Unsupervised Learning Provides Intelligence for Testing Hard to Detect Faults

Soham Roy
Intel Corp., Santa Clara, CA 95054
soham.roy@intel.com

Vishwani D. Agrawal
Auburn Univ., Auburn, AL 36849
agrawvd@auburn.edu

Abstract—**Finding tests for hard-to-detect (HTD) faults in complex designs is challenging. Researchers use testability measures to guide automatic test pattern generation (ATPG) programs to improve fault detection efficiency. However, each measure favors the detection of specific faults in the same circuit. Principal component analysis (PCA), an unsupervised learning technique, has been used to combine several algorithmic testability measures. Guidance from the PCA measure was found to uniformly improve the ATPG efficiency with fewer backtracks, lower ATPG CPU time, detection of many HTD faults, fewer aborted faults, and increased fault coverage. The present work shows that these benefits continue further when we also included topological factors like fan-in and fanout cone base widths, fanout reconvergence data, and even-odd inversions on reconverging paths in a new-PCA measure. This work opens the venue for ongoing improvements.**

I. Introduction

Digital automatic test pattern generation (ATPG) has seen over half a century of developments of algorithms and CAD systems. ATPG is proven to be NP-complete [1], implying the worst-case time complexity to find a test for a fault to be exponential in terms of the circuit size. Practical approaches involve setting time limits and using intuitive heuristics to prioritize search options. To program heuristics, testability measures have been used [2]–[10]. However, no single testability measure is known to work best for all faults [11].

Recent work avoids the dilemma of choosing a testability measure by combining multiple measures by a statistical technique called principal component analysis (PCA) [12]. Any set of testability measures may be combined through PCA. The principal component is then used to guide a PODEM ATPG program's choices at signal nodes during backtraces and forward drives. Our recent work combined several of the well-known measures mentioned in Section II and showed notable improvement in ATPG performance.

The present work examines the questions: Can we keep adding more measures? Is there a limit to the improvement in ATPG performance? In Section IV, the previous results combining the known testability measures are marked as "PCA" and those with additional topological measures as "new-PCA." Conclusions can be found in Section V.

II. Prior Work

Testability analysis identifies test bottlenecks in a circuit. It may determine the controllability and observability of signals, in some approximate form to keep computing complexity low. Various measures such as distance through the circuit, TMEAS [3], SCOAP [4], CAMELOT [10], and COP [6] have been used to improve digital circuit design. However, algorithms like SCOAP and COP have inaccuracies in predicting fault detectability due to their assumption of signal independence. Other methods rely on signal probability computation algebraically [13], or by graph partitioning [8], [9], or by cutting algorithm [14]. Fault detection probability can be computed using probabilistic controllability and observability.

Machine intelligence (MI) applications in this area have begun to emerge [15]–[29]. MI has two key phases: learning from problem-specific data and utilizing that knowledge to solve problems. These phases may be called Artificial Neural Network (ANN) training and ANN guidance in supervised learning. In unsupervised learning, statistical tools like principal component analysis (PCA) [12], [30] are used instead.

The MI tool analyzes the problem-specific data during the first phase to extract relevant characteristics. These characteristics are then leveraged in the second phase to solve problems directly. In other words, the knowledge gained from the data is used to devise solutions to the problems.

Using PCA as the statistical tool, we have applied unsupervised learning to the ATPG problem. This technique extracts the most relevant features from the circuit structure and uses them to guide the test pattern generation. By utilizing this approach, we can efficiently generate test patterns and achieve results superior to those from traditional methods.

A. Dimensionality Reduction

Data is a valuable resource in the digital era, but its storage and computation pose challenges. PCA [12] is a statistical technique that reduces dimensionality by characterizing data into few principal components (PC). The PCs hold the same information as the original data in consolidated form.

We use a singular value decomposition (SVD) method to obtain PCs [31]. A parameter known as the explained variance for a subset of PCs depends on the number of PCs and the individual variance of each PC. Adding up the individual variances of the PCs in the subset gives the total variance, which can be expressed as a percentage of the total explained variance. Often, only the first few PCs are necessary, but in some situations, the last few PCs may be interesting, such as in outlier detection or image analysis.

III. Background and Present Work

1) Putting Together Testability Measures: We generate a composite testability measure using PCA [21] by combining testability measures listed in Section II. Details of this amalgamation process may be found in a recent publication [21]. The present work also includes the addition of primary inputs and outputs (PI and PO) widths, node reconvergences, and even/odd parity of reconvergent fanouts to the list of the conventional testability measures.

