2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID) | 979-8-3503-8440-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSID60093.2024.00123

An Amalgamated Testability Measure Derived from Machine Intelligence

Soham Roy
Design for Test Engineering Group
Intel Corporation, Santa Clara, CA 95054
soham.roy@intel.com

Vishwani D. Agrawal
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849-5201
agrawvd@auburn.edu

***Abstract*—A testability measure provides test-related information about signal nodes of a circuit. Operations like test generation and test point insertion are exponentially complex in terms of the circuit size. Therefore, to be useful testability measure computation is kept linear, which makes the measures like controllabilities and observabilities, approximate. Well-known testability measures like SCOAP (Sandia controllability/observability analysis program) or COP (controllability and observability program) have played important roles in algorithms for test generation, test point insertion, and other test-related functions. Even the quantities such as distances of a node to primary input and output have been used as simple measures. Years of experience have shown that no single measure works for all situations – in test generation SCOAP may work best for one fault, while COP or distance do better for other faults in the same circuit. This study amalgamates all three measures mentioned here using the principal component analysis (PCA), an unsupervised machine learning procedure. This amalgamated measure, when used by a test generation program produced unexpected benefits. First, the measure reduced the test generation program backtracks for hard-to-detect faults below those by any single testability measure; the number of backtracks is a direct indicator of computing effort. Second, the backtracks reduced to 0 for several faults. This study tries to prove the efficacy of amalgamation by running categorical experiments, namely, testability analysis accuracy, ATPG improvements, and faults classification. In the continuing investigation, we plan to combine more testability measures into PCA. We will also investigate applications like test-point insertion and other test related functions in the future.**

***Index Terms*—ATPG, Backtrace, COP, Digital testing, Machine intelligence (MI), Machine learning (ML), PODEM, Principal component analysis (PCA), SCOAP, Testability measures, Unsupervised learning.**

I. Introduction

Testability measures occupy a full chapter in a book on testing [1]. So what is new? This paper discusses the improvements machine intelligence is bringing.

It began in 1960s. Electronic circuits were moving from discrete component assemblies to integrated circuits. With increasing levels of integration the capability of probing signal nodes declined. Thus, arose new test problems. Focusing attention to digital circuits, we note that faults modeled at all signal nodes had to be tested by inputs applied at primary inputs (PI) and observing responses at primary outputs. Although algorithms [2]–[4] could find test inputs, generation of some tests took too long. No surprise, soon it was proven that the worst-case complexity of test generation is exponential in the circuit size [5]. Typically, the design and verification of a circuit would be followed by test generation, and if a large set of faults have close to worst-case test generation complexity, the design is modified for improved testability and re-verified before test generation.

There are two difficulties with this scenario. First, the recycling of design process involves time and cost, and, second, improving testability requires a review of the circuit structure by experts who are in short supply, especially since circuit applications are spreading fast like wild fire.

Testability measures [6]–[14] provide a solution. These are defined for each node as *controllability* and *observability*, two parameters adopted from control theory [15]. A testability measure interprets the parameter as either *effort*, or probability, or simply as logic depth. To keep the computation simpler than actual test generation, the measures use approximation, making them inexact. As a result, a measure may work well for some faults and not not so for others. One measure may work well for shallow circuits while other may favor deep logic structures. Similarly, one measure may deal better than others for reconverging fanout signals.

Interpretations [16], applications [17], limitations and improvements [18] were explored, resulting in many testability measures. A useful application has been automatic test pattern generation (ATPG), where testability measure guides the heuristic choices. Since no measure is perfect, it was suggested that ATPG may be repeated, each time with a different testability measure [19], [20]. This makes the program execution somewhat clumsy. However, the situation improves if several measures can be combined [21]–[26].

The present work gives a procedure to amalgamate several testability measures into a single measure. It is a two step process. First, all measures are set in the same phase and normalized to the same range. To understand the phase, consider two measures, COP [9] and SCOAP [8]. In COP a difficult to observe node will have a small probability of observing at a PO, or a low observability value. On the other hand, SCOAP will show high observability value representing the effort of observing. As explained in Section IV, the measures are subjected to phase-matching and normalization. The second step then combines the measures by principal component analysis (PCA), a statistical procedure used in *unsupervised learning*.

