A Novel Meta-predictor based Algorithm for Testing VLSI Circuits

Shruti Pandey
Electrical Engineering, IIT Delhi
New Delhi, India
shruti.pandey@ee.iitd.ac.in

Jayadeva
Electrical Engineering, IIT Delhi
New Delhi, India
jayadeva@ee.iitd.ac.in

Smruti R. Sarangi
Electrical Engineering, IIT Delhi
New Delhi, India
srsarangi@cse.iitd.ac.in

Abstract—Testing of integrated circuits (IC) is a highly expensive process but also the most important one in determining the defect level of an IC. Manufacturing defects in the IC are modeled using stuck-at-fault models. Stuck-at-fault models cover most of the physical faults that occur during the manufacturing process. With decreasing feature sizes due to the advancement of semiconductor technology, the defects are also getting smaller in size. Tests for these hard-to-detect defects are generated using deterministic test generation (DTG) algorithms. Our work aims at reducing the cost of Path Oriented Decision Making: PODEM (a DTG algorithm) without compromising the test quality. We trained a meta predictor to choose the best model given the circuit and the target net. This ensemble chooses the best probability prediction model with a 95% accuracy. This leads to a reduced number of backtracking decisions and much better performance of PODEM in terms of its CPU time. We show that our ML-guided PODEM algorithm with a meta predictor outperforms the baseline PODEM by 34% and other state-of-the-art ML-guided algorithms by at least 15% for ISCAS85 benchmark circuits.

Index Terms—PODEM, stuck-at-fault, ATPG, ANN

I. INTRODUCTION

VLSI chips are increasingly becoming very pervasive in today’s era of IoT and 5G. ICs (integrated chips) are being employed in novel embedded, IoT, and novel 5G-enabled settings, and are often used for real-life applications that demand a very high degree of reliability and correctness. Hence, ensuring the correct functioning of electronic circuits is crucial (regardless of the ambient conditions). The advancement in semiconductor technology is hugely driven by emerging technologies like metaverse, industry 4.0, AI/ML, quantum computing, AR/VR, 5G, and cloud computing. These technologies are used in applications like autonomous driving, smart medical devices, and industrial IoT. Hence, device failure is not an option in such cases. Thus, digitalization of the IC testing process and the development of novel testing techniques tailored for these technologies is a potential area for research. Classically, testing is done to increase our confidence in the correct functioning of an IC. It is a highly complex and expensive process because of the high coverage requirements. The quality of the test directly affects the quality of the electronic product being shipped. Testing has become more challenging with the exponential growth in the scale of ICs along with aggressive angstrom-level technologies with novel failure mechanisms. The test engineer’s intuition and experience are used as heuristics to make the process manageable in the case of very large circuits [1]. Human experience-based or institutional knowledge-based problem solving is both error-prone and time-consuming. ML algorithms, on the other hand, have proven their efficacy and have been shown to be faster and more accurate in areas such as computer vision, natural language processing, and electronic design automation. Moreover, ML has proven to be effective in providing good approximate solutions to NP-complete problems very quickly [2].

In this work, we designed an ML-guided PODEM with the meta predictor, which shows a two-fold speedup in the CPU time over that of the baseline PODEM. Our contributions are as follows:

- Implementation of PODEM in C++ using the standard template library (STL).
- Development of methods for the computation of design for testability measures.
- Introduction of correct methodology for the computation of no-backtrack probability.
- Integration of ML-model based heuristics with the PODEM algorithm. We have designed PODEM with four modes representing different heuristics: viz. random or no heuristic, distance of the gate from the PI, controllability, and ML guided no-backtrack probability.
- We improved the performance of our ML-guided PODEM further by incorporating a novel meta predictor into the design.

Our work is a significant contribution in the direction of reducing the complexity and cost of the VLSI testing process using ML algorithms. We have leveraged the proven efficacy of ML algorithms in decision-making and design space exploration to accelerate the testing process.

