Principal Component Analysis in Machine Intelligence-Based Test Generation

Soham Roy, Spencer K. Millican, and Vishwani D. Agrawal
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849-5201
{*szr0075, millican, agrawvd*}@*auburn.edu*

Abstract—**In a machine intelligence (MI)-based automatic test pattern generator (ATPG), an artificial neural network (ANN) may guide decisions that would otherwise rely on some heuristic. Heuristics use circuit-specific data such as gate types, logic depth, fan-out data, or various testability measures. Treating these data collectively as a multivariate statistic of circuit topology, this study extracts principal components (PCs). A subset of PCs is then used to train the ANN that facilitates algorithmic decisions in ATPG. This reduces the ANN complexity and enhances ATPG efficiency. Results on benchmark circuits show the benefit of reduced CPU time.**

Index Terms—**ATPG, Backtrace, Digital testing, Heuristics, Machine intelligence (MI), Principal component analysis (PCA)**

I. Introduction

With technology scaling, ICs continue to become more complex, making automatic test pattern generation (ATPG) exceedingly inefficient. This is because test generation belongs to a set of NP-complete problems [1]. In the last century, many algorithms were developed to manage the computation time of test generation and improve the commercial electronic design automation (EDA) tools. When ATPG algorithms faced CPU time problems, advances in computing technology were able to improve test generation efficiency. A serious problem is the ability to examine nearly all possible circuit inputs to find a test with reasonable effort. ATPG algorithms use heuristics.

An ATPG algorithm traces backward, i.e., "backtraces" [2], with an objective to assign certain logic value to a signal. A signal is either a primary input (PI) or output of a gate. When a signal fans out, multiple copies of the signal are often identified as separate signals. Thus, backtracing produces an ordered list of signals starting from an internal signal to a PI and a specified value to be assigned. This PI assignment may or may not lead to a test. In the latter case, one would *backtrack* to undo the PI assignment, and then continue with more backtraces. To reduce the possibility of backtracking, *heuristics* based on designer's intuition help select a backtracing direction from available choices [3]; tracing backward through a multiple input gate is a typical situation.

Supervised learning in machine intelligence (MI) includes the popular artificial neural networks (ANNs). The main advantage of ANNs is they assimilate input-output relationships (i.e., pattern recognition) of a target problem. They are popular because (1) they eliminate the need for mathematical formulation of complex processes and (2) they interpolate quantities of interest without any linear assumptions.

This study improves ANN training quality and efficiency by pre-processing training data with principal component analysis (PCA) [4], [5]. PCA extracts relevant features from a list of many features to train an ANN [6]–[8]. It offers a viable pre-processing step [9] to decrease ANN complexity.

Human clairvoyance in the form of heuristics was successfully used in ATPG programs, but it has been reported [10], [11] that no single heuristic works optimally for all instances, and the use of multiple heuristics can be computationally expensive. Although MI is not a new technique, considerable performance impact of ANN was found in test point insertion (TPI) [12]–[16]. Recent advances in MI-based ATPG demonstrated that heuristics can be easily incorporated in ATPG through MI [17], [18]. Thus, MI could harness the benefits of multiple heuristics. However, as the volume of heuristic data increases, the workhorse of MI, i.e., the ANN, tends to be overloaded to the extent that its efficiency suffers. Principal component (PC) analysis (PCA) [4], [5] can amalgamate training features to enhance supervised learning of ANN. Although the application of PCA-assisted ANN is not new, this study demonstrates its novel application to ATPG.

The heuristics in ATPG are built around topological data of the circuit, and this study uses correlation among data to achieve compaction [4], [5] and extracts the PC of the circuit data, thus the ANN complexity is reduced as it is now trained only with a few selected PCs. Additionally, the ATPG CPU time is reduced since the trained ANN is now less complex, and evaluating weights and biases of ANN edges requires smaller matrix multiplication and fewer computations of non-linear Sigmoid functions [19].

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 that explains the PC extraction with detailed mathematical backgrounds and technique to choose major PCs. Section IV evaluates the performance of PODEM guided by the PC-trained ANN against that of PODEM guided by ANN without PCs [17], [18]. Section V summarizes the study and suggests future work, and Section VI concludes the article.