This article is organized as follows. Section II summarizes testability measures. Section III explains some essentials of PCA. Section IV gives the process of amalgamating testability measures. In Section V, a PODEM [3] ATPG guided by the amalgamated testability measure is compared with those guided by conventional single testability measures on selected samples of faults and on fault groups that have not been covered by random patterns. Section VI suggests future work as we move forward, and Section VII winds up the article.

II. Prior Work

Early work [8] developed a linear complexity algorithm called Sandia controllability/observability analysis program, commonly known as the SCOAP testability measure. It defined 0 and 1 controllabilities for each signal determining the effort (or difficulty) of setting the line to a logic 0 and 1, respectively. Another algorithm COP [9] provided a single-pass probabilistic testability measure. The 1 or 0-controllability is the probability of signal on line *l* being set to 1 or 0 by a random input vector. Some approximations used in the aforementioned testability measures have been examined over the years. Both SCOAP and COP have significant inaccuracies due to the assumption that signals at *reconvergent fanout* stems are independent, which makes them inaccurate at predicting detectability of faults.

A higher accuracy signal probability computing algorithm [27] used an algebraic procedure for line controllabilities. Another procedure, PREDICT [11], [12], proposed a graph-partitioning of the circuit into *supergates* that include reconvergent fanouts, of course, at the expense of computational cost. The cutting algorithm [28] cuts selected fanout lines to make the circuit a tree structure and then initializes the cut lines to a controllability range [0,1]. The modified network has no reconvergent fanout and controllability bounds can be easily computed for all lines. For some signals these bounds may not be narrow enough to be useful. To overcome this disadvantage, COP [9] provided a probability based testability measure that maintains computational efficiency by neglecting signal correlations. A later analysis of the error in detection probability calculation [29] concluded that probabilities of control and observation of a line cannot be multiplied since those are not independent events. The statistical fault analysis procedure, STAFAN [10], defined 0-observability and 1-observability of a line *l* as probabilities of the line being observed with value 0 or 1, respectively. Now, the observabilities are conditional and, therefore, they can be multiplied by appropriate controllabilities, without error, to obtain fault detection probabilities. Other authors [30], [31] used the conditional observabilities to obtain exact detection probabilities. Also worth mentioning are a fast testability analysis program [18] and a high-level testability measure [32]. Applications of machine intelligence in this area have also begun [33]–[36].

Any single testability measure can help provide heuristics for ATPG and test point insertion (TPI) algorithms. Various noteworthy theories [5] show that the ATPG and TPI for combinational circuits belong to the class of NP-complete problems, which means having greater than polynomial computation time complexity. *Heuristic* search techniques are used in ATPG for efficiency [34] and in TPI for superior testpoints (TPs) indicated by higher fault coverage and reduced TPI time [37]–[39]. Recent work [40] presented an ANN-based signal probability predictor for VLSI circuits considering reconvergent fanouts.

III. Principal Component Analysis (PCA)

As data mining and pruning techniques address expanding storage and computation challenges, PCA applications grow. The PCA is classified as *unsupervised learning* in the field of machine intelligence. Although PCA was discovered a century ago [41], [42], its usage expanded when multiple disciplines began to use computer-based applications. It is a dominant data dimension reduction tool that transforming data into principal components (PC). Each PC has linear pivoting on the original dataset that maximizes variance of uncorrelated data while keeping statistical information intact. PCs are evaluated from the original data using single value decomposition (SVD) [43] to choose PCs based on either correlation or covariance matrix. When we apply PCA, it is important to understand how much of data variation is delineated or explained by each PC. There enters the concept of *explained variance* that measures the variance's proportion in the data, explained by each PC. "Explained variance" is a measurement in statistics that tells us how much variation in a dataset can be contributed to each of the PCs (eigenvectors) originated by the PCA method. Simply explained, variance refers to data variability in the dataset that can be contributed to the individual PC. This statistical parameter is vital for ranking the PCs in order of importance while incorporating them in our study of this article.

In our application, PCA combines multiple testability measures, shrinks the entire dataset comprising of testability measures of the entire circuit, and produces PCs with orthogonal characteristics from individual testability measure. This kind of unique potential of a novel MI-based testability measure was never explored before and its efficacy is worth observing in different applications. One such application is ATPG where it shows a remarkable improvement in performance and in detecting faults of varying nature quickly, and with no or near-zero backtracks.