Our paper is organised as follows: Section II presents the background of the test generation algorithm Path Oriented Decision Making (PODEM) and other key concepts involved in our work. We present the related work in section III. Section IV presents the methodology used in the work. We present the experiments and their results in Section V. Finally, we conclude the paper in section VI.
II. BACKGROUND

A. Testing

Testing of a system is an experiment in which the system is exercised and its resulting response is analyzed to ascertain whether it behaved correctly. Testing of an integrated circuit (IC) involves two steps: test generation and test application. The testing procedure is described in Fig. 1.

![Testing of an IC](image)

Fig. 1. Testing of an IC: test generation and test application

B. Fault Modeling

Fault modeling is defined as the abstraction of physical defects to quantify the faults and make their analysis easier. It is required as there can be infinite physical defects in a system. Physical faults are modeled with the help of logical fault models for the reasons described next. As a single logical fault can model different physical faults, fault analysis becomes systematic, and the complexity of physical faults is reduced significantly. Some of the logical fault models are technology independent, and the tests derived for logical faults can also be used for physical faults.

There are some assumptions made while dealing with the fault models. These assumptions are described below.

1) We always assume that we have at most one logical fault in the system. We can assure that the single fault assumption holds true by doing the testing experiments frequently. The frequency of testing experiment is kept such that the probability of developing a new fault between two consecutive experiments is very small. Multiple faults can be detected with the test derived for single faults.

2) The components are fault free and only their interconnections are affected.

3) We assume that the entire signal line is stuck in edge pin testing, hence we do not need to probe the end points of the line for finding whether the fault is a result of any internal fault in the component or it is due to an open or short circuit in the line.

C. Test Generation

It is defined as the process of determining the stimuli necessary to test a digital system. Now, we present a taxonomy for the types of test generation.

Random Test Generation (RTG): Test vectors are generated using a pseudo-random sequence generator. The advantage of RTG is that the cost of test generation is very low. But the length of the generated vectors is usually much higher than that of the vectors generated using deterministic test generation.

Deterministic Test Generation (DTG): Tests are generated based on a model of the circuit and a given fault model. The generated tests include both the input vectors to be applied and the expected response of the fault-free circuit. DTG can be manual or automatic as well as fault-oriented or fault-independent.

Manual TG: It involves the generation of tests by humans using the information present in the functionality or structure of the circuit. Manual TG can be efficient for some circuits like ripple-carry adders that can be tested with 8 test vectors only, irrespective of their width.

Automatic TG (ATG): Test generation algorithms that generate tests, given a model of a system are termed as automatic test generation (ATG) algorithms.

Fault oriented TG: Tests are generated to detect specific faults in the system.

Fault independent TG: It does not target specific faults. We do not need to maintain a set of undetected and target faults. Here, we aim at detecting a large set of single stuck-at-faults (SSFs) without targeting individual faults.

The fundamental steps involved in the generation of tests are:

- Fault Activation: Activating a s-a-v fault means setting primary input (PI) values such that line ‘l’ gets a value $v$. We excite or activate the fault by setting a value opposite to the fault value.

- Fault Propagation: The fault at line ’l’ needs to be propagated to the PO to be observed. We propagate the fault at line l by setting all other inputs of the faulty gate to the non-controlling value (NCV).

We have a five-valued logic system to describe a faulty circuit. The five logic values are 0, 1, X, D and D. D designates a logic ‘1’ for a net in the good circuit and a logic ‘0’ for the same net in the faulty circuit. An input value is said to be controlling if it determines the output value regardless of the other input values. The controlling value for AND/NAND gates is ‘1’ whereas for OR/NOR gates is ‘0’. The gates whose output value is unknown but have one or more error signals at the input are said to form a D-frontier. A signal line may get stuck or take a permanent value of 1 or 0 if it is open or short to power or ground as shown in Figure 2. The faults due to open or short circuits are termed as stuck-at-faults. We have considered a single stuck-at-fault model (SSF) for performing fault-oriented automatic test generation (ATG).

D. Path Oriented Decision Making (PODEM)

PODEM is a fault-oriented ATPG algorithm characterised by a direct search process. It consists of decisions only at the
Pls. PODEM treats a value \( v \) to be justified for line \( l \) as an objective \((l, v)\) that can be achieved via PI assignments.

- **Backtracing:** Backtracing is the procedure that maps a desired objective to a PI assignment. We traverse backwards through the inputs of the faulty gate till we reach a PI. The value assigned at the PI is the complement of the fault value if there are an odd number of inverting gates in the backtrace path, else it is equal to the fault value.