For each signal node of the circuit, ten measures and circuit topologies are defined: distance [32], COP-controllability-1 (CC1) [6], COP-observability (CO) [6], SCOAP-controllability-0 (SC0) [4], SCOAP-controllability-1(SC1) [4], SCOAP-observability (SCO) [4], transitive-fan-in affecting PIs (nPIs), transitive-fan-out affecting POs (nPOs), node reconvergence (ReConvNode), and even/odd parity of reconvergent fanouts (parity).

Data initialization sets the values of all testability measures in the same phase to be combined. This is accomplished by a single-pass, linear time complexity procdure:

- Calculate the minimum distance of the circuit node “N” from PI, i.e., *distance*.
- Calculate CC1 for ”N” that determines the probability of setting ”N” to 1. Since it is a probability, the complement of this value can determine the probability of setting ”N” to 0. Also, calculate CO to determine the observability of ”N” at the circuit PO.
- Calculate SC0 and SC1, the effort of setting the node ”N” to 0 and 1, respectively. Also, calculate SCO to determine the effort required to observe the node ”N” at the circuit PO.
- From the fan-in and fanout cones of “N” determine the PIs and POs connected to “N” as nPIs and nPOs, respectively. They are then normalized by dividing, respectively, by the total number of PIs and POs.
- Determine whether or not “N” is a convergent node by tracing all its fanouts to explore whether or not they reconverge (ReConvNode). If they do, the value of “ReConvNode” is set to 1; otherwise, it is kept as 0.
- Examine the numbers of inversions on pairs of fanout paths to determine whether both are even or odd, and select the minimum offset bound reconvergence point. The minimum offset is taken since the impact of backtracking in ATPG is stronger on the minimum offset reconvergence path, and this conflict further ripples toward PIs and POs, making the backtracing and forward tracing more time-expensive due to sky-rocketing undoing of decisions while choosing backtrace path and D-Frontier (see, D-algorithm [33]) in the circuit. If the number of inversions is odd, “parity” is set to 1; otherwise, it is kept as 0. The details of this algorithm are discussed in Section III-A.

Once the relevant values have been computed, they are normalized to 0 to 1 range, and a phase correction is applied as explained in [21]. Finding these values is not computationally intensive. A one-time computation through the circuit can determine all these values, eliminating the need for recalculation during ATPG. This normalization enables the values to be compared and combined meaningfully, avoiding any bias or disproportionate influence of any particular value.

Finally, all measures are combined using PCA, a statistical technique that identifies patterns and correlations in the data. If n measures are combined, then PCA computes n values for each node in the circuit.

In this study, we have two amalgamation tables: a backtracing table (distance, CC1, SC0, SC1, nPIs, ReconvNode, parity) and a forward tracing table (distance, CO, SCO, nPOs, ReconvNode, parity). The largest of these values is the principal component, used as the combined measure for setting the signal values to 0 and 1, respectively.

A. Algorithm for calculating even/odd parity of reconverging fanout

Firstly, we will delve into reconverging fanouts in a circuit node. These are the points where multiple signals converge after branching out from a single source. Next, we will focus on a node in the circuit with several reconverging fanouts.

For each combination of two signals, A and B, we will:

- Identify multiple reconverging points (represented by N = n1, n2, n3,...N) where the two signals converge.
- Select a reconverging point closest to the node according to minimum distance (d) between the node and N.
- Calculate the parities (even/odd) for both reconverging fanouts while traversing from the node to N.
- Evaluate the XOR of the two parity for the reconverging fanouts (represented by X).
- Store the results in a look-up table that comprises d, X.

Finally, we will sort the look-up table in ascending order and choose the parity for the minimum value of ”d.” In this study, we used the minimum distance to calculate the even/odd parity of the reconvergence fanout. This choice was made to account for the impact of odd inversion that a signal may encounter within the reconverging fanouts.

Odd inversion occurs when a signal is inverted an odd number of times as it passes through a circuit. This can cause the signal to change polarity, significantly impacting the system’s performance. If the reconverging point is closest to the reconverging fanout stem, this odd inversion has a more catastrophic effect on the signal than an even inversion.