IV. The Amalgamation Methodology

In general, any number of testability measures can be amalgamated (combined) by an unsupervised machine learning procedure using PCA. In the present work, we combine three measures, distance [3], COP [9], and SCOAP [8], specified below for each signal in the circuit:

Distance [3]: Two distances, d_{PI} and d_{PO} are logic depths in terms of number of logic gates on shortest paths to primary inputs (PI) and primary outputs (PO), respectively.

COP [9]: Estimated probabilities for signal values 0 and 1 are called combinational controllabilities, $CC0$ and $CC1$, respectively, assuming a random input is applied to the circuit. Since $CC0 = 1 - CC1$, we consider just one quantity, i.e., $CC1$.

SCOAP [8]: Estimated efforts are SCOAP combinational controllabilities $SC0$ and $SC1$, and SCOAP combinational observability SCO, needed for setting the signal to 0 and 1, and observing its value at PO, respectively.

The combination process has following steps:

- For testability measures, e.g., distance [3], COP [9], and SCOAP [8], to be combined, compute relevant values corresponding to each signal node in the circuit.
- All measures are numerical and positive. We normalize each to the range [0,1].
- Phase correction - Consider SCOAP, which is a measure of effort. Thus, low or closer to zero 0-controllability means that the node is easy to set to 0. On the other hand, COP [9] estimates probability and for the same

TABLE I

INPUT IS SELECTED BY BACKTRACING THROUGH A GATE USING TESTABILITY MEASURES. THE ENTRIES WITH BOLDFACE REPRESENT CONFLICTING CRITERIA AGAINST d_{PI} (USED AS REFERENCE). THESE BOLDFACED VALUES OF CONFLICTING TESTABILITY MEASURES ARE COMPLEMENTED BEFORE APPLYING PCA ONTO THEM AND OUTPUT PRINCIPAL COMPONENTS P_0 AND P_1.

Gate type	Output value	Various Testability Measures				PCA	
		d_{PI}	$CC1$	$SC0$	$SC1$	P_0	$P1$
AND	0	min	min	min	**MAX**	min	
	1	max	**MIN**	**MIN**	max		max
OR	0	max	max	max	**MIN**	max	
	1	min	**MAX**	**MAX**	min		min
NAND	0	max	**MIN**	**MIN**	max	max	
	1	min	min	min	**MAX**		min
NOR	0	min	**MAX**	**MAX**	min	min	
	1	max	max	max	**MIN**		max

TABLE II

HEURISTIC SELECTION OF D FOR D-DRIVE FROM D-FRONTIER. VALUES WITH BOLDFACE ARE HAVING CONFLICTS THAT CAN RESOLVED BY TAKING COMPLEMENT BEFORE AMALGAMATING VIA PCA TO P_D. d_{PO} IS TAKEN AS REFERENCE.

Selection criterion for D-drive w.r.t. testability measures			
Distance d_{PO}	COP CO	SCOAP SO	PCA P_D
min	**MAX**	min	min

node the 0-controllability will be closer to 1. Assuming that the combined measure is to have the probability interpretation, the normalized SCOAP values should be complemented (subtracted from 1.0) in order to align with other measures. This process is given in Table I for 0 and 1-controllabilities to be used in the backtrace of ATPG, and in Table II for observability to be used in D-drive. We will refer to it as the min-max criterion, where a boldface **MIN** or **MAX** indicates that the measure should be complemented (subtracted from 1.0). Here distance is taken as the reference. For example, if an AND gate output is to be justified as 1, then all inputs will have to be 1, and the backtrace should proceed through the untraced AND-gate input with largest d_{PI}. Thus, $CC1$ and $SC0$ should be complemented and $SC1$ left unchanged.

- All measures are combined using the principal component analysis (PCA) [77]. If n measures are being combined, then PCA computes n values for each node of the circuit. The largest of these is the principal component and used as the combined measure. The analysis is repeated three times to generate the combined 0-controllability, 1-controllability and observability for each node.

P_0: Four-dimensional data i.e., d_{PI}, $CC1$, $SC0$, $SC1$ of each signal node in a circuit is combined by PCA. For each node of the circuit, a row in Table I corresponds to gate type whose output is being set to logic 0. Min-max criterion shown in bold face in the table requires complementing the measure. PCA produces four principal components, P0#1, P0#2, P0#3 and P0#4. P0#1 (or P_0) is selected as major principal component having maximum explained variance, i.e., carrying orthogonal features (Fig. 1).

