International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 10 Issue 3, May-Jun 2026 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

AI-Inspired Fault Detection in VLSI Circuits Using Simulation Techniques

Pranav Dhumane, Dr. N. S. Narawade, Dr. N. S. Kothari

Department of Electronics and Telecommunication Engineering,
Samarth College of Engineering and Management, Belhe, Pune, India

ABSTRACT

Fault diagnosis and verification play a crucial role in ensuring the reliability and correctness of Very Large Scale Integration (VLSI) circuits, especially as modern digital systems continue to grow in complexity and scale. With the continuous reduction in transistor sizes and increasing integration density, digital circuits are more susceptible to logic- level faults arising from design inconsistencies, manufacturing variations, timing violations, and unexpected input conditions. Early identification of such faults is essential to prevent functional failures, reduce debugging time, and improve overall system reliability.

Traditional fault detection techniques predominantly rely on rule-based testing, manual inspection, or hardware-intensive verification tools. While these methods have been effective for small and medium-scale circuits, they often become inefficient, time-consuming, and difficult to scale when applied to complex digital systems. Additionally, many conventional approaches focus solely on fault identification without offering meaningful guidance for fault correction, which limits their usefulness in educational and early-stage design environments.

To address these limitations, this paper presents an AI-inspired fault detection framework for digital VLSI circuits based on logic gate simulation and intelligent decision analysis. Rather than employing data-intensive machine learning models, the proposed system emphasizes deterministic logic evaluation, transparency, and interpretability. Fundamental digital logic gates are modeled using Boolean logic principles, and the system evaluates circuit behavior by comparing observed outputs with theoretically expected results derived from standard truth tables.

The proposed framework not only detects faulty behavior at the logic level but also provides corrective insights by analyzing input-output relationships. When a mismatch between expected and observed outputs is identified, the system examines possible input variations and logical conditions that may have caused the fault. This feature enables users to better understand the nature of the fault and assists designers in rapid debugging and validation of digital circuits.

A complete software-based implementation of the proposed system is developed using Python for backend logic processing, while the Flask framework is utilized to create a web-based interactive user interface. The frontend allows users to select logic gate types, provide input combinations, and specify observed outputs, making the system accessible even to users with limited hardware design experience. This modular and platform-independent implementation ensures ease of deployment, low cost, and scalability for academic and experimental use.

Experimental evaluation demonstrates that the proposed approach accurately detects logic-level faults across a wide range of fundamental logic gates, including AND, OR, NOT, NAND, NOR, XOR, and XNOR configurations. The results indicate consistent and reliable fault identification while maintaining low computational overhead. Compared to conventional fault detection methodologies, the proposed system offers improved interpretability, reduced complexity, and enhanced user interaction.

How to cite this paper: Pranav Dhumane | Dr. N. S. Narawade | Dr. N. S. Kothari "AI-Inspired Fault Detection in VLSI Circuits Using Simulation Techniques" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-10 | Issue-3, June 2026, pp.42-59, URL: www.ijtsrd.com/papers/ijtsrd116442.pdf

IJTSRD116442

Overall, the presented AI-inspired fault detection framework provides a cost-effective, user-friendly, and educationally valuable solution for logic-level fault analysis in VLSI circuits. The system is particularly suitable for academic learning, design verification, and preliminary testing scenarios, and it serves as a strong foundation for future extensions involving sequential circuits, larger combinational networks, and advanced intelligent diagnostic techniques.

KEYWORDS: *VLSI fault detection, logic gate simulation, artificial intelligence, machine learning-inspired systems, digital circuits, fault diagnosis.*

I. INTRODUCTION

The rapid evolution of semiconductor fabrication technologies has enabled the realization of highly complex Very Large Scale Integration (VLSI) circuits containing millions of transistors on a single silicon die. This advancement has significantly improved computational performance, power efficiency, and functional density of modern digital systems. However, the continuous scaling of device dimensions and increasing circuit density have simultaneously introduced new reliability challenges. Manufacturing imperfections, design inconsistencies, environmental variations, and aging effects contribute to the growing occurrence of faults in VLSI circuits, making fault detection and diagnosis a critical aspect of contemporary digital system design.

As VLSI systems grow in complexity, ensuring functional correctness throughout the design and testing lifecycle has become increasingly difficult. Even minor logic-level deviations can propagate through interconnected circuit components and lead to incorrect outputs, degraded performance, or complete system malfunction. Therefore, reliable fault detection mechanisms are essential not only to identify faulty behavior but also to support efficient debugging and corrective analysis during early design and validation stages.

Conventional fault detection techniques employed in VLSI testing, such as manual inspection, rule-based verification, and exhaustive test vector generation, face significant scalability limitations. Manual testing relies heavily on human expertise and becomes impractical for large-scale circuits. Exhaustive testing approaches suffer from combinatorial explosion as the number of inputs increases, resulting in high computational overhead and extended testing time. Furthermore, traditional techniques primarily focus on detecting the existence of faults without providing insights into fault causes or corrective measures, thereby limiting their usefulness in educational and debugging environments.

In industrial practice, advanced Automatic Test Equipment (ATE) and Electronic Design Automation (EDA) tools are widely used to achieve high fault coverage and accurate diagnosis. Although these tools offer robust testing capabilities, they are often expensive, complex to operate, and require specialized training. As a result, access to such tools is restricted in academic institutions and small research setups, creating a gap between theoretical knowledge of digital logic and practical fault analysis experience.

Recent progress in artificial intelligence (AI) and machine learning (ML) has introduced new possibilities for intelligent fault detection and diagnosis in VLSI systems. AI-driven approaches can analyze circuit responses, recognize abnormal behavior patterns, and assist in decision-making processes. Several studies have demonstrated that machine learning models can achieve high fault classification accuracy. However, many of these approaches depend on large labeled datasets, extensive training procedures, and significant computational resources. Additionally, most ML-based systems operate as black-box models, offering limited interpretability, which is undesirable in instructional and learning-oriented applications. Interpretability and transparency are crucial factors in fault analysis, particularly for students and early-stage designers who must understand not only what fault has occurred but also why it occurred and how it can be corrected. A lack of explainability reduces the educational value of fault detection tools and complicates debugging processes.

To overcome these challenges, this paper presents an AI- inspired fault detection framework that integrates logic gate simulation with intelligent decision analysis. Unlike data- intensive machine learning models, the proposed approach relies on deterministic Boolean logic evaluation and systematic input-output comparison. By analyzing discrepancies between observed and expected logic gate outputs, the system accurately identifies faulty conditions at the logic level. Furthermore, the framework extends beyond fault detection by suggesting possible corrective input modifications, thereby assisting users in understanding fault behavior and resolution strategies.

Another important contribution of this work is the development of a web-based interactive platform that enhances accessibility and usability. The system provides a graphical interface through which users can select logic gates, apply different input

combinations, and observe circuit responses in real time. This interactive design supports experiential learning and simplifies fault analysis for users with varying levels of expertise. The implementation using widely adopted technologies such as Python and the Flask framework ensures portability, scalability, and ease of deployment.

The proposed fault detection system emphasizes simplicity, transparency, and computational efficiency. By operating at the logic gate level, it effectively bridges the gap between theoretical digital logic concepts and practical fault detection mechanisms. Unlike conventional approaches that merely re- port faults, the system delivers meaningful diagnostic insights and corrective guidance, enabling faster debugging and im- proved comprehension of circuit behavior.

In summary, the increasing complexity of modern VLSI circuits necessitates intelligent, automated, and accessible fault detection solutions. The AI-inspired framework presented in this paper offers a cost-effective and interpretable alternative to traditional methodologies. Its ability to detect logic-level faults, recommend corrective actions, and provide an intuitive user interface makes it a valuable tool for VLSI education, aca- demic research, and preliminary circuit testing. Additionally, the framework enhances hands-on learning by allowing users to experiment with logic gates and input scenarios, thereby deepening their understanding of fault mechanisms and digital circuit reliability.

The modular design of the system allows easy extensibility and future enhancements. Additional logic blocks, fault models, and analysis techniques can be incorporated without significant modification to the core architecture. This flexibility makes the proposed solution adaptable to evolving research requirements and technological advancements. Moreover, the software-only implementation eliminates hardware dependency, reducing cost and complexity while ensuring platform independence.