By using the minimum distance, the number of backtracks is expected to reduce, improving the performance of test generation about CPU time. Backtracks occur when a signal has to retrace its steps to avoid conflicts within the circuit. This can slow down the overall system performance; thus, reducing the number of backtracks is highly desirable. Overall, the choice of minimum distance helps optimize the system’s performance by minimizing the impact of odd inversion and reducing the number of backtracks.

IV. ATPG Experiments: Results and Discussion

The absence of access to the internal details of commercial CAD software necessitated using a home-grown program to compare algorithmic improvements. Our experiments were performed on an Intel-i7-10610U processor with 16 GB RAM. An ATPG system was implemented in C++ using MSVC++14.15 compiler with due emphasis on optimizing performance. The PCA algorithm was executed using Python programming language, whereas PODEM ATPG [32] was implemented with an event-driven fault simulator [34] in C++. The PODEM algorithm was programmed to enable the application of any testability measure including, but not limited to, distance [32], COP [6], SCOAP [4], or PCA, to various benchmark circuits [35], [36].

As the ATPG process is computationally expensive, some faults may be aborted. Nonetheless, it was observed that a comparable fault coverage could be accomplished with each testability measure by setting a suitable per-fault time limit. The proposed ATPG system employed in the present study

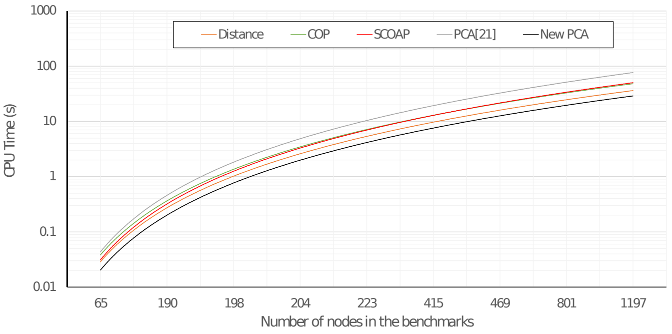

Fig. 1. Total ATPG CPU time for finding a test or proving redundancy for all checkpoint faults left undetected by the random ATPG phase applied to a subset of ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

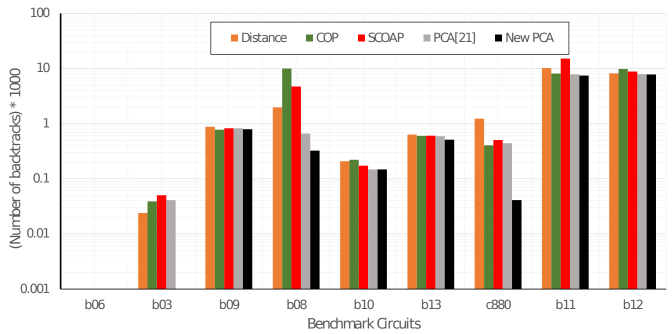

Fig. 2. Total backtracks used while finding a test or proving redundancy for all checkpoint faults left undetected after the random ATPG phase applied to a subset of ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

can identify all checkpoint single stuck-at faults. Furthermore, the system initiates a random pattern detection (RPD) phase to eliminate faults that can be detected using random patterns. The remaining faults are then tested using PODEM ATPG [32], guided successively by a single testability measure, such as distance, COP or SCOAP, or by a combined measure from PCA. Also, a suitable per-fault time limit produces fault coverage similar to all heuristics. The effectiveness of the PCA algorithm was tested in three sets of experiments.

A. Test generation with no aborted faults

This experiment evaluates the performance of the PODEM ATPG algorithm when using various testability measures such as "Distance", "COP", "SCOAP", "PCA [21]", and "new-PCA". The ATPG was applied to nine circuits from ISCAS'85 [35] and ITC'99 [36] benchmarks, b06, b03, b09, b08, b10, b13, c880, b11, b12 (arranged in ascending order of their number of nodes as shown in figures) to test all checkpoint stuck-at faults, using no per-fault time limit across all the testability measures under consideration.

Results of ATPG CPU time and number of backtracks are shown in Figures 1 and 2. Note that circuit b06 has no backtracks for all testability measures. Moreover, the new-PCA guided ATPG outperformed other testability-measure-based methods regarding quality. This is evident as the *new-PCA*-based ATPG reduced the backtracks (even to 0 for b03) compared to other testability measure-based ATPG and substantially reduced the ATPG CPU time.

