Multi-Heuristic Machine Intelligence Guidance in Automatic Test Pattern Generation

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

Spencer K. Millican and Vishwani D. Agrawal
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849-5201
{*millican, agrawvd*}*@auburn.edu*

2022 IEEE 31st Microelectronics Design & Test Symposium (MDTS) | 978-1-6654-6816-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/MDTS54894.2022.9826985

***Abstract*—We present an automatic test pattern generation (ATPG) system using the PODEM (path-oriented decision making) algorithm where backtraces and D-drive are directed by composite heuristics. In a worked out illustration, three heuristic measures are combined by an unsupervised learning procedure of principal component analysis (PCA). The three measures for each signal node are distance (d_{PI} and d_{PO} or minimum distances to primary inputs and outputs), COP (controllability-observability program probabilities, $CC0$, $CC1$, and CO), and SCOAP (Sandia controllabilitiy-observability analysis program combinational testability measures, $SC0$, $SC1$, and SO). PCA combines these into heuristic measures, P_0 and P_1 for directing backtraces, and P_D to advance the D-drive. The new ATPG program for all faults of benchmark circuits shows order of magnitude reduction in total backtracks and notable reduction in CPU time in comparison to the ATPG program with any single heuristic. For individual faults, backtracks often reduced to zero. Even redundant faults were identified by the new program with fewer backtracks.**

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

I. Introduction

Automatic test pattern generation (ATPG) is an essential part of VLSI design and manufacture. An ATPG algorithm finds test patterns for single stuck-at faults in a digital circuit. This problem is proven to be NP-complete [1]–[3], implying that the worst-case complexity of an ATPG program will grow exponentially with the circuit size. Hence, successful completion of an ATPG program for a large circuit and a reasonable run time are conflicting requirements.

ATPG algorithms like D-algorithm [4] and PODEM [5] can guarantee a test, given sufficient run time. As is the widespread practice we will use the PODEM algorithm, which is simpler to implement. However, we rely on the terminology defined by D-algorithm. The symbol D specifies the combined state of a signal being logic 1 in the fault-free circuit and 0 in the faulty circuit. The opposite values are represented by $\overline{D}$. The ATPG starts by placing a D or $\overline{D}$ at the fault site and then setting the primary input (PI) values to propagate these values to a primary output (PO). The process of propagating the objective value through a gate is called D-drive. During the execution of ATPG, the set of gates with D or $\overline{D}$ at input and unknown value (denoted by X) at its output are referred to as a D-frontier.

The PODEM algorithm has two steps, *backtrace* that justifies the logic state of a signal, and D-drive that selects a line from the D-frontier, both potentially benefit from heuristic guidance. Typical heuristics are *distances*, d_{PI} and d_{PO}, the minimum distances to PI and PO [5], COP [6] (controllability-observability program) probabilities $CC0$, $CC1$, and CO, or SCOAP [7] (Sandia controllability-observability analysis program) conbinational testability measures $SC0$, $SC1$, and SO.

An ATPG algorithm requires one more step, *backtrack*. If the D-frontier becomes null before a test is found, successive previous steps are undone until a D-drive become available. The search terminates when either a test is found or all possibilities are exhausted, when the fault is considered redundant or undetectable. The benefit of a heuristic is thus measurable in terms of how few backtracks are required.

An experimental study [8], [9] on the effectiveness of various heuristics in PODEM ATPG found that no single heuristic works best for all faults of a circuit. Rather than using a single heuristic with a high backtrack limit, it is more efficient to use multiple heuristics, each with a low backtrack limit. This is because the program has no way of switching to a second heuristic until the first one has failed.

In a recent study [10], an unsupervised machine learning (ML) procedure of principal component analysis (PCA) combined several heuristics into a single heuristic. The PCA did improve the performance reducing backtracks. However, the procedure was incomplete because the learning procedure was not applied to the forward drive of the D-frontier. The present paper completes that work and shows that the total benefits can be very significant (see Section IV-A and Table III).

ATPG applications of machine learning (ML), both supervised and unsupervised, have recently appeared [11], [12]. Terms machine learning (ML) and machine intelligence (MI) are interchangeably used. Supervised ML uses sample ATPG data and circuit information, that may otherwise direct heuristic decisions, to train an artificial neural network (ANN). The ANN then provides heuristic decisions in the ATPG. Unsupervised ML is more direct as it may not use an ANN is the main topic of this paper. Significant contributions of this work are:

- Development of an ATPG system using the PODEM algorithm guided by three principal components, two for backtraces and one for D-drive, generated through PCA. The ATPG system starts with random vectors to cover easy-to-detect faults and then switches to PODEM.