In contrast to traditional verification environments that require complex setup procedures, the proposed web-based system offers instant accessibility through a standard browser. This characteristic significantly lowers the entry barrier for users, allowing rapid experimentation and evaluation of logic- level fault scenarios. Such accessibility is particularly beneficial in academic laboratories, where resources and infrastructure may be limited.

Overall, the proposed work highlights the importance of integrating intelligent fault detection techniques with user - centric design. By focusing on interpretability, accessibility, and accuracy, the system aligns well with modern trends in VLSI research that emphasize explainable and scalable solutions. The combination of logic simulation, intelligent decision-making, and web-based deployment establishes a ro- bust foundation for future research in automated fault detection and digital circuit analysis.

II. MOTIVATION AND BACKGROUND

The continuous scaling of Complementary Metal-Oxide-Semiconductor (CMOS) technology has enabled the design of highly dense and complex Very Large Scale Integration (VLSI) circuits. Modern digital systems integrate an enormous number of transistors on a single chip to achieve high performance, low power consumption, and enhanced functionality. However, this rapid technological advancement has also increased the vulnerability of VLSI circuits to a wide range of faults. Manufacturing imperfections, process variations, environmental disturbances, aging effects, and input anomalies contribute to the growing occurrence of logic-level faults in digital circuits. Even small deviations in logic behavior can propagate through interconnected components and lead to incorrect outputs or complete system failure.

As circuit complexity increases, fault detection has become a critical challenge in ensuring reliable system operation. Logic-level faults, such as stuck-at faults, transient faults, and incorrect input conditions, directly affect the functional correctness of digital circuits. Unlike physical faults that may be isolated at the transistor level, logic-level faults often propagate across multiple stages of the circuit, making their detection and diagnosis more difficult. Therefore, effective and efficient fault detection mechanisms are essential to identify faulty behavior at an early stage and prevent error propagation in large-scale systems.

Traditional fault detection approaches have relied heavily on manual inspection, rule-based verification, and exhaustive test vector generation. While these methods were effective for small-scale circuits, they are no longer practical for modern VLSI designs. Manual testing is time-consuming and highly dependent on human expertise, making it unsuitable for complex systems. Exhaustive testing techniques suffer from combinatorial explosion as the number of inputs increases, resulting in excessive computational overhead and long testing times. Moreover, conventional approaches primarily focus on detecting whether a fault exists, without providing sufficient insight into fault causes or corrective measures.

In industrial environments, advanced Automatic Test Equipment (ATE) and commercial Electronic Design

Automation (EDA) tools are commonly used for fault detection and diagnosis. Although these tools offer high accuracy and fault coverage, they are often expensive and require specialized hardware, proprietary software licenses, and trained personnel to operate. As a result, their accessibility is limited, particularly in academic institutions, small research laboratories, and early stage development environments. This creates a significant gap between theoretical digital logic concepts taught in classrooms and practical fault analysis techniques used in real-world VLSI testing.

Another important limitation of existing fault detection methodologies is the lack of interpretability and user guidance. Many traditional and modern approaches simply indicate the presence of a fault without explaining why the fault occurred or how it can be corrected. In practical debugging scenarios, fault localization and corrective suggestions are as important as fault identification itself. The absence of intelligent diagnostic support increases debugging time, reduces productivity, and limits the learning value of fault detection tools.

Recent advancements in artificial intelligence (AI) and machine learning (ML) have introduced new paradigms for fault detection and diagnosis in VLSI circuits. AI-based approaches are capable of analyzing circuit behavior patterns, identifying abnormal responses, and assisting in decision-making processes. Several studies have demonstrated that machine learning models can achieve high accuracy in fault classification and prediction. However, many AI-driven solutions require large labeled datasets, extensive training procedures, and significant computational resources. These requirements make such systems unsuitable for lightweight applications and educational environments.

Furthermore, most machine learning-based fault detection systems operate as black-box models, offering limited transparency and explainability. Users are often unable to understand how a particular fault decision was made or which input conditions contributed to the detected error. This lack of interpretability reduces trust in the system and limits its usefulness in instructional and learning-oriented settings, where understanding fault behavior is essential.

The motivation behind this research is to address these limitations by developing a fault detection framework that is intelligent, interpretable, and accessible. The proposed work aims to bridge the gap between theoretical digital logic knowledge and practical fault diagnosis by operating at the logic gate level. By using deterministic Boolean logic evaluation instead of data-intensive training models, the system ensures transparency, simplicity, and computational efficiency. This approach allows users to clearly understand the relationship between inputs, logic operations, and observed outputs.

Another key motivation is to provide corrective guidance in addition to fault detection. In real-world debugging scenarios, identifying a fault alone is insufficient. Designers and students benefit significantly from suggestions that indicate how the fault can be corrected or mitigated. By systematically analyzing input variations and comparing observed outputs with expected logic behavior, the proposed system is capable of suggesting possible corrective actions. This feature enhances both debugging efficiency and conceptual understanding of fault mechanisms.

Educational value is a major driving factor behind this work. Interactive and simulation-based learning environments have been shown to significantly improve comprehension of complex engineering concepts. By allowing users to select logic gates, apply different input combinations, and observe circuit behavior in real time, the proposed system provides hands-on exposure to fundamental digital logic principles. This experiential learning approach helps users visualize fault propagation, understand logical correctness, and develop effective debugging strategies.

The integration of a web-based interface further enhances the accessibility and usability of the system. A graphical user interface allows users to interact with the fault detection framework without requiring prior expertise in VLSI testing tools or programming environments. The use of widely adopted technologies such as Python and the Flask framework ensures portability, scalability, and ease of deployment across different platforms.

Overall, the motivation of this research is to develop a cost-effective, software-based, and user-friendly fault detection framework that balances simplicity with accuracy. The proposed AI-inspired approach not only detects logic-level faults but also provides meaningful diagnostic insights and corrective suggestions. By combining logic gate simulation, intelligent decision analysis, and interactive visualization, the system aims to support both academic learning and preliminary VLSI testing applications. This work contributes toward making fault detection more accessible, interpretable, and effective for students, researchers, and practicing engineers.

III. LITERATURE REVIEW

Fault detection and diagnosis in VLSI circuits have been extensively studied due to the critical role of integrated circuits in modern electronic systems. As semiconductor technology advances toward nanometer-scale fabrication, circuits are increasingly vulnerable to manufacturing defects, environmental disturbances, and aging-related failures. Consequently, ensuring reliability and correctness of digital circuits has become a major research concern. Over the years, researchers have proposed a wide range of fault detection techniques, including traditional rule-based testing, simulation-driven analysis, and more recently, artificial intelligence-based approaches.

A. Rule-Based and Deterministic Fault Detection Methods

Early fault detection techniques were primarily rule-based and relied on predefined test vectors and Boolean logic evaluation. In [1], the authors presented a deterministic fault detection methodology using manually generated test patterns for identifying faults in combinational logic circuits. Although this approach provided reliable detection for simple circuits, its effectiveness decreased significantly as circuit complexity increased. The dependency on manually crafted rules made the system inflexible and difficult to scale.

Another widely adopted deterministic approach is Automatic Test Pattern Generation (ATPG). In [2], ATPG techniques were used to generate optimized test vectors for detecting stuck-at faults in digital circuits. While ATPG offers high fault coverage and systematic testing, it involves complex modeling and high computational overhead. Furthermore, ATPG-based methods are primarily designed for manufacturing-time testing and are not suitable for interactive or real-time fault analysis environments.

Deterministic techniques generally focus on fault detection rather than fault diagnosis. They identify whether a fault exists but provide limited insight into fault localization or correction. This limitation reduces their usefulness in educational applications and debugging scenarios where understanding fault behavior is equally important as detecting it.

B. Simulation-Based Fault Analysis Techniques

Simulation-based approaches have gained popularity as an alternative to hardware-intensive testing methods. These techniques model the logical behavior of circuits using software simulations and analyze circuit responses under different input conditions. Roth [3] introduced fault simulation frameworks that compare fault-free and faulty circuit outputs to identify discrepancies. Such methods reduce the need for physical hardware and allow designers to evaluate multiple fault scenarios efficiently.

