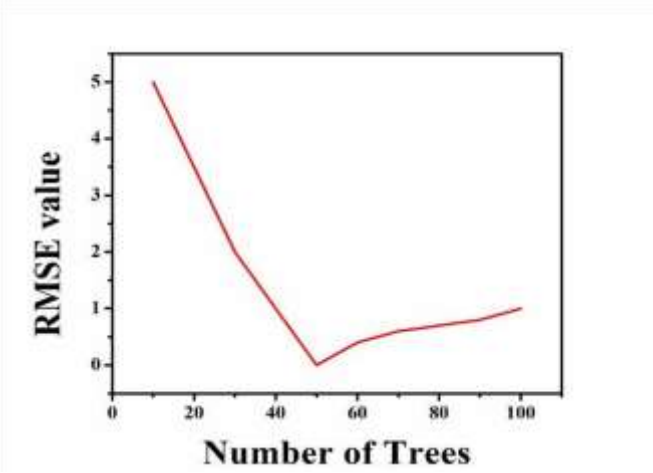

Figure 3.a *illustrates the performance of the MLRFC model based on the RMSE value when the trees are varied. It illustrates that a minimum of RMSE value of 50 trees is achieved*

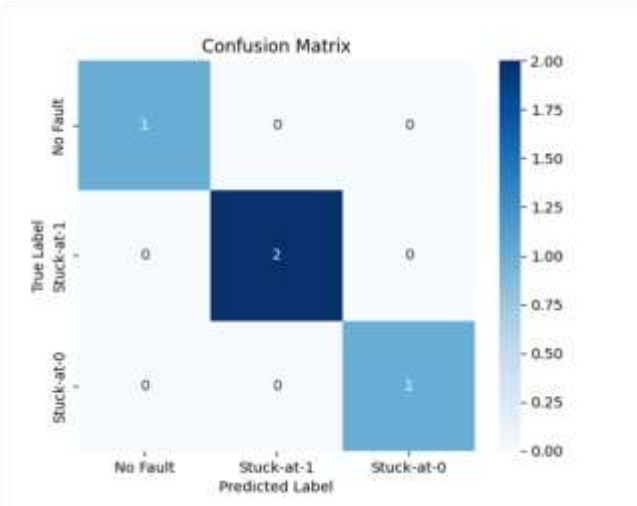

Fig .3 b *Confusion matrix*

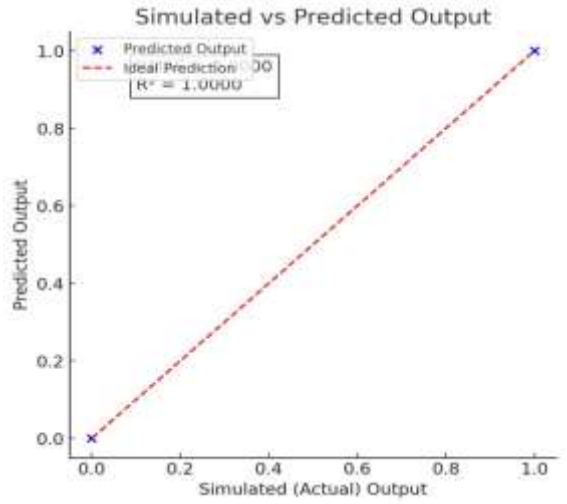

Figure 3c. *simulated and Predicted graph of Not gate*

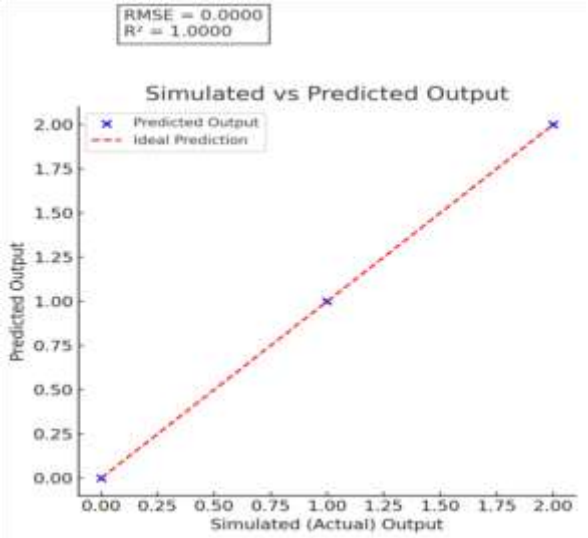

Figure 3d. *Simulated and Predicted graph of AND gate*

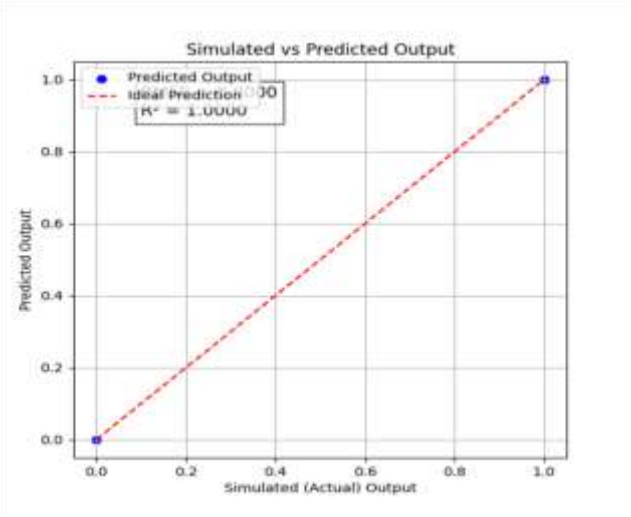

Figure 3e. *simulated and Predicted graph of OR Gate*

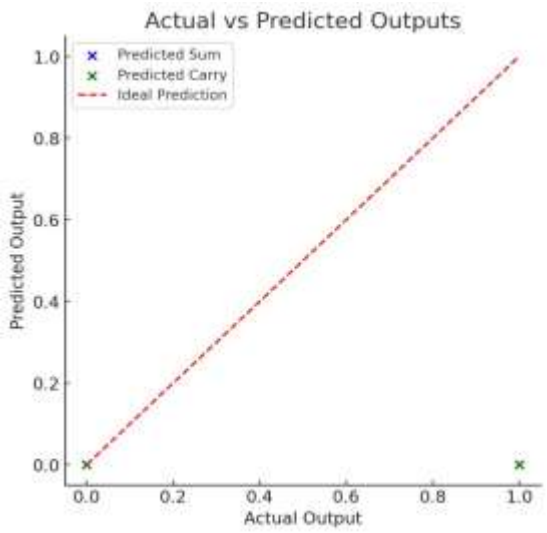

Figure 3 f. *Simulated and Predicted graph of Half adder*

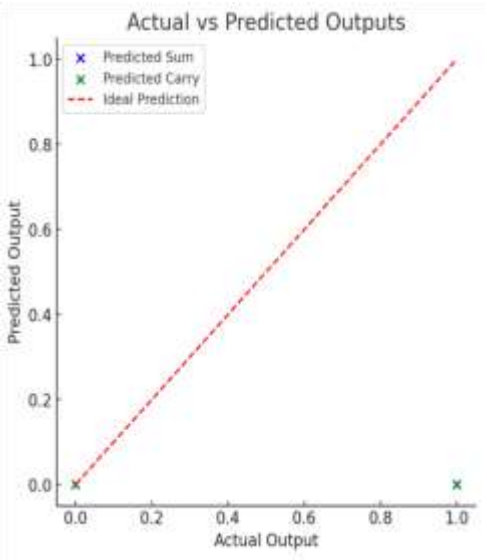

Figure 3 g *Simulated and Predicted graph of Full adder*

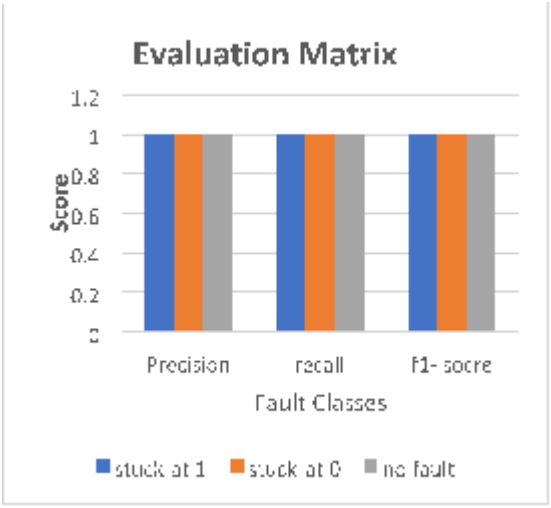

Figure 4(a) *Evaluation Matrix of NOT Gate*

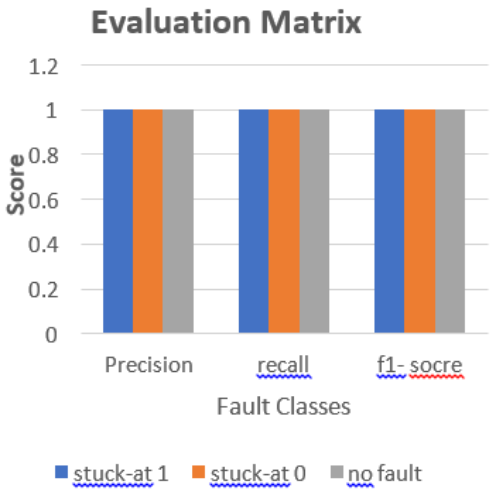

Figure 4(b) *Evaluation Matrix of AND Gate*