- **Implication:** Implication is the process of computing the values due to an assignment and checking the consistency of these values with the previously determined ones. After making a PI assignment in the process of backtracing, we perform implication due to that assignment through logic simulation. Next, we check whether the desired objective is met. If the objective is not met, we keep backtracing to make different PI assignments. When the objective is achieved, we look for a new objective by selecting one of the gates in the D-frontier. Then, in the D-frontier, we want to set all the other inputs of the gates to their non-controlling value (NCV).

- **Backtracking:** Backtracking is the process of recovering from incorrect decisions by a systematic exploration of the complete space of possible solutions (for example, reversing the decision). When the implications due to a PI assignment either prohibit the fault from being activated or propagated, we change our decision and reverse the value of that PI assignment. This is like backtracking a decision tree.

### E. Problems in Test Generation

1) **The cost of the TG:** The cost of the TG depends upon the complexity of the TG algorithm and is measured in terms of the CPU time of the algorithm. The cost is very low for RTG algorithms, whereas it is high for DTG algorithms.

2) **The quality of the test generated:** The quality of the generated test vector set \( T \) is measured by its fault coverage. Fault coverage is defined as the ratio between the number of faults detected using the test vector set \( T \) for a given fault model \( F \) and the total number of faults present in the circuit. Test vectors are generated with the objective of achieving high fault coverage.

3) **The cost of applying the test:** The tests are applied to the circuit under test, and the cost of application depends on the length of the test vectors. The test vectors should have a shorter length to reduce the testing time and memory requirements of the tester.

### F. Design for Testability

- **Controllability/Observability Program (COP):** Here, controllability is defined as the probability of setting a line \( y \) to 0 or 1 using the PI assignments. And observability is defined as the probability of propagating a logical value/error from a line to the primary output (PO). The netlist should be in the levelized order to perform correct computations of CC and CO values.

| Gate | \( \text{CC}(z) \) | \( \text{CO}(x) \) |
|------|------------------|-------------------|
| AND  | \( \text{CC}(x) \times \text{CC}(y) \) | \( \text{CO}(z) \times \text{CC}(y) \) |
| OR   | \( 1 - (\text{CC}(x) \times \text{CC}(y)) \) | \( \text{CO}(z) \times \text{CO}(y) \) |
| NAND | \( 1 - (\text{CC}(x) \times \text{CC}(y)) \) | \( \text{CO}(z) \times \text{CC}(y) \) |
| NOR  | \( \text{CC}(x) \times \text{CO}(y) \) | \( \text{CO}(z) \times \text{CO}(y) \) |

- **Sandia Controllability/Observability Analysis Program (SCOAP):** SCOAP combinational controllability is defined as the difficulty of setting a line’s logic level to 1/0. It is represented as CC0/CC1. It gives an idea about the minimum number of PIs required to justify a value \( v \) at line \( l \). SCOAP combinational observability is defined as the difficulty of observing an error/fault at an internal line at the PO. It is denoted as CO. It gives an idea about the minimum number of PI assignments to be made to propagate the fault to a PO. SCOAP measures do not take into reconvergent fanouts into account.

### III. RELATED WORK

Machine Learning (ML) techniques are being used to accelerate the ATPG process. The ML models are great at performing feature extraction and can therefore learn from several heuristics and combine them to give the best results. We will discuss works that use ML techniques to reduce the number of backtracks in PODEM and improve test point quality in this section.

#### A. Reduction in the number of backtracks in PODEM using ML

Fault simulation allows us to check the quality of test vectors. Test vectors with high fault coverage are desirable. With increasing complexity of designs, the number of hard to detect faults also increases, thus making the TG (test generation) process costlier. In such cases, the use of ML/DL based heuristics proves to be useful in reducing the TG time. Roy et al. [4] proposed the introduction of artificial neural network’s (ANN’s) predictions as a backtracing heuristic in the PODEM algorithm. This work reported a reduction in the number of backtracks compared to conventional heuristics. Each circuit net was input to the ANN having input features...
like distance from the PI (primary input), testability measures and type of the gate driven by the line. The experiments were performed on 100 hard-to-detect faults of the ISCAS’85 \[5\] and ITC’99 \[6\] benchmark circuits.