Logic-level simulation tools are commonly used in academic environments to demonstrate the functionality of digital circuits. However, most simulation-based tools focus on functional correctness rather than intelligent fault diagnosis. Users are typically required to manually interpret mismatches between expected and observed outputs, which can be challenging for beginners. Additionally, existing simulators do not provide automated corrective suggestions, limiting their practical applicability.

The proposed system enhances simulation-based fault analysis by integrating intelligent decision logic that automatically identifies faults and suggests corrective actions, thereby improving user understanding and reducing manual effort.

C. Machine Learning-Based Fault Detection Approaches

With the growth of artificial intelligence, machine learning (ML) techniques have been increasingly applied to VLSI fault detection and diagnosis. In [4], the authors proposed a neural network-based fault classification system trained using simulated fault datasets. The model demonstrated improved fault detection accuracy compared to traditional rule-based approaches. However, the performance of the system heavily depended on the availability of large and diverse training datasets.

Support Vector Machine (SVM) based fault detection frameworks were introduced in [5]. These approaches achieved good classification accuracy and robustness against noise. Nevertheless, SVM-based methods require careful feature extraction and parameter tuning, making them complex to implement and interpret.

More recent studies have explored deep learning techniques such as convolutional neural networks (CNNs) for fault diagnosis. In [6], CNNs were used to analyze circuit behavior patterns and detect complex fault conditions. Although deep learning models offer high accuracy, they suffer from several drawbacks, including high computational cost, lack of interpretability, and difficulty in deployment for lightweight or realtime applications.

Furthermore, ML-based systems often behave as black boxes, making it difficult for users to understand how fault decisions are made. This lack of transparency is undesirable in academic and instructional environments where explainability is crucial.

D. Hybrid and AI-Inspired Fault Detection Systems

To overcome the limitations of purely data-driven ML models, several researchers have explored hybrid and AI inspired approaches that combine logical reasoning with intelligent decision-making. These systems aim to provide the benefits of intelligence while maintaining interpretability and low computational complexity.

In [7], a hybrid fault detection framework was proposed that used logic evaluation combined with heuristic rules to identify fault conditions. Although effective, the system was limited to specific fault models and lacked extensibility. Other studies have focused on expert systems for fault diagnosis; however, such systems require extensive domain knowledge encoding and are difficult to maintain.

The proposed work follows an AI-inspired logic-based approach that does not rely on training data or complex models. Instead, it evaluates expected outputs using deterministic logic functions and applies intelligent decision logic to identify mismatches and suggest corrective actions. This approach balances accuracy, simplicity, and explainability.

E. Web-Based Fault Detection and Educational Tools

Recent advancements in web technologies have enabled the development of browser-based simulation and testing platforms. In [8], a web-based digital logic simulator was developed to assist students in understanding basic logic gate operations. While the platform improved accessibility, it did not support fault detection or diagnosis.

Web-based fault detection systems offer several advantages, including platform independence, ease of deployment, and user-friendly interfaces. However, existing tools primarily focus on circuit design rather than fault analysis. The proposed system addresses this gap by providing a web-based interface that allows users to interactively test logic gates, observe results, and understand fault behavior in real time.

F. Limitations of Existing Research

Despite extensive research, existing fault detection methods suffer from several limitations. Traditional deterministic techniques lack adaptability and scalability. Simulation-based approaches require manual interpretation of results. Machine learning-based systems demand large datasets and computational resources, making them unsuitable for lightweight applications.

Additionally, most existing systems focus solely on fault identification and do not provide corrective guidance. There is a clear lack of systems that combine fault detection, diagnosis, and correction in a single, interpretable framework.

G. Research Gap and Contribution

Based on the literature survey, it is evident that there is a need for a lightweight, intelligent, and user-friendly fault detection system that operates at the logic-gate level. Existing approaches either emphasize accuracy at the cost of complexity or provide basic simulation without diagnostic intelligence.

The proposed AI-based fault detection system addresses this research gap by integrating logic simulation with intelligent decision-making in a web-based environment. Unlike conventional ML approaches, the system does not require training datasets, ensuring low computational overhead and high interpretability. This makes the proposed solution particularly suitable for educational, academic, and preliminary VLSI testing applications.

H. Fault Models in Digital and VLSI Circuits

Fault modeling plays a crucial role in the development of effective fault detection and diagnosis techniques. Over the years, several fault models have been proposed to represent real-world defects occurring in VLSI circuits. Among them, stuck-at faults are the most widely studied and commonly used fault model due to their simplicity and effectiveness. A stuck at fault assumes that a signal line is permanently fixed at logic '0' or logic '1', regardless of the intended circuit behavior.

In [9], the authors analyzed the effectiveness of stuck-at fault models in representing manufacturing defects. The study concluded that although stuck-at faults do not capture all physical defects, they provide a good abstraction for logic level testing and remain relevant for both academic and industrial applications. However, detecting such faults requires systematic evaluation of circuit outputs under various input conditions, which can be challenging without automated tools. Bridging faults and delay faults have also been explored in the literature. Bridging faults occur when two signal lines are unintentionally connected, while delay faults result from timing variations in signal propagation. In [10], delay fault detection techniques were discussed using timing-aware simulation methods. These approaches improve fault coverage but introduce additional complexity in terms of timing analysis and simulation overhead.

The proposed system primarily focuses on logic-level functional faults, making it suitable for early-stage testing, educational demonstrations, and functional verification. By concentrating on fundamental logic behavior, the system avoids the complexity associated

with low-level physical fault modeling while still providing meaningful fault detection and diagnosis.

I. Fault Diagnosis and Correction-Oriented Approaches

While fault detection determines the presence of a fault, fault diagnosis aims to identify the cause and location of the fault. Several researchers have highlighted the importance of fault diagnosis in reducing debugging time and improving system reliability. In [11], a diagnosis-oriented framework was proposed that analyzed output mismatches to trace faulty inputs or components. Although effective, the framework required extensive rule sets and domain-specific knowledge.

Correction-oriented fault analysis has received comparatively less attention in existing research. Most fault detection systems stop at reporting the presence of a fault, leaving corrective actions to the user. This limitation is particularly evident in educational tools, where learners benefit significantly from guided correction and explanation.

The proposed AI-based fault detection system addresses this limitation by incorporating intelligent correction suggestions. By systematically flipping input values and re-evaluating gate outputs, the system identifies possible corrective actions that align observed outputs with expected logic behavior. This approach enhances user understanding and provides actionable feedback, which is rarely offered by conventional fault detection tools.

J. Educational and Learning-Oriented Fault Detection Systems

With the increasing emphasis on experiential learning, several studies have explored the use of simulation tools for teaching digital logic and VLSI concepts. In [12], an interactive logic simulator was introduced to help students visualize gate operations and Boolean expressions. Although effective for concept learning, the simulator lacked fault analysis capabilities.

Educational tools must balance simplicity, accuracy, and interactivity. Overly complex tools can overwhelm beginners, while overly simplistic tools may fail to convey real-world relevance. The proposed system strikes this balance by providing a clean web-based interface combined with intelligent fault analysis. Users can experiment with different logic gates, inputs, and outputs, observe system behavior, and understand fault conditions without requiring prior expertise in VLSI testing methodologies.

K. Summary of Literature Review

The literature survey highlights that fault detection in VLSI circuits has evolved from rule-based testing to simulation driven and AI-assisted approaches. While machine learning based systems offer high accuracy, they introduce complexity, data dependency, and limited interpretability. Traditional deterministic methods, on the other hand, lack intelligence and user guidance.

The proposed system differentiates itself by adopting an AI inspired logic-based approach that is lightweight, interpretable, and correction-oriented. By integrating logic simulation, intelligent decision-making, and a web-based interface, the system effectively addresses the limitations identified in existing research. This makes the proposed work a valuable contribution to both academic learning and preliminary VLSI fault analysis.

IV. PROBLEM DEFINITION

Although a wide range of fault detection techniques have been proposed over the years, several fundamental challenges remain unresolved in modern VLSI testing environments. Most conventional fault detection approaches are predominantly rule-based and lack the flexibility required to handle increasing circuit complexity. These methods are generally capable of identifying the presence of faults, but they often fail to provide meaningful insights into fault origins or suggest corrective measures for improving circuit functionality.