- Evaluation of ATPG for benchmark circuits based on total number of backtracks and CPU time.
- Evaluation of ATPG based upon backtracks for hard-to-detect, easy-to-detect, and redundant faults.

This article is organized as follows. Section II highlights background work on MI applied to testing and ATPG algorithms. Section III outlines the contribution of this study, explains the principal component analysis (PCA), and techniques to choose major PCs. In Section IV, PODEM guided by PCs is compared with those guided by single heuristics on a sample of faults and on all faults that have not been covered by random patterns. Section V summarizes the study and suggests future work, and Section VI concludes the article.

II. Prior Work

Several recent surveys [11], [13], [14] examine ML-based analog and RF circuit testing, memory testing, applications of ML to hardware security, IC counterfeiting, and devices based on emerging technologies. One of these [13] also discusses the use of ML-based PCA model, in which ATPG algorithmic decisions are guided by PCA models to generate tests for digital circuits.

Popularity of PCA grows as data mining and pruning techniques are addressing storage and computation challenges. Despite the discovery of PCA and its extensions almost a century ago [15], [16], its demand burgeoned when computer-based applications spread across multiple disciplines.

Recent algorithms for test point insertion (TPI) and ATPG used MI as heuristics. A deep learning-based technique could tackle the TPI problem of a logic circuit [17]. An ANN evaluates control-0, control-1, and observes types of test points for impact on fault coverage to iteratively select test points [18], [19]. Subsequent work [20], [21] even trained the ANN with random circuits for performance comparable to that of training from real circuits.

The ANN-based backtracing could reduce backtracks and CPU time compared to other conventional ATPG heuristics [22]. Systematic training of ANN has further advantages [23]. The principal component analysis (PCA) can effectively compact the ANN training data [24]. The possibility of using unsupervised learning for PODEM ATPG, suggested recently [10], has never been harnessed completely until now.

III. Methodology

PODEM [5] is an exhaustive search algorithm. It searches the entire vector space, if necessary, to find a test pattern for a target fault. No test would be found in the end for an untestable (redundant) fault, when the faulty circuit has a logically correct behavior. In coverage such faults may be counted either as nonexistent or virtually detected. It is important for the ATPG algorithm to be complete, otherwise necessary (high) fault coverage may not be attained.

Application of deterministic test vectors to a circuit increases the fault coverage but the rate of increase can be circuit dependent. This coverage is dependent on circuit testability. A statistical analysis of fault coverage for random and deterministic vectors [25] can assess circuit testability from fault simulation, predict coverage from testability analysis, estimate test length for required coverage, or help generate test vectors by fault sampling. We used the results of this analysis to devise a practical ATPG system where easy-to-detect faults are covered by a random pattern generator and hard-to-detect faults are left for a complex program like PODEM where the backtrace guidance comes from either MI [10], [12], [22]–[24], or distance heuristic [5], or Controllability and Observability Program (COP) [6], or Sandia Controllability/Observability Analysis Program (SCOAP) [7]. We found MI-guided ATPG showed significant improvement in performance over others.

A. Random Pattern Test Generation

In 1972, random pattern test generation (RPTG) was used for testing the ILLIAC IV parallel computer [26]. The coverage of random patterns always saturated between 60-80% and switching to D-algorithm [4] program at that point proved beneficial. An essential part of the RPTG scheme is a fault simulator that selects useful patterns. In the present work, we first run an RPTG scheme on a set of benchmarks and try to cover easy-to-detect faults, before switching to PODEM [5] ATPG to detect the remaining hard-to-detect faults.

B. MI-Guided ATPG

Machine intelligence (MI) is used in two ways. Supervised learning trains an ANN, which then guides the ATPG. Unsupervised learning uses statistical tools such as PCA [15], [16] and others to compact the ATPG guidance data. In this study, we apply PCA to a variety of data and then use the most significant PC to direct the ATPG for hard-to-detect faults.