Roy et al. \[7\] proposed a new training strategy to improve the backtracking performance of the ANN-guided PODEM. They reported fewer backtracks for all faults in any benchmark circuit as compared to \[4\]. The backtrack history of a fault is only included in the training data if it decreases the number of backtracks in the ANN-guided PODEM. The training data is generated by running PODEM on both easy and hard to detect faults of the circuits.

A method based on unsupervised learning is proposed in \[8\], to combine different heuristics through Principal Component (PC) analysis. The major PCs are used for guiding the backtracking in PODEM. They included testability measures from Sandia Controllability/Observability Analysis Program (SCOAP) and fanout count as well in the input features for a circuit net.

In \[9\], the authors proposed a PCA-assisted supervised learning approach that reduces the complexity of the ANN used for guiding the backtracking in PODEM.

Chowdhury et al. \[10\] proposed an alternative metric to train and evaluate ML models in the context of IC testing. They also presented an analysis of the state-of-the-art DL models’ robustness to perturbations. The work showed that ML model’s output may poorly perform on the task (regression/classification) it is trained for but still be useful for the testing task. Hence, there is a need to redesign metrics for measuring the performance of ML models in the IC test domain.

B. Test Point Insertion using ML

Design for testability techniques play a vital role in making the testing process feasible. Several methods are proposed to identify test insertion points that maximise the gain in fault coverage. In this section, we talk about recent progress in using ML approaches to improve the quality of test points.

Ma et al. proposed a high-performance Graph Convolutional Network (GCN) classifier in \[11\] to predict difficult-to-observe points in a netlist. They achieved similar fault coverage with an 11% and a 6% reduction in observation points and test pattern count respectively when compared to \[4\]. The backtrace history of a fault is only included in the training data if it decreases the number of backtracks in the ANN-guided PODEM. The backtrace is generated by running PODEM on both easy and hard to detect faults of the circuits.

A method based on unsupervised learning is proposed in \[8\], to combine different heuristics through Principal Component (PC) analysis. The major PCs are used for guiding the backtracking in PODEM. They included testability measures from Sandia Controllability/Observability Analysis Program (SCOAP) and fanout count as well in the input features for a circuit net.

In \[9\], the authors proposed a PCA-assisted supervised learning approach that reduces the complexity of the ANN used for guiding the backtracking in PODEM.

Chowdhury et al. \[10\] proposed an alternative metric to train and evaluate ML models in the context of IC testing. They also presented an analysis of the state-of-the-art DL models’ robustness to perturbations. The work showed that ML model’s output may poorly perform on the task (regression/classification) it is trained for but still be useful for the testing task. Hence, there is a need to redesign metrics for measuring the performance of ML models in the IC test domain.

IV. METHODOLOGY

A. ML-Guided PODEM

1) Input Features of the ML Model:

- **The COP Combinational Controllability**: The COP combinational controllability of a line \(l\) (represented as \(CC(l)\)) indicates the probability of setting a line \(l\) to logic ‘1’. First, we assign the CC values for all the PIs to 0.5 as the PIs take values ‘1’ and ‘0’ randomly. We use the truth tables of different gates to derive the formulas for computing the CC values for the output line of these gates. Then, we calculate the CC values for the intermediate nets and POs of the circuit by traversing the circuit graph in the forward direction.

- **The COP Combinational Observability**: The COP Combinational Observability of a line \(l\) (represented as \(CO(l)\)) indicates the probability of observing an error at line \(l\) at the PO. First, we assign the CO values for all the POs to ‘1’ as any error at the POs is always observable. The formulas for computing the CO values for the input lines of different gates, given the CO values of the output line and CC values of other input lines, are derived using the truth tables of these gates. Then, we calculate the CO values for the intermediate nets and PIs of the circuit by traversing the circuit graph in the backward direction.