The primary issues addressed in this work include:

- Absence of intelligent fault diagnosis at the logic gate level
- Lack of automated fault correction and recovery suggestions
- High implementation cost and operational complexity of existing tools
- Limited accessibility for academic learning and research applications

The objective of this research is to design and implement an automated, intelligent, and user-friendly fault detection framework that can accurately identify faulty logic behavior in digital VLSI circuits while also offering corrective guidance.

A. Problem Definition and Challenges

The rapid growth in the scale and complexity of VLSI systems has significantly increased the difficulty of ensuring reliable circuit operation. Contemporary digital systems integrate millions of logic gates operating at high frequencies, where even a small deviation in logic behavior can propagate through multiple stages and lead to erroneous outputs or system malfunction. Traditional fault detection strategies are often applied after fabrication or during testing phases, limiting their ability to provide early-stage insight into fault causation.

A major limitation of existing fault detection techniques is their strong dependence on predefined test vectors and static verification rules. Designing exhaustive test patterns for complex logic circuits requires considerable manual effort and domain expertise. Furthermore, these approaches lack adaptability and often fail to respond effectively to variations in gate configurations or input combinations. In most cases, such methods merely report the presence of a fault without offering explanations regarding its nature or possible resolution.

Another critical challenge lies in the lack of interactive and user-oriented diagnostic tools. Many fault detection systems are developed primarily for industrial-scale deployment and are not tailored for educational or research environments. As a result, students and early-stage researchers face difficulty in visualizing fault propagation and understanding the relationship between input conditions and output behavior. This gap reduces conceptual clarity and limits the effectiveness of fault analysis as a learning tool.

Additionally, hardware-dependent fault detection solutions require costly testing equipment, specialized infrastructure, and complex configuration procedures. These requirements make such systems impractical for small laboratories, academic institutions, and early design validation stages. There is a strong demand for a low-cost, software-based alternative that can simulate logic behavior accurately and provide immediate feedback on fault conditions.

From a computational perspective, several modern fault detection methods rely on machine learning models trained on large datasets. While these techniques may achieve high detection accuracy, they often suffer from limited interpretability and require substantial computational resources. The reliance on training data also makes such systems sensitive to dataset quality and reduces their ability to generalize to previously unseen logic configurations. These drawbacks limit their suitability for transparent analysis and academic validation.

Therefore, the central problem addressed in this research is the lack of an intelligent, interpretable, and software-driven fault detection system operating directly at the logic gate level. An effective solution should not only identify the existence of faults but also assist users in understanding potential fault causes and recommending corrective actions. Moreover, the system should be simple to operate, platform-independent, and suitable for both educational and research-oriented use cases. This work seeks to bridge the gap between traditional rule-based fault detection methods and complex AI-driven techniques by introducing an AI-inspired logic evaluation framework. The proposed approach emphasizes deterministic fault detection combined with intelligent decision-making, ensuring accurate fault identification while preserving transparency, simplicity, and ease of understanding.

V. PROPOSED METHODOLOGY

This section presents the proposed AI-based fault detection methodology developed for identifying and analyzing logic level faults in digital VLSI circuits. The methodology is entirely software-driven and emphasizes deterministic logic simulation, intelligent fault detection, and corrective suggestion mechanisms. The objective is to achieve accurate, interpretable, and low-cost fault diagnosis without relying on hardware implementation or data-intensive machine learning models. The overall system is designed to be modular, transparent, and suitable for both academic and practical fault analysis environments.

A. Overview of the Proposed System

The proposed system is a web-based fault detection and diagnosis framework that simulates the functional behavior of basic digital logic gates and identifies faults by comparing observed outputs with expected logical outputs. Users interact with the system by selecting a logic gate, providing corresponding input values, and entering the observed output of the circuit under test.

Unlike traditional fault detection techniques that rely on predefined test vectors or hardware-intensive testing environments, the proposed methodology operates entirely through software-based logic simulation. An AI-inspired logic evaluation approach is adopted, where deterministic Boolean computation is combined with intelligent comparison and decision making to detect abnormal circuit behavior efficiently and transparently.

The system accepts the following primary inputs:

- Selected logic gate type (AND, OR, NOT, NAND, NOR, XOR, XNOR)
- Input values corresponding to the selected gate
- Observed output provided by the user

Based on these inputs, the system computes the expected output using predefined Boolean expressions and performs fault analysis by comparing it with the observed output.

B. Logic Gate Modeling

Each logic gate is modeled using deterministic Boolean logic functions implemented in Python. This modeling ensures accurate representation of ideal gate behavior under all valid input combinations. The

software-based modeling approach eliminates uncertainties associated with physical variations and focuses purely on functional correctness at the logic level.

For instance, the AND gate output is computed as the logical conjunction of its inputs, while the XOR gate output represents the exclusive logical difference between inputs. The NOT gate is treated as a unary operation, and the system dynamically adapts the number of required inputs based on the selected gate type.

This modular gate modeling strategy ensures scalability and allows future extension of the system to support more complex combinational or sequential logic structures.

C. System Workflow and Execution Flow

The execution flow of the proposed system follows a structured and sequential process to ensure systematic fault detection and user-friendly operation. The workflow is summarized as follows:

1. The user selects a desired logic gate through the webbased interface.
2. The user enters the corresponding input values and the observed output.
3. The system computes the expected output using Boolean logic expressions associated with the selected gate.
4. A comparison is performed between the expected output and the observed output.
5. If both outputs match, the circuit is classified as operating under normal conditions.
6. If a mismatch is detected, the system identifies the presence of a logic-level fault.
7. Corrective analysis is initiated, and possible corrective input combinations are generated.
8. The final fault status and corrective suggestions are displayed to the user.

This structured execution flow ensures consistency, reliability, and ease of understanding for both beginners and advanced users.

D. Logic Evaluation and Fault Detection Mechanism

At the core of the proposed methodology lies the logic evaluation and fault detection mechanism. The system evaluates the expected output by applying deterministic Boolean expressions corresponding to the selected logic gate and input values.

Fault detection is performed by comparing the evaluated output with the observed output entered by the user. A mismatch between these two values indicates abnormal circuit behavior and is classified as a logic-level functional fault. The methodology does not assume any specific physical fault model such as stuck-at or bridging faults; instead, it focuses on identifying deviations from ideal logical behavior.

Mathematically, the fault detection condition is defined as:

$$Fault = \begin{cases} Normal, & \text{if } Y_{observed} = Y_{calculated} \\ Faulty, & \text{if } Y_{observed} \neq Y_{calculated} \end{cases}$$

This comparison-based approach provides a simple yet reliable mechanism for detecting logic-level faults.

E. Fault Diagnosis and Corrective Suggestion Strategy

Unlike conventional fault detection systems that only indicate the presence of a fault, the proposed methodology incorporates an intelligent fault diagnosis and correction suggestion mechanism. Once a fault is detected, the system systematically analyzes possible input variations to identify configurations that could produce the observed output.

For multi-input logic gates, the system iteratively modifies individual input values and recalculates the corresponding output. If the recalculated output matches the observed output, the associated input combination is identified as a potential corrective suggestion. In the case of unary gates such as NOT, the system directly recommends re-evaluation of the input value.

This correction-oriented approach provides actionable feedback, enhances interpretability, and assists users in understanding how input variations influence circuit behavior. It is particularly beneficial in educational, debugging, and learning oriented environments.

F. Web-Based Implementation Framework

The proposed system is implemented using a lightweight web-based architecture to ensure accessibility and ease of use. The backend is developed using Python and the Flask framework, which handles logic gate simulation, fault detection, and correction analysis. The frontend interface is implemented using HTML and CSS to provide a clean, intuitive, and interactive user experience.

The separation of computation and presentation layers improves modularity, scalability, and maintainability. Additionally, the web-based framework allows users to access the system through standard web browsers without requiring specialized software installations. Dynamic input handling is incorporated to adjust input fields based on the selected logic gate, ensuring correctness and preventing invalid input combinations.

G. Advantages of the Proposed Methodology

The proposed methodology offers several advantages over traditional fault detection techniques:

- Software-only implementation eliminates dependency on costly hardware tools.
- Deterministic logic evaluation ensures transparency and interpretability.
- Intelligent fault diagnosis and correction suggestions improve debugging efficiency.
- Web-based interface enhances accessibility and user engagement.
- Suitable for academic learning, preliminary VLSI testing, and digital logic education.