P_1: Four-dimensional data i.e., d_{PI}, $CC1$, $SC0$, $SC1$ of each signal node in a circuit is combined by PCA. For each node of the circuit, a row in Table I corresponds to gate type whose output is being set to logic 1. Min-max criterion shown in bold face in the table requires complementing the measure. PCA produces four principal components, P1#1, P1#2, P1#3 and P1#4. P1#1 (or P_1) is selected as major principal component having maximum explained variance, i.e., carrying orthogonal features (Fig. 2).

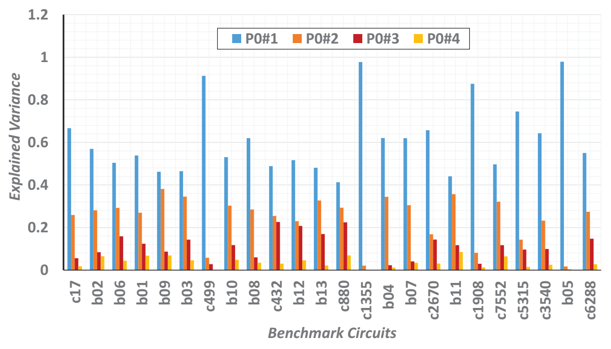

Fig. 1. PCA for backtracing 0 in ISCAS'85 and ITC'99 benchmarks. Logic 0 output state is assumed for all gates during PCA. Items with conflicting criteria, shown in bold in Table I, were complemented. Four PCs are P0#1, P0#2, P0#3 and P0#4. The major PC P0#1, tallest bars shown in blue, are selected as P_0 of Table I.

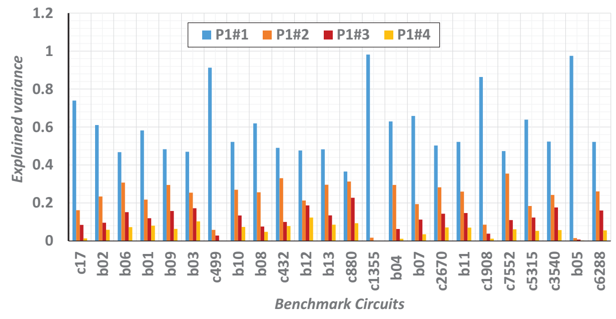

Fig. 2. PCA for backtracing 1 in ISCAS'85 and ITC'99 benchmarks. Logic 1 output state is assumed for all gates during PCA. Items with conflicting criteria, shown in bold in Table I, were complemented. Four PCs are P1#1, P1#2, P1#3 and P1#4. The major PC P1#1, tallest bars shown in blue, are selected as P_1 of Table I.

P_D: Three-dimensional data i.e., d_{PO}, CO, SO of each signal node in a circuit is combined by PCA. For each node in the circuit, Table II provides a min-max criterion (bold face means complement the measure). PCA produces three principal components, P0#1, P0#2, and P0#3. P0#1 (or P_D) is selected as major principal component having maximum explained variance, i.e., carrying orthogonal features (Fig. 3).

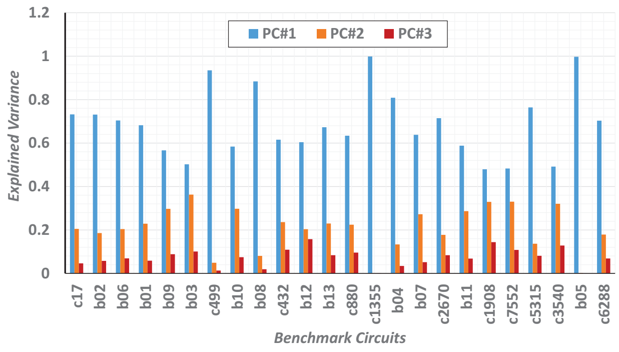

Fig. 3. PCA for directing D-drive in ISCAS'85 and ITC'99 benchmarks. Items with conflicting criteria, shown in bold in Table II, were complemented. Three PCs are PC#1, PC#2 and PC#3. The major PC PC#1, tallest bars shown in blue, are selected as P_D of Table II.

