Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

AI-Based Built-In Self-Test for CMOS Fault Detection Using Machine Learning: A Two-Input NAND Gate Case Study

Authors name: [1]Dr. Nuthan A C ,[2] Keerthana M U , [3]Kusuma C ,[4]Nayana A R ,[5] Nandan M G
[1]Professor, [2]Student , [3]Student , [4]Student, [5]Student
Department of Electronics and Communication Engineering
G .Madegowda Institue of Technology ,Bharathi nagara , Mandya, Karnataka, India

Abstract - Built-in self-test (BIST) integrates test generation and response analysis with a circuit under test. This paper proposes a machine-learning-assisted BIST framework for a complementary metal–oxide–semiconductor (CMOS) two-input NAND gate. The framework applies all four input combinations and observes the gate response; where transistor-level faults are studied, ordered input transitions and analog or timing features are added. A supervised classifier is proposed to distinguish fault-free behavior from modeled fault classes, while a conventional truth-table comparator serves as the essential baseline. A worked analytical example shows that all six single stuck-at faults at the gate inputs and output are detectable with exhaustive input patterns, but several faults have identical output responses and cannot be individually diagnosed from those responses alone. No simulation or experimental performance is claimed. The paper specifies a reproducible fault-injection and evaluation procedure and identifies the measurements required before publication as an experimental study.

Index Terms - BIST, CMOS, NAND gate, machine learning, stuck-at fault, fault diagnosis.

I. INTRODUCTION

Manufacturing defects, aging and process variation can change the behavior of CMOS circuits. Testing is therefore needed to determine whether a device conforms to its intended logic and timing. Built-in self-test incorporates test generation and response evaluation into the system, reducing reliance on external test equipment during operation or maintenance [1]. However, compact signatures can alias, and the same observed response may correspond to more than one fault [2].

This study uses one two-input NAND gate as the circuit under test (CUT), not as a claim that all BIST components are implemented with NAND gates. Its small input space enables an exact baseline. Machine learning (ML) is proposed as an optional diagnostic layer for richer measurements, rather than as a replacement for the ideal comparator. Published studies demonstrate ML-based diagnosis in larger circuits and cell-aware settings, but their performance cannot be assumed for this gate [5], [6].

The contributions are a transparent gate-level fault example, a proposed architecture spanning static and transistor-level tests, and an evaluation plan that avoids unverified timing accuracy or fault-coverage claims. The experimental design is prospective; novelty relative to prior art must be assessed before submission.

II. CMOS NAND GATE AND FAULT MODEL

A conventional static CMOS NAND gate contains two parallel pMOS transistors in the pull-up network and two series nMOS transistors in the pull-down network. With inputs A and B, the ideal output is Y = NOT(A AND B). Table I gives the expected response. For this single gate, all input combinations can be applied directly; an LFSR is optional and is not needed to obtain full input-space coverage.

TABLE I. IDEAL TWO-INPUT NAND TRUTH TABLE

A	B	Expected Y
0	0	1
0	1	1
1	0	1
1	1	0

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

A stuck-at fault forces a modeled signal to logic 0 or 1. Here the modeled fault sites are A, B and Y, with one fault at a time. These are logic-level abstractions, not a complete list of physical transistor defects. A stuck-open transistor may create state-dependent behavior requiring an initialization vector followed by a detection vector [3]. Resistive shorts or delay defects can also require timing-sensitive tests [4]. Their detection must be verified at transistor level under specified electrical conditions.

III. PROPOSED BIST AND ML ARCHITECTURE

Test controller → pattern generator → NAND CUT → response capture → comparator / ML classifier

The test controller selects the test mode and records the applied input. For the static experiment, a two-bit counter can generate 00, 01, 10 and 11 in that order. A response-capture register stores each settled digital output; the comparator checks it against the expected value. An optional signature register may compress responses, but for a single gate direct capture retains more diagnostic information and makes any compression aliasing avoidable.

For physical-fault studies, the controller should apply a documented set of ordered vector pairs, including reset or initialization sequences where needed. The capture circuit may also record propagation delay, supply current, or output voltage at selected sample times, if these measurements are available. The source of each feature—simulation, on-chip monitor or external measurement—must be reported. The ML classifier is assumed off-chip unless an on-chip implementation and its hardware cost are demonstrated.

The classifier receives a feature vector formed from input identity, measured outputs and, when available, analog or timing values. Its first task is binary detection: fault-free versus faulty. Fault-class prediction is a separate task and must allow an ambiguous or unresolved outcome. Models such as a decision tree or support vector machine can be evaluated against the comparator and a fault dictionary [7].

IV. ANALYTICAL STUCK-AT EXAMPLE

For input order 00, 01, 10, 11, the ideal response vector is 1110. Table II gives the response expected when each single line fault is modeled at the CUT boundary. A fault is detected if at least one captured bit differs from 1110. Exhaustive testing detects all six modeled faults in this ideal example: 6/6 = 100% analytical model coverage. This is not measured silicon coverage and does not include transistor faults.