The results also show that the new-PCA guided ATPG required lower ATPG CPU time across circuits b09, b08, b10, b13, c880, b11, and b12, with a substantial reduction in backtracks except for b09, b10, b11, b12, and b13 where the reduction in backtracks was not substantial but still not higher than the other testability measure-based ATPG.

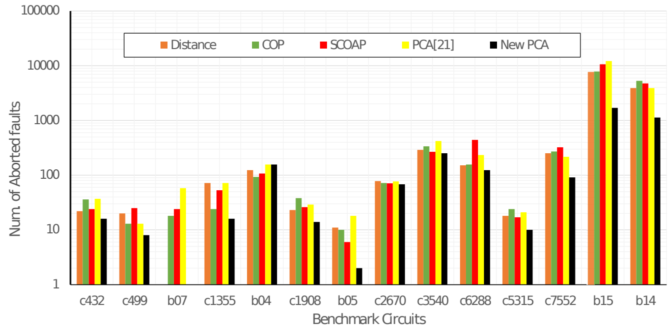

Fig. 3. Total aborted faults after the random and deterministic ATPG phase applied to ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

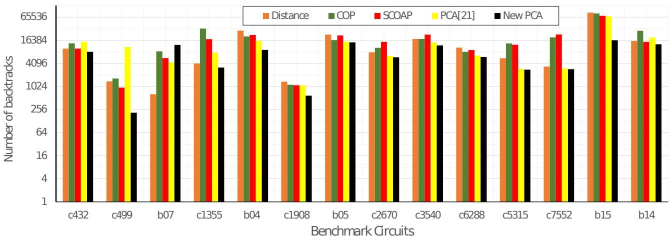

Fig. 4. Total backtracks used while finding a test or proving redundancy for all checkpoint faults left undetected after the random ATPG phase applied to ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

This study demonstrates that the PCA-based ATPG method is an effective and efficient way to perform ATPG and improve the quality of backtracing. This could be a significant breakthrough in testing methodology, as it may detect all faults without any aborted faults or backtracks for specific circuits.

It is worth noting that the improvement in backtracing quality is a significant achievement in itself. Other testability-measure-based ATPG methods can lead to backtracks for faults, even in circuits with no redundant faults. However, this study's PCA-based ATPG method ensures that all faults can be detected with no aborted and unidentified redundant faults and with zero backtracks for specific circuits, making it an essential method for any testing methodology.

B. Test generation with per-fault time limit

In an ATPG run, aborted faults often appear when a per-fault CPU time limit is set. For these faults test generation is aborted because within the set time limit neither a test is found not the fault is proven redundant.

A typical ATPG tool may not target aborted faults by default. However, it could retarget them using dynamic learning the second time. Such a strategy would significantly increase run times. Our experiment evaluates the performance of PODEM ATPG [32] using various testability measures such as "Distance", "COP", "SCOAP", "PCA [21]", and "new-PCA" to observe the impact on the detection of aborted faults, ATPG CPU time, and number of backtracks, and most importantly impact on fault coverages as illustrated, respectively, in Figures 3, 4, 5 and 6.

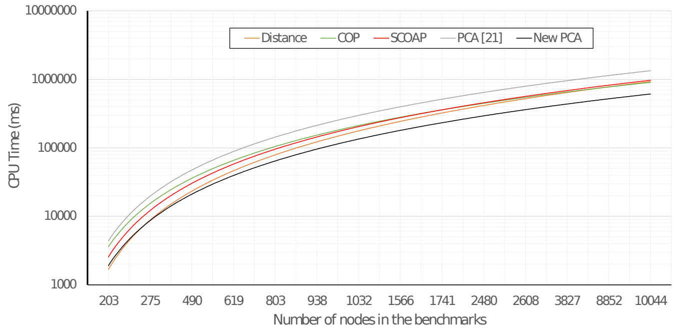

Fig. 5. Total ATPG CPU time for finding a test or proving redundancy for all checkpoint faults left undetected after the random ATPG phase applied to ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

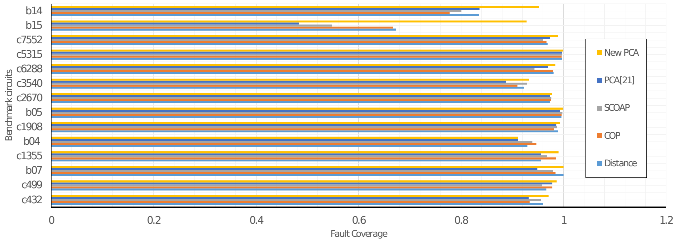

Fig. 6. Total fault coverages while finding a test or proving redundancy for all checkpoint faults of ISCAS'85 [35] and ITC'99 [36] benchmark circuits.