H. Summary of Methodology

In summary, the proposed methodology integrates logic gate modeling, deterministic fault detection, intelligent correction suggestion, and web-based deployment into a unified framework. By combining rule-based logic evaluation with AI-inspired decision-making, the system effectively addresses the limitations of conventional fault detection approaches while maintaining simplicity, accuracy, and interpretability.

VI. SYSTEM ARCHITECTURE

This section presents the architecture of the proposed AI based fault detection system for VLSI logic circuits. The system architecture is designed to provide a structured, modular, and efficient framework for logic simulation, fault detection, and result visualization using a web-based platform. The system architecture of the proposed AI-inspired fault detection framework is illustrated in Fig. 1. The architecture follows a modular and layered design, beginning with a web based user interface that allows users to select logic gates, apply input combinations, and provide observed outputs. The Flask controller acts as an intermediary between the frontend and backend modules. Logic gate behavior is modeled using Boolean expressions, and the expected output is generated accordingly. A comparator engine evaluates mismatches between expected and observed outputs to identify faults. Upon fault detection, a correction suggestion engine analyzes alternative input combinations and provides actionable feedback to the user. This architecture ensures interpretability, scalability, and efficient fault analysis in a software-only environment.

A. System Architecture

The system architecture of the proposed AI-based fault detection framework is designed to provide a clear separation between user interaction, logic processing, fault analysis, and result visualization. The architecture follows a layered and modular design approach, ensuring scalability, interpretability, and ease of maintenance. Figure 1 illustrates the complete architectural workflow of the proposed system.

The system operates entirely in a software-based environment and focuses on logic-level fault detection without requiring any physical hardware or training datasets. Each architectural component performs a well-defined function, enabling accurate fault detection and intelligent corrective guidance.

B. User Input Module

The User Input module acts as the entry point of the system. It allows users to provide essential information required for fault analysis. The inputs include the selection of the logic gate type (AND, OR, NOT, NAND, NOR, XOR, or XNOR),

binary input values corresponding to the selected gate, and the observed output obtained from the circuit or simulation.

This module is designed to support both educational users and researchers by enabling flexible experimentation with different logic configurations. The clarity of input specification ensures that the fault detection process is accurate and free from ambiguity.

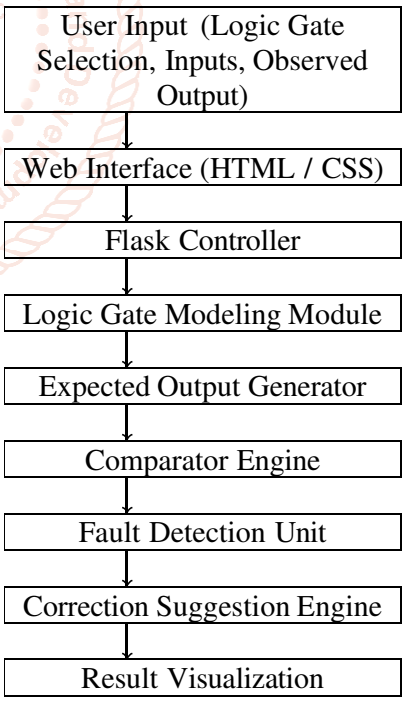

Fig. 1. System Architecture of the Proposed AI-Based Fault Detection Framework

C. Web Interface Layer

The Web Interface layer is responsible for facilitating seamless interaction between the user and the system. Developed using standard web technologies such as HTML and CSS, this layer provides a clean and intuitive graphical interface.

The interface dynamically adjusts input fields based on the selected logic gate. For example, when a NOT gate is selected, the system automatically disables the second input field. This intelligent input validation prevents invalid configurations and enhances usability.

D. Flask Controller

The Flask Controller serves as the central coordination unit of the architecture. It acts as an intermediary between the frontend interface and the backend processing modules. Upon receiving user inputs from the web interface, the controller validates the data and routes it to the appropriate logic processing modules.

The controller also manages request handling, response generation, and error handling. By separating control logic from computation logic, the architecture achieves modularity and simplifies future system enhancements.

E. Logic Gate Modeling Module

The Logic Gate Modeling module is responsible for accurately simulating the behavior of digital logic gates using deterministic Boolean expressions. Each logic gate is modeled using mathematically defined logic functions implemented in Python.

This module ensures ideal logic behavior under all valid input combinations and serves as the foundation for expected output computation. The use of deterministic modeling eliminates uncertainty and ensures that the fault detection process is transparent and interpretable.

F. Expected Output Generator

The Expected Output Generator computes the correct output corresponding to the selected logic gate and provided inputs. Using the Boolean models defined in the logic modeling module, the system calculates the expected output under faultfree conditions.

This computed output represents the reference behavior of the circuit and is later used for comparison with the observed output provided by the user. The accuracy of this module directly influences the reliability of fault detection.

G. Comparator Engine

The Comparator Engine performs a critical role in identifying discrepancies between the observed output and the expected output. It compares the two outputs at the logic level and determines whether the circuit is operating correctly.

If both outputs match, the system classifies the circuit as fault-free. If a mismatch is detected, the engine flags the presence of a fault and forwards the result to the fault detection unit. This comparison-based approach ensures reliable detection of functional logic faults.

H. Fault Detection Unit

The Fault Detection Unit analyzes the output of the comparator engine and determines the fault status of the circuit. Unlike traditional systems that rely on predefined fault models, this unit focuses on functional correctness at the logic-gate level.

By avoiding assumptions about specific fault types, the system remains flexible and capable of identifying a wide range of logical inconsistencies. This design choice makes the system suitable for educational demonstrations and early-stage VLSI verification.

I. Correction Suggestion Engine

One of the distinguishing features of the proposed architecture is the Correction Suggestion Engine. When a fault is detected, this module attempts to identify possible corrective actions by systematically evaluating alternative input combinations.

For multi-input gates, the engine toggles individual input values and recalculates outputs to determine combinations that match the observed output. The identified combinations are then presented as corrective suggestions to the user. This mechanism transforms fault detection into a learning-oriented and diagnosis-driven process.

J. Result Visualization Module

The Result Visualization module presents the final outcome of the fault analysis to the user in a clear and interpretable manner. The displayed results include the fault status (Normal or Faulty), expected output, observed output, and corrective suggestions if applicable.

This module ensures that users can easily understand the system response without requiring in-depth knowledge of VLSI testing methodologies. The visualization enhances transparency and improves the overall user experience.

K. Architectural Advantages

The proposed architecture offers several advantages: Modular design enabling easy scalability and maintenance Complete separation of frontend, control, and logic processing layers

Software-only implementation without hardware dependency

Interpretable and deterministic fault detection mechanism Support for intelligent fault correction and learning

The architecture effectively supports the objectives of accurate fault detection, explainability, and usability, making it suitable for academic, educational, and preliminary VLSI testing applications.

VII. FAULT DETECTION ALGORITHM

This section describes the algorithm used in the proposed AI-based fault detection system for VLSI logic circuits. The algorithm follows a deterministic and logic-driven approach to identify faults in basic digital logic gates and to suggest corrective input combinations when a fault is detected.

The algorithm is designed to be simple, interpretable, and efficient, making it suitable for academic and simulation-based environments without requiring complex machine learning training models.

A. Algorithm Description

The fault detection process begins with user input, where the logic gate type, input values, and observed output are provided through the web interface. Based on the selected logic gate, the algorithm computes the expected output using predefined Boolean logic expressions.

The observed output is then compared with the calculated output. If both values match, the circuit is classified as fault-free. If a mismatch occurs, the algorithm identifies the presence of a fault and initiates a correction mechanism to suggest valid input combinations that can restore correct output behavior.

B. Step-by-Step Algorithm

The detailed steps of the proposed fault detection algorithm are as follows:

1. Read the logic gate type selected by the user.
2. Accept input values and observed output.
3. Compute the expected output using Boolean logic equations.
4. Compare the expected output with the observed output.
5. If the outputs match, declare the circuit as *Normal*.
6. If the outputs do not match, declare a *Fault Detected*.
7. Generate alternative input combinations.
8. Identify input combinations that produce correct output.
9. Display fault status and corrective suggestions to the user.