2) The shortest distance of a line from the PIs: The ‘distance’ feature is normalized to bring the values in the range \([0, 1]\) using the following formula

\[
distance(l) = \frac{\text{distance}(l) - distance_{\text{min}}}{distance_{\text{max}} - distance_{\text{min}}} \quad (1)
\]
where \( \text{distance}_{\min} \) and \( \text{distance}_{\max} \) are the minimum and maximum values that the distance feature takes considering all the circuit line samples, respectively.

- **The type of the gate**: The supported gate types are PI, PO, PPI, PPO, NOT, AND, NAND, OR, NOR, XOR, XNOR, DFF, BUF and BAD. The type of the gate feature is represented using 14 bits based on one-hot encoding or can be normalized in the range \([0, 1]\).

The attributes of these input features are mentioned in TABLE II.

### TABLE II
**THE INPUT FEATURES OF THE ML MODEL AND THEIR ATTRIBUTES.**

| Input Features                              | Data Type | Size (in bytes) | Range   |
|---------------------------------------------|-----------|-----------------|---------|
| The type of the gate                        | one-hot encoded | 14 bits         | \([0, 1]\) |
| Controllability, CC(l)                     | float     | 4               | \([0, 1]\) |
| Observability, CO(l)                       | float     | 4               | \([0, 1]\) |
| Distance from the PI                        | float     | 4               | \([0, 1]\) |

2) **Procedure for finding 100 hard-to-detect faults in the circuit**: First, we calculate the COP controllability and observability values of all the lines in the circuit. Then, we calculate the detection probability of a fault using Eq. 2 and Eq. 3. The detection probability of a s-a-v fault takes into account the probability of activating the fault (given by \(\text{CC}(\pi)\)) and probability of propagating the fault to a PO (given by \(\text{CO}(l)\)).

\[
\text{CC}(l) \times \text{CO}(l) \text{ for s-a-0 fault at line } l 
\]

\[
(1 - \text{CC}(l)) \times \text{CO}(l) \text{ for s-a-0 fault at line } l \]

The fault list is sorted in the ascending order of the detection probability. There is a custom to perform random test generation first and then use deterministic test generation to generate tests for hard-to-detect faults. This ensures a balance between the cost of TG and the length of tests generated. Hence, we run PODEM on the first 100 faults from the sorted fault list that have the lowest detection probability.

3) **Output of the Model**: We have generated the ground truth label for each circuit net by running PODEM without any heuristics for 100 hard-to-detect faults of the circuit. During the backtracking procedure of PODEM, if a PI assignment does not lead to any conflict, we label the nets involved in the backtrace as ‘1’, else as ‘0’. A circuit net occurs several times while running PODEM. Its input features remain the same, but the ground truth label keeps changing. Hence, we computed the probability of no backtrack for a circuit net \(l\) as follows:

\[
\frac{\text{number of times the ground truth label for circuit line } l \text{ is } 1}{\text{the frequency with which the circuit line } l \text{ appears in the data}} = \frac{\text{number of times the ground truth label for circuit line } l \text{ is } 1}{\text{the frequency with which the circuit line } l \text{ appears in the data}} \tag{4}
\]

The attributes of the output feature are mentioned in Table III.

### TABLE III
**THE OUTPUT FEATURES OF THE ML MODEL AND THEIR ATTRIBUTES.**

| Output Features                              | Data Type | Size (in bytes) | Range   |
|---------------------------------------------|-----------|-----------------|---------|
| Probability of no backtrack                 | float     | 4               | \([0, 1]\) |

4) **Network architectures**:

- **Artificial Neural Network**: We designed a fully connected feed forward neural network having an input layer, a single hidden layer, and an output layer to perform the task of regression. Each neuron in a layer is connected to all the neurons in the next layer. This ANN model takes the features of a circuit net as input and gives the probability of no backtrack as the output. The architecture of the ANN model is shown in Fig. 4. The number of neurons in the hidden layer is a hyperparameter leading to optimal results at a value of 25. The activation functions used in the hidden and the output layers are **rectified linear unit (ReLU)** and **sigmoid** respectively. The loss function and the optimizer used for training the model are mean squared error (MSE) and Adam respectively. We have used a uniform distribution in the range \((-\sqrt{k}, \sqrt{k})\) to initialize the model weights, where \(k = \frac{1}{\text{number of input features}}\).