1) Data for Heuristic Guidance: The data used for heuristic guidance in ATPG, as listed in Section I (second paragraph), are *distances*, d_{PI} and d_{PO}, COP [6] probabilities $CC0$, $CC1$, and CO, and SCOAP [7] conbinational testability measures $SC0$, $SC1$, and SO, for all nodes of the circuit.

The 0 and 1 COP probabilities [6] have perfect correlation because $1-CC0 = CC1$. Hence, we discard $CC0$, only using $CC1$. Since each item has a different numerical range, all are normalized to interval [0,1] through division by the largest value occurring in the circuit. The data are then placed in two groups, one to guide backtraces and other to guide D-drives. Data used for backtrace guidance are d_{PI}, $CC1$, $SC0$, and $SC1$. Data used to guide the D-drive are d_{PO}, CO, and SO.

2) Phase Alignment: It is important that items of data do not contradict each other. Suppose, to justify logic 0 at the output of an AND gate whose inputs are all unknown (X) we are selecting an input for backtracing. Using distance heuristic, we will select the input with minimum d_{PI}. According to COP heuristic, we select the input with minimum $CC1$. SCOAP [7] is a measure of the effort of setting a value. Therefore, we would select the input with minimum $SC0$, or if using $SC1$ we select the input with maximum $SC1$. Notice that $SC0$ and $SC1$ of SCOAP are independently computed measures that are not completely correlated.

Table I shows input selection criteria for backtracing specific logic through various gate types and output states, based upon individual heuristics. We observe that the selection criteria does not remain the same across any row in the table, showing conflicts among various heuristics, which must be resolved

TABLE I

BACKTRACING THROUGH A GATE USING HEURISTICS FOR INPUT SELECTION. BOLDFACE ENTRIES INDICATE CONFLICTING SELECTION CRITERION WITH RESPECT TO d_{PI}, USED AS REFERENCE. THE VALUES OF CONFLICTING HEURISTICS ARE COMPLEMENTED BEFORE COMBINING VIA PCA INTO PRINCIPAL COMPONENTS P_0 AND P_1.

Gate type	Output value	Input selection criterion for d_{PI}	$CC1$	$SC0$	$SC1$	PCA 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

D-DRIVE BY HEURISTIC SELECTION OF A D FROM D-FRONTIER. BOLDFACE ENTRY INDICATES CONFLICTING SELECTION CRITERION WITH RESPECT TO d_{PO}, USED AS REFERENCE. THE VALUES OF CONFLICTING HEURISTICS ARE COMPLEMENTED BEFORE COMBINING VIA PCA INTO A PRINCIPAL COMPONENT P_D.

D-drive selection criterion w.r.t. heuristics			
Distance d_{PO}	COP CO	SCOAP SO	PCA P_D
min	**max**	min	min

before they are combined. We use the distance (d_{PI}) as reference and complement the data whose criteria differs (shown in bold in the table). For example, in the first row $SC1$ will be replaced by $1 - SC1$. We recall that all data have been normalized to the range [0,1].

Similar approach is taken to calculate the composite heuristic for D-drive that selects a D from the D-frontier to be propagated toward PO. Once again, various heuristic data, d_{PO}, CO, and SO, all normalized to the range [0,1], are examined in Table II for conflicts. Thus, heuristic data for CO, shown in boldface, are complemented as $1 - CO$, while d_{PO} and SO remain unchanged.

3) Reduction of Data Dimensions: PCA is a powerful tool to reduce data dimension by transforming to new variables called principal components (PCs). Each PC has linear dependency on the original data that maximizes uncorrelated data variance while preserving statistical information. Evaluation of PCs from the original data uses single value decomposition (SVD) [27] that chooses PCs based on either correlation or covariance matrix.

The first (major) PC has the highest variance (see Figures 1 through 3). The explained variance, π_j of the jth PC, is the ratio of its variance λ_j to the total variance (sum of variances of all PCs) [10]:

$$\pi_j = \frac{\lambda_j}{\sum_{i=1}^{p} \lambda_i} \tag{1}$$

where, p is the number of PCs and λ_i is the individual variance of ith PC. A proportion of total explained variance for a subset S of q PCs is expressed as a percentage of the total variance: $\sum_{i \in S} \pi_i$. It is common to set a threshold for this total variance to decide how many PCs to use; only 1 to 3 PCs may be required. However, in some cases, such as outlier detection [27] or image analysis, more PCs may be of interest.