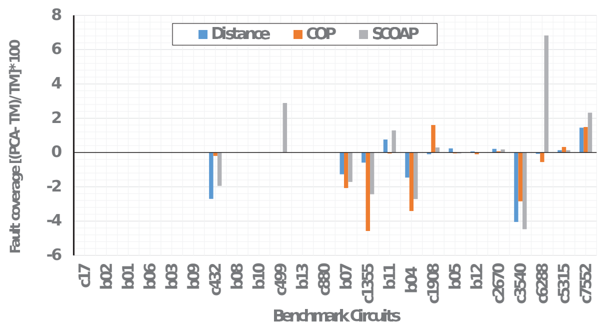

Fig. 4. Comparison of fault coverage when a single testability measure (TM) guides backtracing as opposed to amalgamated testability measure (PCA) to detect testable and redundant checkpoint faults left over from random pattern detection. PCA provides steady and often considerable benefits; positive or no bars show improved or same fault coverage by PCA as by a single measure.

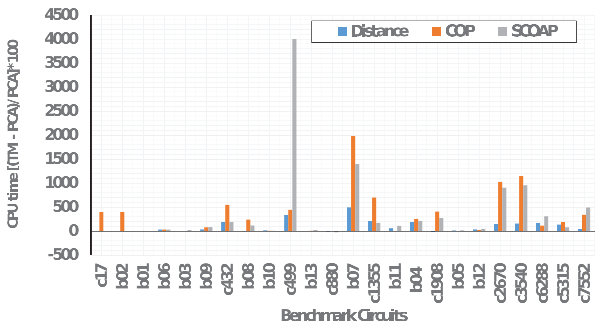

Fig. 5. Comparison of CPU time when single testability measure (TM) is applied to PODEM backtrace as opposed to amalgamated testability measure (PCA) to detect the left over testable and redundant checkpoint faults after random pattern detection. Clearly, PCA provides steady and often considerable benefit; positive bars show reduced CPU time by PCA over a single measure.

V. Experimental Results

We hope our study will inspire electronic design automation (EDA) vendors to incorporate MI in there ATPG programs. Without access to internal details of commercial software, we restricted our experiments to in-house EDA tools for comparing algorithmic improvements.

All experiments were run on Intel-8700 processor and 8 GB RAM workstation. This EDA software was implemented in C++ using MSVC++14.15 compiler with optimizing performance. The PCA was executed using python and PODEM ATPG [3] was implemented with an event-driven simulator [1]. The PODEM was programmed such that any testability measure, distance [3], COP [9], SCOAP [8], or PCA, could be applied to benchmarks [44], [45]. Since the ATPG is time expensive, some faults may be aborted. Almost similar fault coverage could be obtained with each testability measure by implementing appropriate per-fault time limit.

Our ATPG system works on all checkpoint single stuck-at faults and has an intial phase of random pattern detection (RPD) using a fault simulator. Only the faults not detected by random patterns are supplied to PODEM ATPG guided successively by a single testability measure, distance, COP, or SCOAP, and by amalgamated (combined) measure referred to as PCA. To establish the efficacy of PCA, three sets of experiments are recorded in the following subsections.

A. Testability Measure Accuracy

These experiments are also carried out on faults left over from random pattern detection to examine the effectiveness of amalgamating testability measure (PCA) in PODEM ATPG application. Per-fault ATPG time limit for each single measure guidance as well as for PCA guidance was kept constant. Results in Fig. 4 show zero relative change in fault coverages in most cases (no bars), some higher coverages by PCA (positive bars), and a few dropped coverages (negative bars).

B. ATPG Performance Improvement

The next set of experiments examines the performance in terms of CPU time and backtracks for PODEM ATPG guided by single testability measures (TM) and the amalgamated measure (PCA). Figures 5 and 6 show the results where, as before, circuits are arranged left to right in order of increasing

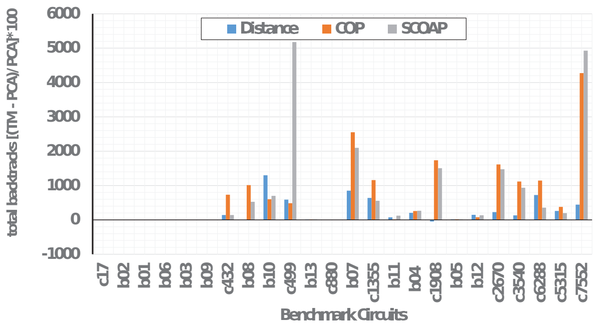

Fig. 6. Comparison of total backtracks when single testability measure (TM) is applied to PODEM backtrace as opposed to amalgamated testability measure (PCA) to detect the left over testable and redundant checkpoint faults after random pattern detection. Again, PCA provides steady and often considerable benefits, positive bars show reduced backtracks by PCA over a single measure.

node count. Target faults are all checkpoint stuck-ats left over from random ATPG. Result for each circuit contains three bars showing the CPU time (Fig. 5) or total number of backtracks (Fig. 6) corresponding to TM, normalized percentage with respect to the same quantity for PDA.