The ATPG was applied to fourteen circuits from the ISCAS'85 [35] and ITC'99 [36] benchmarks, c432, c499, b07, c1355, b04, c1908, b05, c2670, c3540, c6288, c5315, c7552, b15, b14 (arranged from left to right according to the number of nodes) to test all checkpoint stuck-at faults.

We observe a constant reduction in ATPG CPU time and substantially smaller numbers of backtracks and aborted faults, as well as a significant increase in fault coverages, with few exceptions. More prominence can be found in circuits comprising more nodes, shown on extreme right. In a circuit like b07, the new-PCA reduced the number of aborted faults to zero but at the cost of more backtracks and ATPG CPU time with a lesser impact on the tool efficiency. Also, in a circuit like b04, fault coverage is not improved; however, it has reduced backtracks and CPU time (not a visually promising reduction).

C. Test Generation for hard-to-detect faults in larger circuits

This experiment examines the effectiveness of the new-PCA method in detecting difficult-to-detect faults in more complex and extensive circuits. We ran a third-party ATPG tool on three circuits, b20, b21, and b22, to obtain lists of aborted faults. Then we extracted 100 hard-to-detect faults for each circuit based on COP probabilities [6]. Next, we applied our PODEM ATPG system using various testability measures, "Distance", "COP", "SCOAP", "PCA [21]", and "new-PCA" to observe the impact on the detection of aborted faults, ATPG CPU time, number of backtracks, and most importantly, the impact on fault coverages. The results are illustrated in Figures 7, 8, 9, and 10, respectively.

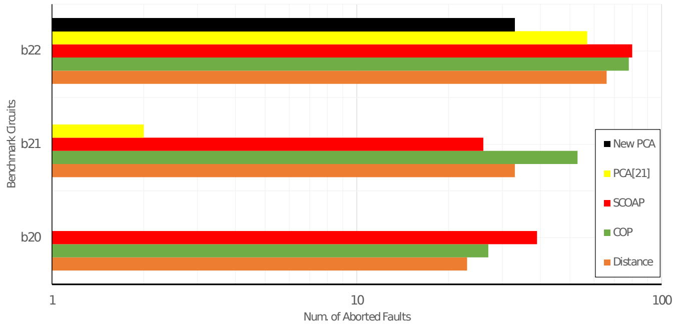

Fig. 7. Total aborted faults after the deterministic ATPG phase applied to 100 hard-to-detect faults in three ITC'99 [36] benchmark circuits.

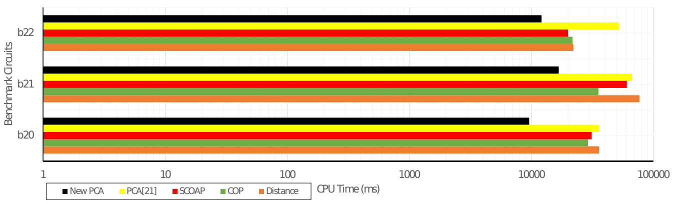

Fig. 8. Total ATPG CPU time for finding a test or proving redundancy for 100 hard-to-detect faults in three ITC'99 [36] benchmark circuits.

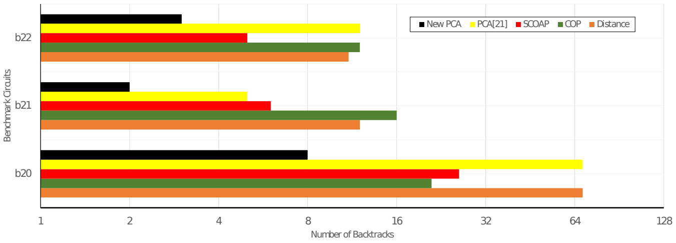

Fig. 9. Total backtracks used while finding a test or proving redundancy for 100 hard-to-detect faults in three ITC'99 [36] benchmark circuits.

Our study finds that using new-PCA reduced backtracks, ATPG CPU time, and aborted faults, and all together increased the fault coverage. This indicates that the method discussed in this study can detect much harder faults in larger circuits that a third-party (commercial) tool with preset (default) bounds and limits may not detect.