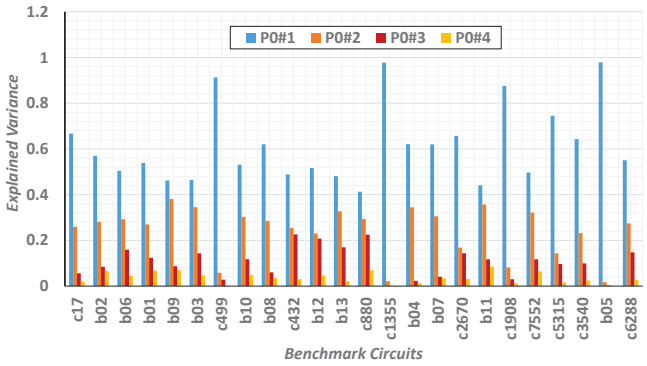

Fig. 1. PCA for backtracong 0 in ISCAS'85 and ITC'99 benchmarks. Logic 0 output state is assumed for all gates during PCA. Items with conflicting criteria, shown in bold in Table I, were complemented. Four PCs are P0#1, P0#2, P0#3 and P0#4. The major PC, P0#1 shown in blue, is selected as P_0 of Table I [10].

We used SVD to obtain PCs based on "explained variance" [28]. PCs represent the same amount of information as the original data. The total variances of the original data and PC data are same but unequally distributed among PCs.

PCA was conducted in two phases. The first phase combines d_{PI}, $CC1$, $SC0$, and $SC1$ as organized in Table I to produce principal components P_0 and P_1 for each signal node in the circuit to provide backtrace guidance. The second phase applies PCA to d_{PO}, CO, and SO to obtain P_D for every signal to guide the D-drive. This requires three applications of PCA:

1) Generation of P_0: Logic 0 output is temporarily assumed for all gates. PCA was applied to the four-dimensional data of all signal nodes. For each node, the row in Table I corresponding to the source gate type with output 0 was considered. Data items with min-max criterion, shown in boldface, were complemented. The PCA produced, as shown in Fig. 1, four PCs P0#1, P0#2, P0#3 and P0#4. The major PC, P0#1 shown in blue, was selected as P_0 of Table I.
2) Generation of P_1: Logic 1 output is temporarily assumed for all gates. PCA was applied to the four-dimensional data of all signal nodes. For each node, the row in Table I corresponding to the source gate type with output q was considered. Data items with min-max criterion, shown in boldface, were complemented. The PCA produced, as shown in Fig. 2, four PCs P1#1, P1#2, P1#3 and P1#4. The major PC, P1#1 shown in blue, was selected as P_1 of Table I.
3) Generation of P_D: Items with conflicting criteria, shown in boldface in Table II, were complemented for all signal nodes. PCA applied to the three-dimensional data for all signal nodes produced, as shown in Fig. 3, three PCs PC#1, PC#2 and PC#3. The major PC, PC#1 shown in blue, was selected as P_D in Table II.

IV. EXPERIMENTAL RESULTS

We expect electronic design automation (EDA) system vendors to incorporate MI in their ATPG software. Since they are reluctant to divulge program source code, it was not

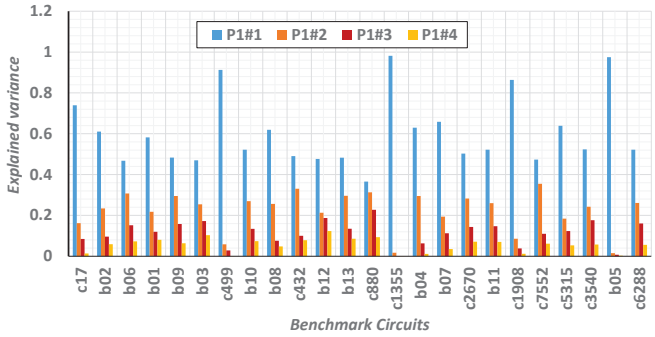

Fig. 2. PCA for backtracing 1 in ISCAS'85 and ITC'99 benchmarks. Logic 1 output state is assumed for all gates during PCA. Items with conflicting criteria, shown in bold in Table I, were complemented. Four PCs are P1#1, P1#2, P1#3 and P1#4. The major PC, P1#1 shown in blue, is selected as P_1 of Table I [10].