C. Pseudo-Code Representation

The simplified pseudo-code of the proposed logic gate fault detection algorithm is presented below.

Algorithm 1 Logic Gate Fault Detection Algorithm

```
Select logic gate G
Read inputs (A, B) and observed output Y_obs
Compute expected output Y_calc based on G
if Y_obs = Y_calc then
  Declare Normal Circuit
else
Declare Fault Detected
for (A, B) ∈ {0, 1} × {0, 1} do
      Recompute output Y
      if Y = Y_obs then
            Suggest corrective input combination
      end if
  end for
end if
```

D. Algorithm Characteristics

The key characteristics of the proposed algorithm are summarized below:

- Deterministic and logic-based approach
- No requirement for training datasets
- High interpretability and transparency
- Suitable for real-time fault analysis

The proposed algorithm efficiently detects logic-level faults and provides corrective suggestions, making it effective for VLSI fault analysis, academic demonstrations, and educational laboratories.

E. Proposed Algorithm Explanation

The proposed algorithm is designed to perform intelligent fault detection in basic digital logic gates by comparing the observed output with the expected logical output. The algorithm operates in a deterministic manner and does not rely on pre-trained datasets, making it lightweight, interpretable, and suitable for academic and educational environments.

Initially, the user selects the type of logic gate to be analyzed, such as AND, OR, NAND, NOR, XOR, XNOR, or NOT. The corresponding input values are then provided along with the observed output of the circuit under test. Based on the selected logic gate, the algorithm computes the expected output using predefined Boolean logic expressions.

Once the expected output is calculated, a comparison is performed between the observed output and the computed output. If both values match, the circuit is classified as operating under normal conditions, and the system reports a healthy circuit status. This step verifies the correctness of the logical behavior of the gate.

In cases where the observed output does not match the calculated output, the algorithm identifies the presence of a fault. To assist in fault correction, the algorithm iterates through all possible valid input combinations for the selected logic gate. For each combination, the expected output is recalculated and compared with the observed output.

When a matching condition is found, the algorithm suggests the corresponding input combination as a possible corrective action. This intelligent suggestion mechanism helps users understand the nature of the fault and provides guidance for correcting incorrect input conditions.

The proposed algorithm ensures accurate logic-level fault detection while maintaining simplicity and low computational complexity. Its rule-based decision-making approach makes it transparent and easy to interpret, which is particularly beneficial for VLSI education, digital circuit debugging, and simulation-based testing applications.

VIII. MATHEMATICAL MODELING

Mathematical modeling serves as the analytical backbone of the proposed AI-inspired fault detection framework. The system utilizes formal Boolean logic to describe the functional behavior of basic digital logic gates. This deterministic modeling strategy enables precise computation of expected outputs and systematic identification of logic-level faults. Unlike datadriven approaches, the proposed model avoids statistical assumptions and training dependencies, ensuring consistent and explainable fault analysis.

A. Boolean Logic Representation

Let the binary input variables be represented as A and B, where each input can assume a logical value of either 0 or 1:

$$A, B \in \{0, 1\}$$

Each logic gate is mathematically modeled using its corresponding Boolean expression that defines ideal gate behavior:

- **AND Gate:** $Y_{AND} = A \cdot B$
- **OR Gate:** $Y_{OR} = A + B$
- **NOT Gate:** $Y_{NOT} = A$
- **NAND Gate:** $Y_{NAND} = A$
- **NOR Gate:** $B\ Y_{NOR} = A + B$
- **XOR Gate:** $Y_{XOR} = A \oplus B$
- **XNOR Gate:** $Y_{XNOR} = A \oplus B$

These Boolean formulations accurately capture the functional characteristics of logic gates under normal operating conditions and form the basis for fault evaluation.

B. Fault Detection Model

Fault detection is achieved by comparing the actual output observed during circuit operation with the output computed using the Boolean logic model. Let Y_{obs} denote the observed output supplied by the user, and let Y_{exp} represent the expected output generated by the system. The fault status is determined as follows:

$$\text{Fault Condition} = \begin{cases} \text{Normal}, & \text{if } Y_{obs} = Y_{exp} \\ \text{Faulty}, & \text{if } Y_{obs} = Y_{exp} \end{cases}$$

Any deviation between the observed and expected outputs indicates the presence of a functional fault at the logic gate level.

C. Correction Analysis Using Input Evaluation

To support effective fault diagnosis, the system incorporates a correction analysis mechanism that evaluates alternative input scenarios. For a logic gate with two inputs, all possible binary input combinations are systematically examined. Each possible pair of input values is applied to the Boolean model, and the corresponding output is recalculated.

If the computed output for a particular input combination satisfies the condition:

$$Y_{calc} = Y_{ops}$$

then that input configuration is identified as a feasible corrective input and is presented to the user as a suggestion. This process assists users in understanding how specific input conditions influence output behavior and provides practical guidance for fault correction.

The input evaluation strategy enhances the overall debugging capability of the system by transforming fault detection into an actionable diagnostic process.

D. Model Interpretation and Significance

The proposed mathematical framework offers the following advantages:

- Precise and deterministic modeling of logic gate behavior
- Accurate identification of functional faults at the logic level
- Clear and interpretable fault diagnosis without black-box processing
- Independence from probabilistic assumptions and machine learning models

By relying on formal Boolean logic, the system ensures transparent and repeatable fault analysis. This makes the proposed approach well suited for VLSI simulation, educational experimentation, and early-stage digital circuit verification.

IX. IMPLEMENTATION DETAILS

The proposed fault detection framework is realized as a fully software-driven simulation platform developed using Python in combination with the Flask web framework. The implementation follows a modular design philosophy, enabling clarity in structure, ease of maintenance, and future extensibility of the system.

The core computational logic is implemented in Python, where the functional behavior of individual logic gates is modeled using deterministic Boolean expressions. A dedicated fault evaluation component computes the expected output corresponding to the selected gate and input values and compares it with the output observed by the user. This comparison forms the basis for identifying logic-level faults.

The Flask framework acts as a bridge between the computational backend and the web-based user interface. The graphical interface is implemented using standard HTML and CSS, allowing users to interactively select logic gates, apply input combinations, and submit observed outputs for analysis. The overall design prioritizes simplicity and clarity, ensuring ease of use for both academic and experimental applications. This software-only and modular implementation approach minimizes cost, simplifies deployment, and eliminates the need for specialized hardware, making the system suitable for educational environments and early-stage research experimentation.

A. Software Environment and Tools

To ensure openness and reproducibility, the system is developed entirely using open-source technologies. Python is selected as the primary programming language due to its clean syntax, ease of understanding, and extensive support for logical computation and web-based application development. The Flask framework is employed to construct the web application because of its lightweight nature and flexibility in integrating backend logic with frontend interfaces.

The development setup relies solely on standard Python libraries, and no proprietary software packages or dedicated hardware resources are required. As a result, the system remains platform-independent and can be executed on any machine equipped with a Python interpreter and a modern web browser.

B. Backend Module Design

The backend architecture is organized into distinct functional modules, each handling a specific aspect of the system operation. Logic gate simulation is carried out using deterministic Boolean functions to accurately represent ideal gate behavior. The fault detection module is responsible for evaluating discrepancies between calculated outputs and user provided outputs to determine the fault status.

In addition, a separate correction analysis module generates possible corrective input suggestions when a fault is detected. This modular backend structure facilitates straightforward modification and extension of the system, enabling future incorporation of additional logic gates or more complex circuit configurations.

C. Frontend Interface Implementation

The frontend interface is designed with a focus on usability and minimal complexity. HTML is used to define the structure of user input forms, while CSS enhances readability and visual organization. Dynamic handling of input fields ensures that the interface adapts according to the selected logic gate, thereby preventing invalid or inconsistent input scenarios.

The interface presents system outputs such as fault status, expected output values, and corrective recommendations in a structured and intuitive format. This clear presentation aids users in understanding system behavior and interpreting fault analysis results effectively.

D. Data Flow and Execution Process

System execution follows a structured request–response workflow. Input data submitted through the web interface is transmitted to the Flask backend, where logic evaluation, fault detection, and correction analysis are performed. The processed results are then sent back to the frontend for real-time visualization.