V. Conclusion and Future Directions

This study investigates whether a PCA-based combination of testability measures and circuit topological features can enhance the efficiency of ATPG programs. Clearly, this approach can significantly reduce the number of backtracks, reduce the ATPG CPU time, and improve the detection of hard-to-detect faults. It increased the fault coverages for the ISCAS'85 and ITC'99 benchmarks and provided more robust detection of aborted faults than traditional heuristic-based ATPG. The new method is likely to be scalable when applied on larger

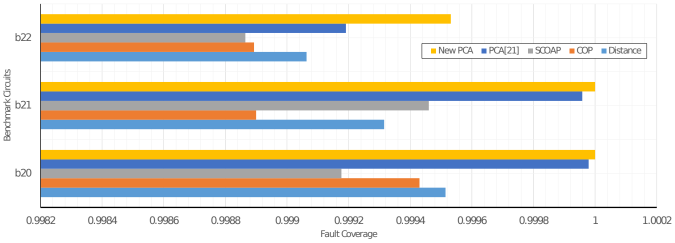

Fig. 10. Total fault coverages while finding a test or proving redundancy for 100 hard-to-detect faults in three ITC'99 [36] benchmark circuits.

industrial circuits or applied to other fault models, detect user-defined faults, transition faults, cell-aware stuck-at faults more efficiently, and provide greater fault coverages with raster ATPG runs.

This research can be the driving force behind developing silent data error detection and can be an incremental upgrade to any current third-party ATPG tool. Hence, this method can be an excellent addition to current third-party ATPG tools, improving their quality.

The study has demonstrated that the PCA-based ATPG method is a reliable and efficient way to improve the quality of backtracing followed by a reduction in ATPG CPU time, detection of otherwise aborted faults, and providing higher fault coverage. These are a significant step forward in testing various NP-hard methodologies that can help create more robust and efficient testing methods for electronic systems. This research can be the foundation for future testing methodologies and provide a more reliable and efficient way to ensure the quality of various electronic systems.

We should address the questions posed in Section I, "Can we keep adding more measures?" For now, the answer is "yes", as new-PCA guidance reduced CPU time and increased fault coverage. The question of the limit to ATPG performance is still not answered. Will the performance increase continue or asymptotically reach a plateau? May need investigation on what new measures? Perhaps an answer could be found in a closer examination of backtracks.

This paper shows that ATPG performance improves when topological factors are added to backtrace guidance. Can such improvement continue with more data about the circuit? What type of data? Will backtracks continue to drop? To zero?

References

[1] O. H. Ibarra and S. K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Transactions on Computers*, vol. C-24, pp. 242–249, 1975.

[2] J. Stephenson, "A Testability Measure for Register-Transfer Level Digital Circuits," Ph.D. dissertation, Carneigie-Mellon Univ., 1974.

[3] J. Grason, "TMEAS, A Testability Measurement Program," in *16th Design Automation Conference*, 1979, pp. 156–161.

[4] L. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Transactions on Circuits and Systems*, vol. CAS-26, no. 9, pp. 685–693, Sep. 1979.

[5] V. D. Agrawal and M. R. Mercer, "Testability Measures – What Do They Tell Us?" in *Proc. International Test Conf.*, Philadelphia, PA, Nov. 1982, pp. 391–396.

[6] F. Brglez, "On Testability Analysis of Combinational Circuits," *Proc. International Symp. Circuits and Systems*, pp. 221–225, 1984.

[7] S. K. Jain and V. D. Agrawal, "Statistical Fault Analysis," *IEEE Design & Test of Computers*, vol. 2, no. 1, pp. 38–44, 1985.

[8] S. C. Seth, B. B. Bhattacharya, and V. D. Agrawal, "An Exact Analysis for Efficient Computation of Random-Pattern Testability in Combinational Circuits," in *Proc. Fault Tolerant Computing Symposium (FTCS)*, Vienna, Austria, Jul. 1986, pp. 318–323.

[9] S. C. Seth and V. D. Agrawal, "A New Model for Computation of Probabilistic Testability in Combinational Circuits," *Integration*, vol. 7, no. 1, pp. 49–75, 1989.

[10] R. G. Bennetts, C. M. Maunder, and G. D. Robinson, "CAMELOT: A Computer-Aided Measure for Logic Testability," *IEE Proceedings E - Computers and Digital Techniques*, vol. 128, no. 5, pp. 177–189, 1981.

[11] J. Patel and S. Patel, "What Heuristics are Best for PODEM?" in *Proc. First International Workshop on VLSI Design*, 1985, pp. 1–20.