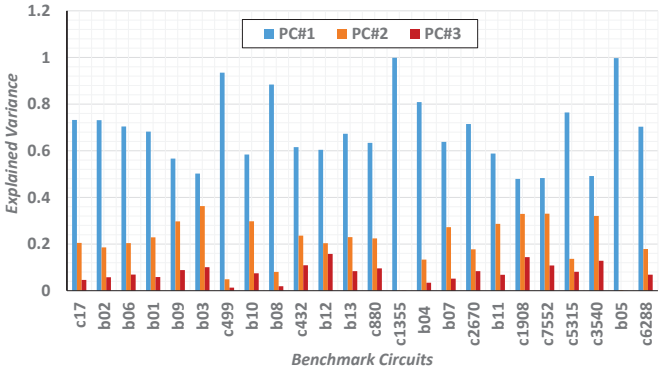

Fig. 3. PCA for directing D-drive in ISCAS'85 and ITC'99 benchmarks. Items with conflicting criteria, shown in bold in Table II, were complemented. Three PCs are PC1#1, PC1#2 and PC3#4. The major PC, PC#1 shown in blue, is selected as P_D of Table II.

possible to incorporate our procedures in a commercial tool to assess the advantage. Therefore, we used an EDA system developed in-house to observe the comparative improvement. Given that our EDA system cannot match the mature performance of commercial tools, the results in this section may only be interpreted as the potential relative benefits of the new procedure over conventional methods.

We used a workstation comprised of Intel i7-8700 processor and 8-GB of RAM. Our EDA tools were implemented in C++ using the MSVC++ 14.15 with maximum performance optimization, and all PCA activities were executed in Python. PODEM ATPG [5] and an event-driven fault simulator [29] were developed such that any topological data or testability measure, e.g., distance [5], COP [6], SCOAP [7], or PCA can be applied across ISCAS'85 [30] and ITC'99 [31] benchmarks without restricting to any single heuristic.

The ATPG system starts with all checkpoint single stuck-at faults. It has an initial random test pattern generation (RTPG) phase using a pattern generator and fault simulator. Only the vectors that detect new faults are retained. The faults left over by RTPG are given to the PODEM ATPG, which can use any one of the four heuristics distance, COP, SCOAP, or PCA. This phase generates tests for one fault at a time. Either a test vector is generated or the fault is found to be redundant. Generation of a test is followed by fault simulation to update the fault list. ATPG ends when the fault list becomes empty.

TABLE III
C6288: COMPARING ATPG SYSTEM IMPLEMENTATIONS.

Characteristic	Performance with respect to various heuristics				
	Dist.	COP	SCOAP	PCA composite	
				Prev. [10]	Present
CPU time (s)	125.9	101.9	191.7	47.0	1.9
Backtracks	43,675	65,996	24,268	5,303	128

A. A Comparison with Previous ATPG System

In our previous work [10] we applied unsupervised learning or PCA only in the backtrace step. The D-drive selection used the conventional distance heuristic [5]. The present work completes that investigation by applying PCA to both backtrace and D-drive. In addition, we have a complete ATPG system with random and algorithmic phases and a fault simulator. For evaluating the effects of the added D-drive, we compared the supervised learning versions of the previous algorithm [10] and the present one. Table III gives the results for c6288. Our version of the netlist had 7,744 checkpoint faults; no further fault collapsing was done. RTPG produced 50 vectors leading to 99.14% fault coverage. In both cases, random pattern phase and fault simulation were common. The only difference was the absence of PCA guided D-drive in the previous system.

We also observe, as pointed out by other researchers [8], [9], that no single heuristic is ideal for all cases. For example, COP gives lowest CPU time (101.9 s) but SCOAP produces fewest backtracks. In the case of PCA, the previous implementation [10] using composite heuristic on backtraces and distance for D-drive did better than individual heuristics, taking 47 s with 5,303 backtracks. The best result is from the present PCA composite heuristics applied to both backtrace and D-drive, requiring only 1.9 s and 128 backtracks.

B. Sample of Faults

In another experiment we used sample of faults (easy-to-detect, hard-to-detect, and redundant) for circuits to prove the power and efficiency of PCA-guidance via a fault-by-fault approach. Tables IV and V show a reduced number of backtracks (almost zero in several cases) for 12 faults (7 hard-to-detect, 3 easy-to-detect, and 2 redundant faults). Although, the number of backtracks for redundant faults were reduced, they did not drop to zero, confirming that it must take at least one backtrack to complete the search for a test for a redundant fault. These results are shown for circuits c6288 and b07 in Tables IV and V, respectively.