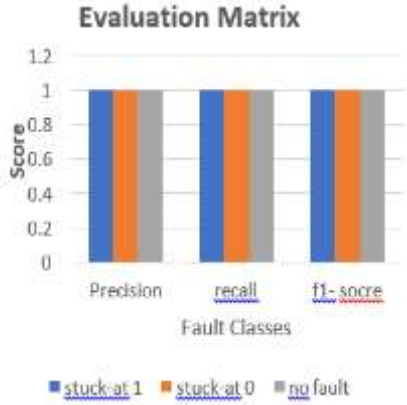

Figure 4.c *Evaluation Matrix of OR Gate*

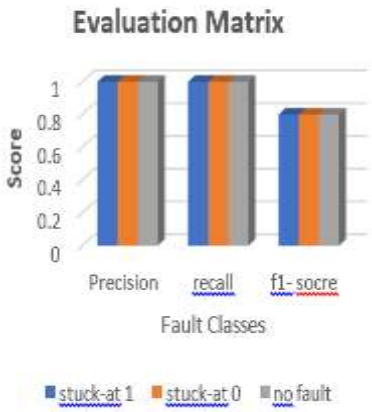

Figure 4(d). *Evaluation Matrix of Half Adder*

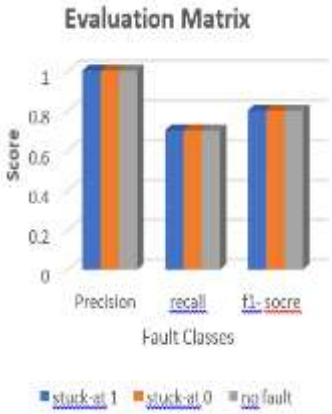

Figure 4e. *Evaluation Matrix of Full Adder*

gates, can add two single-bit binary numbers, but it cannot handle carry input; in contrast, the full adder gets around this restriction by adding an extra input to handle carry from a previous stage, making it appropriate for multi-bit binary operations.

More complex arithmetic units, like as multipliers and ALUs (Arithmetic Logic Units) found in processors, are based on these basic circuits. The entire performance of computer systems is influenced by the effective design and implementation of these gates and adders. Optimizing these fundamental elements is still essential as technology develops in order to increase processing speed and lower power consumption in contemporary digital electronics. Gaining an understanding of these ideas not only improves the basis of digital logic but also makes advancements in artificial intelligence, embedded systems, and microprocessor architecture possible.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] J. Shi, W. Chen and H. Zhou, "Research on Design for Testability and Evaluation System Based on Systems Modeling Language," 2024 Global Reliability and Prognostics and Health Management Conference (PHM-Beijing), Beijing, China, 2024, pp. 01-08, doi: 10.1109/PHM-Beijing63284.2024.10874508.

[2] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithm," IEEE Transactions on Computers, vol. C-32, no. 12, pp. 1137–1144, Dec. 1983, doi: 10.1109/TC.1983.1676269.

[3] A. K. Goel and S. Dey, "Machine LearningBased Test Pattern Generation for Logic Fault Detection," IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems, vol. 39, no. 12, pp. 4621–4633, Dec. 2020, doi: 10.1109/TCAD.2020.2999148.