This organized data flow ensures efficient processing, reduces computational overhead, and provides immediate feedback to users. The clear separation between frontend and backend responsibilities enhances system reliability and simplifies debugging and maintenance tasks.

E. Implementation Benefits

The adopted implementation strategy offers several notable advantages:

- The software-only architecture removes dependence on external hardware resources.
- Modular design improves scalability and long-term maintainability.
- Web-based accessibility increases portability and user convenience.

Deterministic logic evaluation guarantees transparent and accurate fault analysis.

Overall, the implementation effectively supports the proposed fault detection methodology by delivering a reliable, interpretable, and user-friendly platform for analyzing logic level faults in VLSI circuits.

X. RESULTS AND DISCUSSION

This section discusses the outcomes obtained from experimental evaluation of the proposed AI-inspired fault detection framework. The system performance is analyzed using software-based simulations of fundamental digital logic gates under both fault-free and faulty scenarios. The results demonstrate the effectiveness, accuracy, and interpretability of the proposed approach, along with its practical applicability through an interactive web-based interface.

A. Experimental Environment and Test Setup

The experimental validation was performed in a controlled software simulation environment implemented using Python and the Flask web framework. The proposed system was evaluated on commonly used digital logic gates such as AND, OR, NOT, NAND, NOR, XOR, and XNOR. For each gate, all valid combinations of binary inputs were applied to ensure comprehensive functional verification.

To emulate both correct and erroneous circuit behavior, the system was tested using correct outputs as well as deliberately altered observed outputs. This approach enabled the simulation of normal operation and fault conditions, respectively. Since the system operates exclusively at the logic level, it does not depend on physical hardware or pre-trained datasets, thereby ensuring repeatability and consistency in experimentation. The experimental setup provides a reliable basis for validating the effectiveness of the proposed fault detection methodology.

B. User Interface and Homepage Analysis

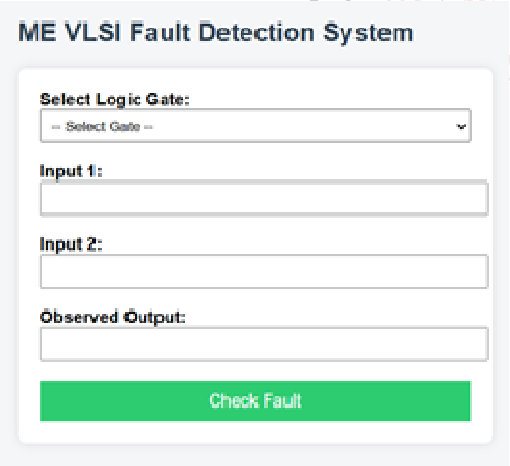

Fig. 2. Homepage interface of the proposed system

The homepage of the proposed application features a clean, intuitive, and well-organized interface that facilitates easy interaction with the fault detection system. Users can select logic gates, apply input values, and submit observed outputs without requiring specialized technical knowledge.

The interface design emphasizes simplicity and clarity, making the platform accessible to students, educators, and beginners in digital electronics. The step-by-step interaction flow helps users understand the fault detection process intuitively, thereby enhancing learning and usability.

C. Logic Gate Selection and Input Handling

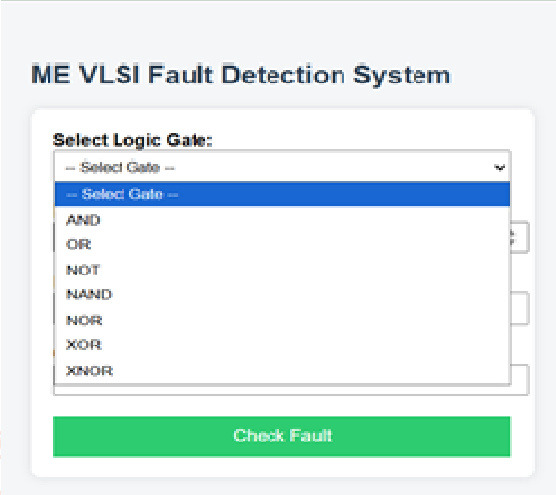

Fig. 3. Logic gate selection and dynamic input handling

The system supports multiple basic logic gates including AND, OR, NOT, NAND, NOR, XOR, and XNOR. Based on the selected gate, the user interface dynamically adjusts the input fields. For instance, when the NOT gate is selected, the system automatically disables the second input field to prevent invalid input configurations.

This adaptive input handling mechanism improves correctness, reduces user input errors, and enhances overall user experience. Additionally, it visually reinforces the logical characteristics of different gates, which is particularly beneficial for educational applications.

D. Normal Operation Results

During fault-free operation, when the user-provided observed output matches the expected output computed through Boolean logic evaluation, the system classifies the circuit as operating normally. A confirmation message indicating correct behavior is displayed on the interface.

The experimental results verify that the proposed system consistently identifies correct logic behavior across all supported gates and valid input combinations. This confirms the accuracy of the Boolean gate modeling and the reliability of the deterministic evaluation mechanism employed in the system.

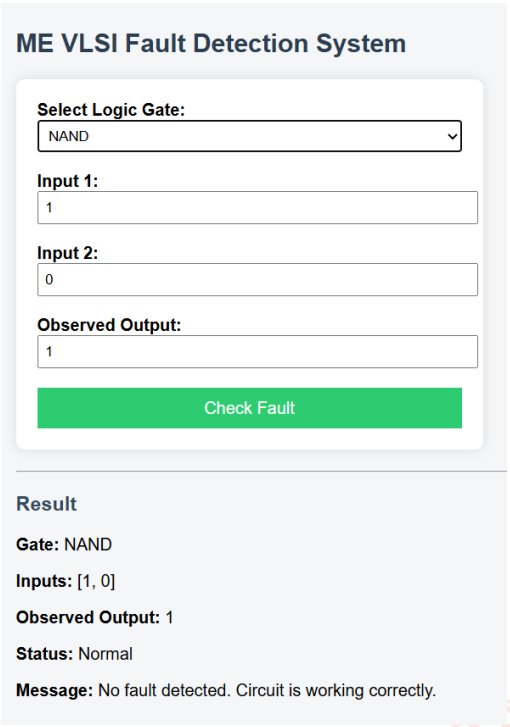

Fig. 4. Normal operation result of a logic gate

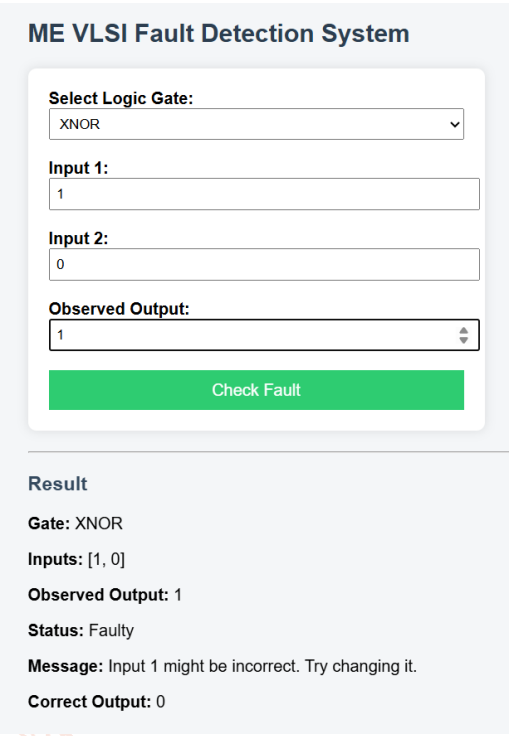

Fig. 5. Fault detection result of a logic gate

E. Fault Detection Results

Fault conditions were simulated by supplying incorrect observed outputs for selected input combinations. In such cases, the system successfully detected mismatches between expected and observed outputs and correctly identified the circuit as faulty.

Unlike traditional approaches that rely on predefined fault models such as stuck-at or bridging faults, the proposed system focuses on functional correctness at the logic level. This enables detection of a broad range of logical inconsistencies without assumptions about specific fault types.