Results illustrate that multiple single testability measures amalgamated as a linear combination effectively brings down backtracks and ATPG CPU time over any single testability measure. Circuits b03, c432, b10, b13, c880, b07, b05, b12, c5315, c7552, c1355, c2670, c3540, b04, b11, b08, c499 and c6288 had significant lowering of backtracks and CPU times (Figures 5 and 6). PCA is frequently the best testability measure, but even when it is not, it is never the worst. There are no backtracks in c17, b02, b01, b06, and hence there is no scope for improvement by PCA. However, the elusive goal of achieving zero backtracks by amalgamated testability measure-based PODEM ATPG is achieved for b09 as shown in Fig. 6; the percent reduction in CPU time is shown as positive bars in Fig. 5, but the percent reduction to zero backtracks could not be shown.

C. Fault Classes

We used samples of easy-to-detect (ETD), hard-to-detect (HTD), and redundant (RED) faults in circuits c6288 and

TABLE III
BACKTRACKS IN PODEM ATPG WITH HEURISTIC GUIDANCE FROM DISTANCE, COP, SCOAP, AND PCA FOR 12 SAMPLE FAULTS (HARD-TO-DETECT, EASY-TO-DETECT, AND REDUNDANT) OF c6288.

Testability class of fault	**Backtracks for various guidance data**			
	Distance	**COP**	**SCOAP**	**PCA**
Hard-to-detect #1	128	1	129	**0**
Hard-to-detect #2	64	10	64	**0**
Hard-to-detect #3	3	10	1	**0**
Hard-to-detect #4	2	10	3	**0**
Hard-to-detect #5	3	10	128	**0**
Hard-to-detect #6	3	10	1	**0**
Hard-to-detect #7	2	4	3	**1**
Easy-to-detect #1	3	1	4	**0**
Easy-to-detect #2	2	9	1	**0**
Easy-to-detect #3	3	1	6	**0**
Redundant #1	7	7	4	**3**
Redundant #2	11	10	7	**4**

TABLE IV
BACKTRACKS IN PODEM ATPG WITH HEURISTIC GUIDANCE FROM DISTANCE, COP, SCOAP, AND PCA FOR 12 SAMPLE FAULTS (HARD-TO-DETECT, EASY-TO-DETECT, AND REDUNDANT) OF b07.

Testability class of fault	**Backtracks for various guidance data**			
	Distance	**COP**	**SCOAP**	**PCA**
Hard-to-detect #1	122	720	105	**56**
Hard-to-detect #2	86	92	94	**80**
Hard-to-detect #3	36	170	68	**0**
Hard-to-detect #4	2	978	2	**0**
Hard-to-detect #5	22	19	154	**5**
Hard-to-detect #6	2	2	2	**0**
Hard-to-detect #7	13	2	2	**0**
Easy-to-detect #1	1	26	7	**0**
Easy-to-detect #2	1	22	9	**0**
Easy-to-detect #3	2	250	2	**0**
Redundant #1	94	92	94	**80**
Redundant #2	98	92	96	**86**

b07 to evaluate the efficacy of guidance by amalgamated testability measure in ATPG application through a fault-by-fault approach. Tables III and IV show reduced number of backtracks (almost zero in many cases) to detect ETD, HTD, and RED faults. Another interesting observation is that the number of backtracks for redundant faults never drops to zero, which confirms that at least one backtrack is needed to finish off the search for such a fault.

This experiment provokes some interesting thoughts. With amalgamated testability measure, a circuit can either have detectable faults (fast enough in terms of detection) or redundant faults (fast enough to declare a fault as redundant) since PCA-guided ATPG can reduce backtracks to zero for both HTD and ETD faults. This will remove the demarcation of easy or hard faults and boils down the fault classification list to only two categories of faults, i.e., detected faults and redundant faults. However, this experiment points to a prominent disadvantage, i.e., one cannot classify faults in a circuit without applying ATPG. That could change since amalgamated testability measure-guided ATPG runtime is substantially low.