C. Faults Filtered through Random Pattern Testing

Experiments used faults filtered out using random testing to prove the efficacy of guidance provided by PCA to PODEM ATPG. In Figures 4 and 5 circuits are arranged left to right in order of increasing number of nodes. Fig. 4 gives total CPU time and Fig. 5 shows combined backtracks for all stuck-at faults left over from RTPG. Corresponding to the four versions of PODEM, there are four bars for total backtracks in the bar chart and four points for CPU times, for each circuit. Trend curves in Fig. 4 graph are the power-law fit for the four

TABLE IV
BACKTRACKS IN PODEM ATPG WITH HEURISTIC GUIDANCE FROM DISTANCE, COP, SCOAP, AND PCA FOR 12 SAMPLE FAULTS (EASY-TO-DETECT, HARD-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 V
BACKTRACKS IN PODEM ATPG WITH HEURISTIC GUIDANCE FROM DISTANCE, COP, SCOAP, AND PCA FOR 12 SAMPLE FAULTS (EASY-TO-DETECT, HARD-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**

PODEM versions. Notably, the black bars and black curve indicate consistent improvement by the PCA guidance.

Results show that multiple heuristics combined as a linear combination effectively reduce backtracks and CPU time of ATPG over any single heuristic. Circuits b03, c432, b10, b13, c880, b07, b05, b12, c5315, c7552, c1355, c2670, c3540, b04, b11, b08, c499 and c6288 had significant reduction in backtracks and CPU times (Figures 4 and 5). PCA is frequently the best guidance for ATPG, but even when it is not, it is never the worst. There are no reconvergent fanouts in c17, b02, b01, and b06, and hence there is no backtrack possibility. An example of zero backtracks by PCA-based PODEM ATPG is b09 (Fig. 5).

V. DISCUSSION AND FUTURE WORK

The ATPG CPU time of PCA-guidance was reduced only slightly for circuits with fewer nodes compared to that of conventional heuristic guidance (see Fig. 4). However, the ATPG CPU time of PCA-guidance was reduced uniformly across all the benchmarks. Therefore, it can be concluded that the PCA-guidance may be the best novel composite heuristic compared to other conventional heuristic guidance for ATPG.

Many years of research and development has significantly advanced the state-of-the-art of PODEM ATPG programs. However, MI-enabled PODEM was never explored and this is the first time in the VLSI testing history it is shown to

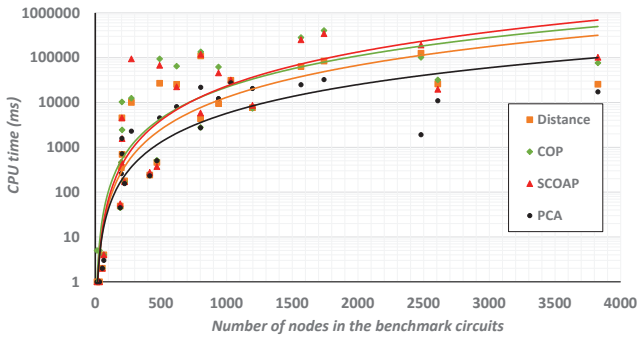

Fig. 4. CPU time to find testable and redundant checkpoint faults filtered out of random pattern testing.

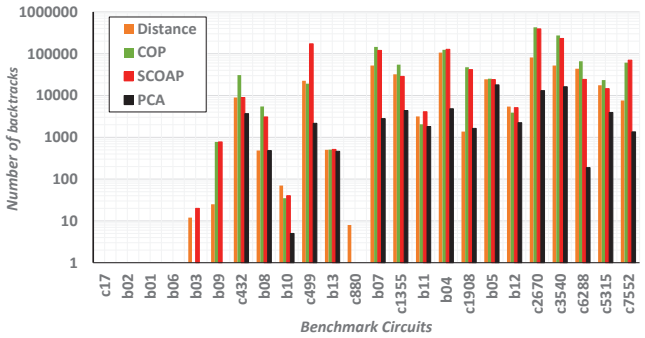

Fig. 5. Backtracks required to find testable and redundant checkpoint faults filtered out of random pattern testing.

further improve the detection of both easy as well as hard-to-detect faults in typical large circuits with long paths and many reconvergent fan-outs.