[12] H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[13] K. Parker and E. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Transactions on Computers*, vol. C-24, no. 6, pp. 668–670, 1975.

[14] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability," *IEEE Transactions on Computers*, vol. C-33, no. 1, pp. 79–90, Jan. 1984.

[15] Y. Sun and S. K. Millican, "Test Point Insertion Using Artificial Neural Networks," in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 253–258.

[16] Y. Sun, "Novel Test Point Insertion Applications in LBIST," Ph.D. dissertation, Auburn University, USA, 2021.

[17] S. K. Millican, Y. Sun, S. Roy, and V. D. Agrawal, "Applying Neural Networks to Delay Fault Testing: Test Point Insertion and Random Circuit Training," in *Proc. IEEE 28th Asian Test Symposium (ATS)*, 2019, pp. 13–18.

[18] Y. Sun and S. K. Millican, "Applying Artificial Neural Networks to Logic Built-in Self-test: Improving Test Point Insertion," *Journal of Electronic Testing: Theory and Applications*, vol. 38, no. 4, pp. 339–352, Oct. 2022.

[19] S. Roy, S. K. Millican, and V. D. Agrawal, "Principal Component Analysis in Machine Intelligence-Based Test Generation," in *Proc. IEEE Microelectronics Design and Test Symp. (MDTS'21)*, USA, May 2021, pp. 1–6.

[20] S. Roy, "Toward Zero Backtracks in Test Pattern Search Algorithms with Machine Learning," Ph.D. dissertation, Auburn University, USA, 2021.

[21] S. Roy and V. D. Agrawal, "An Amalgamated Testability Measure Derived from Machine Intelligence," in *Proceedings of 37th International Conference on VLSI Design & 23rd International Conference on Embedded Systems*, Jan. 2024, pp. 696–701.

[22] S. Roy, B. Stiene, S. K. Millican, and V. D. Agrawal, "Improved Pseudo-Random Fault Coverage Through Inversions: a Study on Test Point Architectures," *J. Electron. Testing: Theory and Applic.*, vol. 36, no. 1, p. 123–133, Feb. 2020.

[23] S. Roy, S. K. Millican, and V. D. Agrawal, "Machine Intelligence for Efficient Test Pattern Generation," in *Proceedings of the IEEE International Test Conference*, Washington D.C, Nov. 2020, pp. 1–5.

[24] S. Roy, S. K. Millican, and V. D. Agrawal, "Training Neural Network for Machine Intelligence in Automatic Test Pattern Generator," in *Proceedings of 34th International Conference on VLSI Design & 20th International Conference on Embedded Systems*, 2021, pp. 316–321.

[25] S. Roy, S. K. Millican, and V. D. Agrawal, "Unsupervised Learning in Test Generation for Digital Integrated Circuits," in *Proceedings of the IEEE European Test Symposium*, 2021, pp. 1–4.

[26] 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 *Proc. IEEE VLSI Test Symp. (VTS'21)*, USA, Apr. 2021, pp. 1–10.

[27] S. Roy, S. K. Millican, and V. D. Agrawal, "Multi-Heuristic Machine Intelligence Guidance in Automatic Test Pattern Generation," in *Proc. 31st Microelectronics Design and Test Symposium (MDTS)*, 2022, pp. 1–6.

[28] S. Millican, Y. Sun, S. Roy, and V. Agrawal, "System and Method for Optimizing Fault Coverage Based on Optimized Test Point Insertion Determinations for Logical Circuits," U.S. Patent 17226950, Oct. 2021.

[29] S. Roy, S. K. Millican, and V. D. Agrawal, "A Survey and Recent Advances: Machine Intelligence in Electronic Testing," *Journal of Electronic Testing: Theory and Applications*, vol. 40, pp. 139–152, 2024.

[30] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[31] I. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 2002.

[32] 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, Mar. 1981.

[33] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 5, pp. 567–580, Oct. 1967.

[34] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer Publishing Company, Incorporated, 2013.

[35] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," *Proceedings of the IEEE Int. Symposium on Circuits and Systems (ISCAS)*, pp. 677–692, Jun. 1985.

[36] F. Corno, M. S. Reorda, and G. Squillero, "RT-Level ITC'99 Benchmarks and First ATPG Results," *IEEE Design & Test of Computers*, vol. 17, pp. 44–53, Jul. 2000.