VI. Discussion and Future Work

This study entails experiments mentioned above whose conclusiveness can be gauged by their empirical data. The CPU time in Fig. 5 and total number of backtracks in Fig. 6 for ATPG reduced steadily across benchmarks when ATPG was guided by the amalgamated testability measure as opposed to a single testability measure. Therefore, it can be concluded that amalgamated testability measure may be the best compared to single testability measure. Research and development to significantly advance the state-of-the-art has always been a constant thrive in the VLSI industry and this is the first time an MI-based testability measure is shown to radically improve the detection of hard-to-detect faults in large circuits, containing long paths and reconverging fan-outs.

This work is meant to be thought-provoking and should encourage us to drive more experiments in the near future. First, amalgamating more testability measures using machine intelligence whether it is ANN or PCA or any other MI model. We know reconvergent fan-out injects nuances in the circuit and thereby changes the detection characteristics of respective faults and leads to backtracking in ATPG. Also, testability measures have significant inaccuracies due to the assumption that signals at reconvergent fanout stems are treated as independent. MI hopefully solves this problem of detecting faults on reconverging fan-out stem, that further eases the complexity of NP-hard problems like ATPG and test point insertion and also eases the complexity of testability measure calculation by instrumenting a method that can make MI-based testabiity measure, a one pass fault classifier without performing ATPG. Also, a technique may be devised to have a probabilistic MI-based testability measure so that it is portable and accountable for applying direct translation to fault coverage without applying fault simulation and ATPG. Various other applications such as assessment of random pattern testability [28], effective approach to approximate fault simulation [46], structural partitioning into cones [9], critical delay path tracing and characterization of the module in terms of "testability signature" [47], improvement in yield and quality, test point insertion [40], [48] might show bump in their performance metrics using MI-based testability measure.

VII. Conclusion

Human thought process often hits a wall while exploring a hard problem. Such are the NP-hard problems in VLSI testing, like ATPG and test point insertion – using a single testability measure to lower the test generation time or test point insertion time still presents a challenge. Although, no commercial tool exists, the SAT-based techniques [49] and exploration of MI-based application using SAT has been reported [50] as an in-house tool. Research on quantum computing is advancing and quantum-based test generation algorithms may surpass our expectation. Attempts have been made to manifest the expectation [51], [52]. Finally, until a solution is proven to be optimal, there is always hope for further improvements.

Amalgamated testability measure-guided ATPG reduced the total number of backtracks and ATPG CPU time. In practice, easy faults are detected by random vector ATPG and the left over hard-to-detect faults are detected by algorithmic ATPG. Tables III and IV empirically establish that aforesaid detection process of hard-to-detect faults may consume much less time when an amalgamated testability measure is ported inside ATPG. The increasing complexity of VLSI provides strong motivation to amalgamate many more testability measures in the future.

References

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

[2] 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.

[3] P. Goel, “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits,” *IEEE Transactions on Computers*, vol. C-30, pp. 215–222, 1981.

[4] Fujiwara and Shimono, “On the Acceleration of Test Generation Algorithms,” *IEEE Trans. on Computers*, vol. C-32, pp. 1137–1144, 1983.

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

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

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

[8] L. Goldstein, “Controllability/Observability Analysis of Digital Circuits,” *IEEE Trans. on Circuits and Systems*, vol. 26, pp. 685–693, 1979.

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

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

[11] 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.

[12] 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.

[13] S. C. Seth, V. D. Agrawal, and H. Farhat, “A Statistical Theory of Digital Circuit Testability,” *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 582–586, 1990.

[14] R. G. Bennetts, C. M. Maunder, and G. D. Robinson, “COMELOT: a computer-aided measure for logic testability,” *IEE Proceedings E - Computers and Digital Techniques*, vol. 128, no. 5, pp. 177–189, 1981.

[15] R. E. Kalman and R. S. Bucy, “New Results in Linear Filtering and Prediction Theory,” *Journal of Basic Engineering (American Society of Mechanical Engineers)*, pp. 95–108, Mar. 1961.

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

[17] D. M. Singer, “Testability Analysis of MOS VLSI Circuits,” in *Proc. Int. Test Conf.*, 1984, pp. 690–696.

[18] I. Ratiu, “VICTOR: A Fast VLSI Testability Analysis Program,” in *Proc. of the International Test Conf.*, 1982, pp. 397–401.