SAT-based techniques are another possible area where MI application may be explored. Although no commercial tools exist, the technique is reported in research [32] and as an in-house tool [33].

Within unsupervised learning, the PCA-based ATPG as described here, opens up new avenues for future work. First, reconvergent fanout-free circuits do not have any backtracks, but backtraces can be reduced to generate more efficient tests requiring fewer PI assignments. Second, a total backtrack elimination has a cost in CPU time, finding a "sweet-spot" may be feasible to obtain optimized non-zero backtracks for minimal CPU time. Third, a quick detection of redundant faults in a circuit may expedite ATPG by MI-guided ATPG. Fourth, comparing k-means clustering against PCA as the second unsupervised learning technique could reveal some interesting results. Fifth, state-of-the-art ATPG tools are unable to detect some faults either due to the circuit size or some characteristics of faults. Sixth, this study used academic benchmark circuits instead of large industry-standard circuits; the authors believe the observed performance trends are likely to be valid for larger circuits and prove to be valuable in the future.

Supervised learning using PCA may have specific benefits worth investigating. Tables IV and V show reduced backtracks by PCA-based unsupervised learning in redundancy identification, which is regarded as a difficult ATPG problem. We should expect greater advantage with supervised learning

where training data could be derived from ATPG runs on redundant faults of the training circuit.

VI. Conclusion

Researchers are limited in their imagination to find solutions for NP-complete problems. All ATPG algorithms use various heuristics to achieve a lower test generation runtime, but achieving the lowest ATPG run-time is still an open problem. This study uses MI to combine various ATPG inputs as heuristics, narrow down the search space, and achieve speed-up in ATPG runtime [10], [22]–[24]. However, with the dramatic rise in research on quantum computing, it is only natural to use those ideas to break the VLSI testing area out of its plateau. The discovery of quantum-based test generation algorithms may break the computational barrier and achieve the theoretical run-time complexity of $\sqrt{N}$ [34]. Until it is proven that a given solution is the most optimal, there is always a pursuit to research. Attempts have been reported but the field is still open [35], [36].

Unsupervised learning of MI when integrated with PODEM ATPG decreased backtrack and CPU time. Practical ATPG systems combine a simple program (e.g., random vector ATPG) for easy faults and a complex program (e.g., MI-based or quantum-computing ATPG) for hard-to-detect faults. PCA was used to combine multiple features, and a linear transformation formulated the ATPG backtracing using a new major PC (the first PC) that replaces the tradition single-heuristic guidance found in practical ATPG (tested on hard-to-detect faults).

Acknowledgment: We express our sincere gratitude to Dr. SueAnne Griffith for valuable comments and suggestions.

References

[1] H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems for Combinational Logic Circuits," *IEEE Transactions on Computers*, vol. 31, pp. 555–560, 1982.

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

[3] G. Seroussi and N. H. Bshouty, "Vector Sets for Exhaustive Testing of Logic Circuits," *IEEE Trans. Information Theory*, vol. 34, no. 3, pp. 513–522, 1988.

[4] 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.

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

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

[13] H. Stratigopoulos, "Machine Learning Applications in IC Testing," in *Proc. IEEE 23rd European Test Symposium (ETS)*, 2018, pp. 1–10.

[14] M. Pradhan and B. B. Bhattacharya, "A Survey of Digital Circuit Testing in the Light of Machine Learning," *WIREs Data Mining Knowl. Discov.*, pp. 1–18, 2020.

[15] 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.

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

[17] 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 *Proc. 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[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. 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.

[26] V. D. Agrawal and P. Agrawal, "An Automatic Test Generation System for Illiac IV Logic Boards," *IEEE Transactions on Computers*, vol. C-21, pp. 1015–1017, 1972.

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

[28] J. Guo, G. James, E. Levina, G. Michailidis, and J. Zhu, "Principal Component Analysis With Sparse Fused Loadings," *Journal of Computational and Graphical Statistics*, vol. 19, no. 4, pp. 930–946, 2010.

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

[30] 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.

[31] 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.

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

[33] 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.

[34] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219.

[35] 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.

[36] 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.