- **Traditional ML Models**: We trained different ML models, viz. Decision Tree Regressor (DTR), Random Forest Regressor (RFR) and Support Vector Regressor (SVR), implemented using the built-in functions of the sklearn library, on our dataset. The details of these models are tabulated in Table IV. We performed inference using these trained models on all the circuit nets of each circuit to obtain the no-backtrack probability \(p\). Then, in PODEM, we used \(p\) to guide backtracking, selecting the input net with the highest \(p\).

**B. ML-Guided PODEM with Meta Predictor**

1) **Dataset**: We observed that DTR and RFR are the best performing models for different circuits. We assigned a ground truth label of “0” for circuits where RFR performs the best and “1” for DTR. We generated a balanced dataset using c1355, c1908, c3540, and c2670 benchmark circuits. The nets of
TABLE IV  
DETAILS OF THE REGRESSOR MODELS PREDICTING THE PROBABILITY OF NO-BACKTRACK

| Model                        | #Train Samples | #Test Samples | MSE   | Hyperparameter         |
|------------------------------|----------------|---------------|-------|------------------------|
| Decision Tree Regressor (DTR)| 1782           | 763           | 0.022 | Max. depth = 13        |
| Random Forest Regressor (RFR)| 1782           | 763           | 0.017 | Max. number of estimators = 93 |
| Support Vector Regressor (SVR)| 1782           | 763           | 0.027 | kernel = RBF           |

Fig. 5. The concept of the ML-guided PODEM. \( X_{\text{net}_i} \) represent the input features of the \( i_{\text{th}} \) net.

circuits c1908 and c2670 belong to class 0, whereas those of c1355 and c3540 belong to class 1. This dataset was then split in the ratio of 80:20 to obtain train and test data respectively. The input features are standardised by removing the mean and scaling to unit variance using the inbuilt function StandardScaler from the sklearn.preprocessing library.

2) Design of the meta predictor: We trained a random forest classifier (RFC) model, implemented using an inbuilt function from the sklearn library, on the train data. This RFC model trained on our dataset for the DTR/RFR classification task is termed the “meta predictor”. We performed inference on all the circuit nets of each circuit to get the class that net belongs to. Then, the inferred values are used to get the probability of no backtrack from the model corresponding to the class of the net. This probability of no backtrack is then used to guide the backtracing in PODEM. The concept of ML-guided PODEM with meta predictor for the part of the circuit of Fig. 3 is illustrated in Fig. 5.

V. EXPERIMENTS AND RESULTS

A. ML-Guided PODEM

We implemented the baseline PODEM algorithm from scratch in C++ using the standard template library. We have used Flex and Bison for parsing the netlists in the ISCAS format. We implemented the code for the ANN in PyTorch. We conducted experiments to find the optimal learning rate of 0.01 for the Adam optimizer during the ANN training. The MSE obtained on the test data for DTR, RFR and SVR models is 2.2%, 1.7% and 2.7% respectively. After running PODEM on 100 hard-to-detect faults in each circuit, we computed the CPU time, the number of backtracks, and the number of backtraces. The graph in Fig. 6 represents the ratio of the CPU time of the baseline PODEM to that of the the ML-guided PODEM. Here, we take PODEM without any heuristics as the baseline PODEM, having the CPU time ratio equal to 1. In this figure, we can see that the random forest regressor is performing the best for the circuits c1908 and c2670, whereas the decision tree is the best performing model for all the remaining circuits. The geometric mean (GM) of the CPU time ratio of all the circuits is computed for all the models. Based on the GM values, we observed that DTR-guided PODEM shows a reduction of 17.4% in CPU time compared to the baseline PODEM. Also, the results indicate that there is no clear winner among these ML models considering reduction in CPU time as the key evaluation metric. So, we came to the conclusion that we need a meta predictor that predicts at runtime the best ML model for each circuit.

B. ML-Guided PODEM with meta predictor

We obtained a test accuracy of 94.9% with an RFC model having a maximum number of estimators equal to 100 for the DTR/RFR classification task. Figure 7 presents the ratio of the CPU time of the baseline PODEM to that of the ML-guided PODEM with the meta predictor. The results indicate that the meta predictor performs as good as the best model for each circuit or better than that. The geometric mean value indicates that the meta predictor shows an improvement of 34% over the baseline PODEM. Hence, we achieved a further improvement of 16.6% over the proposed DTR-guided PODEM, which showed an improvement of 17.4% over the baseline PODEM.