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

[20] S. Patel and J. Patel, “Effectiveness of Heuristics Measures for Automatic Test Pattern Generation,” in *Proc. 23rd ACM/IEEE Design Automation Conference (DAC)*. IEEE Press, 1986, p. 547–552.

[21] 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.

[22] 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.

[23] 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 Symposium (VTS’21)*, 2021, pp. 1–10.

[24] S. Roy, S. K. Millican, and V. D. Agrawal, “Principal Component Analysis in Machine Intelligence-Based Test Generation,” *Proc. IEEE Microelectronics Design and Test Symposium (MDTS’21)*, 2021.

[25] S. Roy, S. K. Millican, and V. D. Agrawal, “Multi-Heuristic Machine Intelligence Guidance in Automatic Test Pattern Generation,” *Proc. IEEE Microelectronics Design and Test Symposium (MDTS’22)*, 2022.

[26] 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 (ETS)*, 2021.

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

[28] 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.

[29] J. Savir, “Good Controllability and Observability Do Not Guarantee Good Testability,” *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1198–1200, 1983.

[30] V. D. Agrawal and S. C. Seth, *Tutorial: Test Generation for VLSI Chips*. Computer Soc. Press, 1988.

[31] S. Seth, L. Pan, and V. D. Agrawal, “PREDICT - Probabilistic Estimation of Digital Circuits Testability,” in *Proc. of the International Fault-Tolerant Computing Symp.*, 1985, pp. 220–225.

[32] T. Lee, W. Wolf, N. Jha, and J. Acken, “Behavioral Synthesis for Easy Testability in Data Path Allocation,” in *Proceedings 1992 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, 1992, pp. 29–32.

[33] M. Pradhan, B. B. Bhattacharya, and K. Chakrabarty, “Predicting X -Sensitivity of Circuit-Inputs on Test-Coverage: A Machine-Learning Approach,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2343–2356, 2019.

[34] S. Roy, “Toward Zero Backtracks in Test Pattern Search Algorithms with Machine Learning,” Ph.D. dissertation, Auburn Univ., USA, 2021.

[35] 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.

[36] 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.

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

[38] S. Roy, B. Stiene, S. K. Millican, and V. D. Agrawal, “Improved Random Pattern Delay Fault Coverage Using Inversion Test Points,” in *Proc. IEEE 28th North Atlantic Test Workshop (NATW)*, 2019, pp. 206–211.

[39] 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. Electronic Testing: Theory and Applications*, vol. 36, no. 1, p. 123–133, Feb. 2020.

[40] J. Immanuel and S. K. Millican, “Calculating signal controllability using neural networks: Improvements to testability analysis and test point insertion,” in *Proc. IEEE 29th North Atlantic Test Workshop (NATW)*, 2020, pp. 1–6.

[41] 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.

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

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

[44] 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, June 1985.

[45] 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.

[46] R. Lisanke, F. Brglez, A. de Geus, and D. Gregory, “Testability-Driven Random Test-Pattern Generation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 6, pp. 1082–1087, 1987.

[47] M. Abramovici, P. R. Menon, and D. T. Miller, “Critical Path Tracing: An Alternative to Fault Simulation,” *IEEE Design & Test of Computers*, vol. 1, no. 1, pp. 83–93, 1984.

[48] H.-C. Tsai, K.-T. Cheng, C.-J. Lin, and S. Bhawmik, “A Hybrid Algorithm for Test Point Selection For Scan-based BIST,” in *Proceedings of the 34th Design Automation Conference*, 1997, pp. 478–483.

[49] T. Larrabee, “Test Pattern Generation Using Boolean Satisfiability,” *IEEE Trans. on CAD*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[50] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, “A Transitive Closure Algorithm for Test Generation,” *IEEE Trans. CAD*, vol. 12, pp. 1015–1028, Jul. 1993.

[51] M. Venkatasubramanian, “Failure Evasion: Statistically Solving the NP Complete Problem of Testing Difficult-to-Detect Faults,” Ph.D. dissertation, Auburn University, Auburn, Alabama, USA, 2016.

[52] M. Venkatasubramanian and V. D. Agrawal, “Quest for a Quantum Search Algorithm for Testing Stuck-at Faults in Digital Circuits,” in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Amherst, MA, Oct. 2015, pp. 128–133.