TABLE II. SINGLE STUCK-AT RESPONSE VECTORS

Condition	Response (00, 01, 10, 11)	Detected?
Fault-free	1110	—
A stuck-at-0	1111	Yes
A stuck-at-1	1010	Yes
B stuck-at-0	1111	Yes
B stuck-at-1	1100	Yes
Y stuck-at-0	0000	Yes
Y stuck-at-1	1111	Yes

A stuck-at-0, B stuck-at-0 and Y stuck-at-1 all produce 1111. They are detectable but cannot be uniquely distinguished using only these four output bits. A classifier receiving the same information cannot recover a distinction that the observations do not contain. Further tests or independently informative features may improve diagnosis, but such improvement must be demonstrated. The example also shows that a truth-table comparator detects these faults without ML.

V. EXPERIMENTAL METHODOLOGY

A. Circuit and data generation: Implement the CMOS NAND in a transistor-level simulator using a stated process design kit, transistor dimensions, supply voltage, load, temperature, input slew and timing conditions. Record the exact tool and version. Simulate fault-free operation and individual, explicitly specified injected defects. Start with the six logic-level faults above; add stuck-open, short and resistive defects only with physically meaningful models and injection locations. Do not represent a transistor defect as an ideal line fault without justification.

B. Pattern application and capture: Apply the complete static pattern set and, for dynamic faults, define and apply ordered transition pairs. Capture the output after a stated settling interval. For timing or current features, specify stimulus edge, sample window, threshold, units and instrument or simulator. Include process, voltage, temperature and noise variation where relevant. Label each observation from the known injected condition, not from the classifier prediction.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

C. Model development: Separate training, validation and final test sets by device instance or simulation configuration before preprocessing. Keep correlated waveforms and repeated measurements from the same instance in one split. Fit scaling or feature selection on the training set only. Specify model parameters, class balance, random seed and threshold; select hyperparameters on validation data, then evaluate the held-out set once. A split by defect instance or condition assesses generalization; list any fault classes missing from training.

VI. EVALUATION AND DISCUSSION

Compare (1) an exhaustive truth-table comparator, (2) an uncompressed response fault dictionary, and (3) the ML model using the same test cases and observables. Compare signature compression separately if used. For detection, report the confusion matrix, sensitivity = TP/(TP + FN), specificity = TN/(TN + FP), and false-positive rate = FP/(FP + TN). Define TP, TN, FP and FN with "faulty" as the positive class. Report diagnostic accuracy or macro-F1 separately, alongside the number of unresolved or indistinguishable cases. For class imbalance, include per-class counts and results.

Fault coverage = detected modeled faults / total modeled faults × 100%, with the fault list and detection criterion disclosed. Report the number of independent test instances and confidence intervals where feasible. Give simulation cost, test time and any measured area or power overhead. To justify ML, demonstrate a benefit over the baseline on held-out, physically realistic data; the small truth-table example alone does not establish one.

TABLE III. RESULTS TO BE FILLED AFTER EXPERIMENTS

Metric	Comparator	ML model
Detected / modeled faults	[measure]	[measure]
Sensitivity / specificity	[measure]	[measure]
Diagnostic macro-F1	[measure]	[measure]
Test time / hardware cost	[measure]	[measure]

Limitations include limited generality of a single gate, the sensitivity of analog features to process and environmental variation, and ambiguity between faults with identical observable behavior. A larger circuit, physical measurements, and an explicit comparison of instrumentation cost would be needed to establish broader practical value.

VII. CONCLUSION

This paper proposes a BIST and ML framework for a CMOS NAND gate and presents an exact logic-level reference example. Exhaustive input testing detects all six stated single stuck-at faults, while identical responses limit individual diagnosis. Experimental verification is still required; neither ML superiority nor physical-defect coverage is claimed.

REFERENCES

[1] E. J. McCluskey, "Built-In Self-Test Techniques," IEEE Design & Test of Computers, 1985. doi:10.1109/MDT.1985.294856.
[2] D. K. Pradhan and S. K. S. Gupta, "A New Framework for Designing and Analyzing BIST Techniques and Zero Aliasing Compression," IEEE Transactions on Computers, 1991. doi:10.1109/12.90252.
[3] H. K. Lee and D. S. Ha, "An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits," VLSI Design, 1994. doi:10.1155/1994/71941.
[4] S. Pandey et al., "Two Pattern Timing Tests Capturing Defect-Induced Multi-Gate Delay Impact of Shorts," IEEE VLSI Test Symposium, 2021. doi:10.1109/VTS50974.2021.9441005.
[5] L. Rodríguez Gómez and H.-J. Wunderlich, "A Neural-Network-Based Fault Classifier," Asian Test Symposium, 2016. doi:10.1109/ATS.2016.46.
[6] S. Mhamdi et al., "Cell-Aware Defect Diagnosis of Customer Returns Based on Supervised Learning," IEEE Transactions on Device and Materials Reliability, 2020. doi:10.1109/TDMR.2020.2992482.
[7] L. Zhao and V. D. Agrawal, "Net Diagnosis Using Stuck-at and Transition Fault Models," IEEE VLSI Test Symposium, 2012. doi:10.1109/VTS.2012.6231106.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)