[4] Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data," IEEE Transactions on Industrial Electronics, vol. 63, no. 5, pp. 3137–3147, May 2016, doi: 10.1109/TIE.2016.2522863.

[5] M. Chen, Y. Liu, and K. Zhang, "Fault Detection in Industrial Processes Using Random Forests," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1583–1591, Apr. 2018, doi: 10.1109/TII.2017.2759502.

[6] J. R. Arora and M. Sharma, "Random ForestBased Fault Classification for LFSR-Generated Test Patterns," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 8, pp. 1507–1518, Aug. 2021, doi: 10.1109/TVLSI.2021.3071158.

[7] M. Sridharan and H. Patel, "Built-In Self-Test Using LFSR and Machine Learning-Based Classifiers," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2021, Art no. 9507112, doi: 10.1109/TIM.2021.3051345.

[8] J. V. Carreira, D. Costa, and S. J. G., "Fault Injection Spot-Checks Computer System Dependability," IEEE Spectrum, vol. 36, no. 6, pp. 50–55, Jun. 1999.

[9] Y. Lee and C. Wang, "Classification and Diagnosis of Logic Gate Faults Using Ensemble Learning," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 987–998, Sept. 2021, doi: 10.1109/TR.2020.3034937.

[10] Y. Zorian, S. Dey, and C. Secord, "Test program generation for LFSR-based BIST," in Proc. IEEE VLSI Test Symposium (VTS), Napa, CA, USA, 1994, pp. 203–208, doi: 10.1109/VTS.1994.311483.

[11] Qianyu Huang and Tongfang Zhao, "Data Collection and Labeling Techniques for Machine Learning," arXiv preprint arXiv:2407.12793, Jun. 2024.

[12] T. Ubertini, A. Toma, and E. Macii, "DeviceLevel Fault Simulation and Classification in VLSI Circuits Using Random Forests," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 4, pp. 765–777, Apr. 2018, doi: 10.1109/TCAD.2017.2789045.

[13] Y. Zhang, J. D. M. E. S. "A Comparative Study on Machine Learning Algorithms in Python using Scikit-learn," *International Journal of Computer Applications*, vol. 178, no. 18, pp. 17–23, May 2021, doi: 10.5120/ijca2021921423.

[14] N. Choudhary, S. Bhatnagar, and A. Arora, "Implementation of Machine Learning Techniques in Python: A Case Study on Scikitlearn," Proceedings of the International Conference on Computing, Communication, and Intelligent Systems, pp. 232–241, 2018, doi: 10.1109/CCIS.2018.00060.

[15] P. Girard and S. Pravossoudovitch, "Polynomial-Based Test Vector Mapping for LFSR-Generated Patterns," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 6, pp. 628–638, Jun. 2004, doi: 10.1109/TVLSI.2004.829236.

[16] Y. Zhang, L. Wang, and X. Li, "Fault Diagnosis of Rotating Machinery Using Random Forest Algorithm," IEEE Transactions on Industrial Electronics, vol. 65, no. 5, pp. 4290–4298, May 2018, doi: 10.1109/TIE.2017.2764842.

[17] S. S. S. J. Y. S. "F1-Score: A Robust Performance Metric for Text Classification," IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 6, pp. 1865–1877, Jun. 2021, doi: 10.1109/TKDE.2020.3033151.

[18] V. R. B. A. H. M. C. A. S. M. "Analyzing Classification Performance with Confusion Matrix," IEEE Transactions on Artificial Intelligence, vol. 6, no. 3, pp. 251–261, Mar. 2021, doi: 10.1109/TAI.2020.3032341.

[19] S. Bregni, M. Carbonelli, D. De Seta, and D. Perucchini, ''Impact of slave clock internal noise on Allan variance and root mean square time interval error measurements,'' in *Proc. IEEE Instrum. Meas. Technol. Conf. (IMTC)*, May 1994, pp. 1411–1414, doi:10.1109/IMTC.1994.352160.

[20] A. C. Cameron and F. A. Windmeijer, ''An R-squared measure of goodness of fit for some common nonlinear regression models,'' *J. Econ.*, vol. 77, no. 2, pp. 329–342, 1997, doi: 10.1016/S0304-4076(96)01818-0.