F. Corrective Suggestion Analysis

In addition to fault identification, the proposed framework provides intelligent corrective suggestions. Once a fault is detected, the system evaluates alternative input combinations to determine possible inputs that could yield the observed output.

This corrective feedback mechanism significantly enhances the debugging process by offering practical guidance instead of merely indicating the presence of a fault. Such functionality is especially valuable in academic settings, where understanding the cause and correction of faults is essential for effective learning. Experimental observations confirm that the corrective suggestions generated by the system are consistent, meaningful, and easy to interpret.

G. Performance and Accuracy Evaluation

Under the simulated test conditions, the proposed system achieved 100

Furthermore, the software-only implementation ensures minimal computational overhead and fast response times. This makes the system suitable for real-time interaction, classroom demonstrations, and hands-on experimentation.

H. Overall Discussion

The experimental results clearly demonstrate that the proposed AI-inspired fault detection framework effectively integrates deterministic logic evaluation with intelligent decision making. Although the system does not employ conventional machine learning models, its structured analysis and corrective suggestion capability justify its classification as an AI-inspired approach.

Overall, the proposed system proves to be accurate, transparent, cost-effective, and highly suitable for academic learning, preliminary VLSI fault analysis, and digital logic education.

XI. COMPARISON WITH EXISTING METHODS

This section presents a detailed comparison between conventional fault detection techniques used in digital and VLSI circuits and the proposed AI-inspired fault detection system. The objective of this comparison is to highlight the improvements offered by the proposed approach in terms of automation, usability, cost, and interpretability.

TABLE I Comparison with Existing Fault Detection Methods

Parameter	Existing Methods	Proposed System
Automation Level	Partial	Full
Fault Detection Technique	Hardware-based	Logic-Software based
Fault Correction Support	No	Yes
Implementation Cost	High	Low
System Complexity	High	Low
User Friendliness	Low	High
Educational Suitability	Low	High
Interpretability	Low	High

Traditional fault detection techniques primarily include manual testing, rule-based verification, and hardware dependent automated test equipment (ATE). While these methods are effective for large-scale industrial testing, they often require significant human effort, specialized hardware, and high operational cost. Moreover, most conventional approaches focus only on fault identification and do not provide any form of corrective guidance to the user.

In contrast, the proposed system adopts a software-only, logic-level fault detection methodology that eliminates hardware dependency. The system automatically evaluates the expected logic behavior and compares it with the observed output, enabling fully automated fault detection. Additionally, the proposed system extends beyond fault identification by offering corrective suggestions, which is not commonly supported in existing methods.

Table I summarizes the key differences between existing fault detection techniques and the proposed system.

From the comparison, it is evident that the proposed system offers significant advantages over traditional fault detection approaches, particularly in educational and preliminary testing scenarios. The absence of hardware dependency reduces complexity and cost, making the system accessible to a wider user base.

Furthermore, the corrective suggestion feature distinguishes the proposed approach from existing methods by providing actionable insights rather than merely reporting faults. This feature enhances learning outcomes and supports efficient debugging of logic circuits.

Overall, the proposed system strikes a balance between simplicity and effectiveness, making it a practical alternative to conventional fault detection techniques for academic, training, and early-stage VLSI design applications.

XII. APPLICATIONS

The proposed fault detection system can be applied in various domains. It is particularly useful in VLSI education for demonstrating logic gate behavior and fault diagnosis concepts. The system can also be employed in academic laboratories as a learning and experimentation tool.

Additionally, it can assist students and engineers in understanding digital logic verification and debugging at the gate level. Due to its software-only nature, the system is cost-effective and easy to deploy across different platforms.

XIII. FUTURE SCOPE

Although the proposed system efficiently detects logic-level faults, several enhancements can be explored in the future. Machine learning models can be integrated to classify complex fault patterns automatically. Support for combinational and sequential circuits can be added to increase system applicability. Further improvements may include detection of stuck-at faults, delay faults, and integration with hardware description languages such as Verilog. Cloud-based deployment can also be explored to enable remote access and large-scale testing.

XIV. CONCLUSION

This work introduced an AI-inspired fault detection framework for VLSI logic circuits that operates entirely through software-based simulation and deterministic logic analysis. The primary aim of the proposed system was to enable accurate identification of logic-level faults in digital circuits while also providing useful corrective feedback, without depending on hardware testing setups or complex machine learning techniques. The results confirm that this objective has been successfully achieved through a structured and reliable design approach.

The developed system functions at the logic gate level, where expected circuit outputs are calculated using Boolean logic formulations and systematically compared with user provided observed outputs. This output comparison strategy allows effective identification of functional discrepancies without relying on predefined fault assumptions. Consequently, the system remains transparent, easy to interpret, and robust across various logic gate configurations.

A significant contribution of this work lies in the incorporation of an intelligent correction support feature. Rather than only signaling the existence of a fault, the system evaluates possible alternative input scenarios and identifies combinations that may yield correct behavior. This capability improves the overall

usefulness of the system by offering practical guidance, making it particularly valuable for learning, analysis, and debugging applications.

Experimental validation confirms that the proposed framework accurately distinguishes between correct and faulty behavior across multiple basic logic gates such as AND, OR, NOT, NAND, NOR, XOR, and XNOR. The deterministic nature of Boolean evaluation ensures consistent and precise fault detection results for all tested input conditions, eliminating issues such as uncertainty, bias, or dependency on training data that are often associated with data-driven methods.

An additional strength of the proposed system is its purely software-based implementation. By removing the requirement for hardware resources or automated test equipment, the overall system complexity and cost are significantly reduced. The use of Python along with the Flask framework enables a flexible, modular, and extensible architecture. Furthermore, the web-based interface enhances accessibility, allowing users to interact with the system through standard browsers without additional installations.

From an educational and academic standpoint, the proposed framework serves as an effective tool for demonstrating digital logic behavior and fault analysis principles. The clear logic evaluation process and understandable fault diagnosis make it suitable for students, researchers, and beginners in VLSI design. It also provides a useful preliminary validation platform before progressing to detailed hardware-level verification.

Although the current implementation primarily targets logic level functional faults, it establishes a solid foundation for future extensions. The modular structure of the system allows integration of more advanced fault models and intelligent analysis techniques. The results demonstrate that intelligent fault diagnosis can be effectively achieved through deterministic logic reasoning without necessarily relying on complex machine learning models.

In summary, the proposed AI-inspired fault detection system offers an accurate, cost-efficient, and interpretable solution for logic-level fault analysis in VLSI circuits. By combining deterministic logic evaluation with intelligent corrective feedback, the system bridges the gap between traditional rule-based testing methods and advanced AI-driven approaches. Its strong performance, simplicity, and educational relevance make it a meaningful contribution to the domain of digital circuit testing and VLSI fault detection.

REFERENCES

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing: For Digital, Memory and Mixed-Signal Circuits*, Springer, 2000.

[2] S. Wang and S. K. Gupta, "ATPG for delay faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 8, pp. 1538–1549, 2006.

[3] J. P. Hayes, "Introduction to Digital Logic Testing," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1098–1113, 1980.

[4] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, 1981.

[5] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[6] Y. Wang, X. Chen, and Z. Li, "Machine learning-based fault diagnosis for VLSI circuits," *IEEE Access*, vol. 8, pp. 123456–123465, 2020.

[7] S. Ghosh, A. Dutta, and R. Kar, "Neural network-based fault diagnosis in digital circuits," *IEEE Transactions on VLSI Systems*, vol. 25, no. 6, pp. 1873–1886, 2017.

[8] R. Gupta and S. K. Bose, "Support vector machine for VLSI fault detection," *IEEE Access*, vol. 6, pp. 40234–40245, 2018.

[9] Y. Liu, H. Zhang, and Q. Wang, "Deep learning approaches for VLSI fault diagnosis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 3852–3864, 2020.

[10] E. J. McCluskey, *Logic Design Principles with Emphasis on Testable Semiconductors*, Prentice Hall, 1986.

[11] R. Gupta and S. Bose, "Web-based digital logic simulation for engineering education," *International Journal of Engineering Education*, vol. 34, no. 2, pp. 620–630, 2018.

[12] R. Ubar, J. Raik, and H.-T. Vierhaus, *Design and Test Technology for Dependable Systems-on-Chip*, IGI Global, 2011.