The radar plot of Fig. 8 shows the importance of each feature used to train the meta predictor (RFC) for each decision tree in the trained random forest. The greater the feature importance, the more it is used in making decisions for the data samples. We observed that SCOAP one controllability is the most important feature of all.
VI. Conclusion

Test generation is a NP-complete problem and hence requires the use of heuristics to reduce its complexity. ML-based heuristics are superior to human knowledge based heuristics in many ways. We presented the no-backtrack probability ($p$) as a heuristic for guiding the PODEM backtracing procedure. The use of $p$ as a heuristic results in a great reduction in the CPU time of PODEM compared to the use of conventional or no heuristics, as $p$ is learnt by taking into account the different conventional heuristics. We observed through extensive experiments that a robust method for data generation and selection of features is instrumental in determining the performance of the ML models and thus ML-guided PODEM. We observed that the ML model that performs the best in the probability prediction task may not be the best for reducing the CPU time of PODEM. Also, no single model performed the best for all the benchmark circuits. We used an ensemble method to select the best model for a circuit at runtime to achieve a substantial performance improvement.

We further conclude that in our work, traditional ML models have outperformed the ANN models for the ISCAS’85 benchmark circuits. For more complex circuit benchmarks, the performance analysis of traditional and ANN models is a potential future task. We look forward to extending our work to more complex combinational and sequential benchmarks.

REFERENCES

[1] S. Roy, S. K. Millican, and V. D. Agrawal, “Special session–machine learning in test: A survey of analog, digital, memory, and rf integrated circuits,” in 2021 IEEE 39th VLSI Test Symposium (VTS). IEEE, 2021, pp. 1–14.

[2] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong et al., “Machine learning for electronic design automation: A survey,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 26, no. 5, pp. 1–46, 2021.

[3] L. Goldstein, “Controllability/observability analysis of digital circuits,” IEEE Transactions on Circuits and Systems, vol. 26, no. 9, pp. 685–693, 1979.

[4] S. Roy, S. K. Millican, and V. D. Agrawal, “Machine intelligence for efficient test pattern generation,” in 2020 IEEE International Test Conference (ITC). IEEE, 2020, pp. 1–5.

[5] F. Bigle, “A neutral netlist of 10 combinatorial benchmark circuits and a target translator in fortran,” in Int. Symposium on Circuits and Systems, Special Session on ATPG and Fault Simulation, June 1985, 1985, pp. 663–698.

[6] F. Corno, M. S. Reorda, and G. Squillero, “Rt-level ic’99 benchmarks and first atpg results,” IEEE Design & Test of computers, vol. 17, no. 3, pp. 44–53, 2000.

[7] S. Roy, S. K. Millican, and V. D. Agrawal, “Training neural network for machine intelligence in automatic test pattern generator,” in 2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID). IEEE, 2021, pp. 316–321.

[8] ——, “Unsupervised learning in test generation for digital integrated circuits,” in 2021 IEEE European Test Symposium (ETS). IEEE, 2021, pp. 1–4.

[9] ——, “Principal component analysis in machine intelligence-based test generation,” in 2021 IEEE Microelectronics Design & Test Symposium (MITS). IEEE, 2021, pp. 1–6.

[10] A. B. Chowdhury, B. Tan, S. Garg, and R. Karri, “Robust deep learning for ic test problems,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 1, pp. 183–195, 2021.

[11] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, “High performance graph convolutional networks with applications in testability analysis,” in Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6.

[12] Y. Sun and S. Millican, “Test point insertion using artificial neural networks,” in 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2019, pp. 253–258.

[13] S. Millican, Y. Sun, S. Roy, and V. Agrawal, “Applying neural networks to delay fault testing: Test point insertion and random circuit training,” in 2019 IEEE 28th Asian Test Symposium (ATS). IEEE, 2019, pp. 13–135.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., “Pytorch: An imperative style, high-performance deep learning library,” Advances in neural information processing systems, vol. 32, 2019.