II. Prior Work

Various VLSI studies incorporate ANNs to model circuits and in algorithms [20]–[22]. ANN has been used to model a digital circuit where a bi-directional binary neuron represents the state of a signal [23]. Since the network energy function depends on all signals and has many local minima, finding a test becomes very difficult. In contrast, other work [17] used a conventional digital circuit model and a search algorithm that guarantees a test given unlimited computing resources. However, an ANN can guide the search for test while avoid-

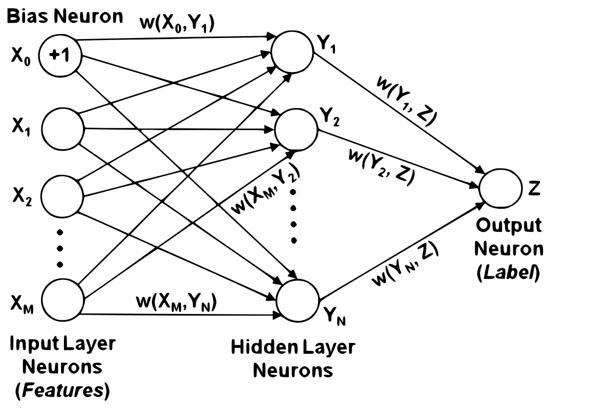

Fig. 1. An ANN used to guide ATPG [17] comprises of inputs (bias X_0 fixed at 1.0, and M inputs, X_i), a hidden layer of N neurons, $\{Y_j\}$, and a single output neuron, Z. Directed edges connect input to the output neuron, via hidden layer neurons.

ing unproductive decisions. Hence, conventional backtracing heuristics were replaced by an ANN. Sample conventional ATPG steps were used to train the ANN, which then guided the ATPG. The training data also contained topological and functional features used for conventional ATPG heuristics. Although the authors restricted themselves to the Path-Oriented Decision Making (PODEM) [3] ATPG algorithm to demonstrate the efficacy of the ANN-based heuristic, the technique can be applied to most ATPG algorithms. The ANN-guided ATPG consistently outperformed the conventional heuristics, but no noticeable improvement in backtracking performance for some circuits was reported [17]. Since, the initial ANN training was ad hoc and elementary, a structured training methodology could elevate the performance of the ANN-guided PODEM [18]. This included recursive training of the ANN from both hard and easy-to-detect faults, conflict resolution among training data patterns, e.g., the same ANN input requiring different outputs, and discarding training data that did not positively impact the guidance.

MI can combine multiple heuristics and improve the efficiency of ATPG [17], [18]. The ANN comprises an input layer, a single hidden neuron layer, and an output neuron, as illustrated in Fig. 1 [17]. The output of a neuron lies between 0 and 1. $X_0, X_1, \cdots, X_M$ are inputs referred to as "features" of the ANN. Their values are normalized in the range [0,1]. X_0 is a bias input and its value is fixed at 1.0. $Y_1, Y_2, \cdots, Y_N$ are known as hidden neurons (HNs), and the output neuron Z is called the "label". Each neuron's output value is denoted as x_i, y_i or z. $w(A, B)$ denotes the directed edge from any neuron to another neuron and carries a signed floating-point value. The output of any neuron Y_j is,

$$y_j = f(\sum_{i=1}^{M} x_i \times w(X_i, Y_j)) \tag{1}$$

where f is called the activation function [24] for which the Sigmoid function [19], expressed below, is used.

$$f(v) = 1 - \frac{1}{1 + e^{-v}} \tag{2}$$

The inputs (features) and expected output (label) values are the main components of training data. During training, the output label is computed based on any given input features and compared to the expected output neuron value. The mean square error (MSE) is computed by calculating the square of the difference between the computed and expected value of the output neuron, which is averaged over all training datasets. Weights and biases are adjusted during successive training "epochs" to minimize MSE, and tuning other hyper-parameters such as the number of hidden layers, number of hidden neurons (HNs)/hidden layers, learning rate, activation functions, etc., minimizes the MSE. The input features of the ANN comprise (1) numerically encoded gate-type of a gate which is driven by the circuit line being traced, (2) COP [25] CC (probability of setting the line to "1") and CO (observablity of the traced line), and (3) the minimum distance between the traced line and any PIs of the circuit.

The performance of ANN-guided PODEM [17] was further improved by optimizing the training methodology [18]. However, the addition of more features to the ANN would cause problems of high volume of the training dataset and the ANN complexity to absorb and retain the information. This leads to the storage crunch of such a high volume of training data and leads to high ANN training time. Also, it may be possible that some training features are irrelevant, and therefore extraction of useful training features is one such novel contribution of this article, which enhances ATPG performance (both in terms of backtracks and CPU time).

An increase in the training data set is prevalent and alarming. A multivariate statistical method, popularly known as PCA [4], [5], reduces the data sets' dimensionality and increases interpretability with minimum information loss. PCA creates new uncorrelated variables (also known as PCs) with maximum variances. Finding PCs reduces solving an eigenvalue/eigenvector problem; the new variables are not defined a priori, but by the data set at hand, making PCA a pliant data analysis technique.

PCA is a technique to identify patterns in data and express the data to show the similarities and dissimilarities. Any patterns in high-dimensional data are hard to find, but PCA plays a vital role in analyzing these data, where the luxury of graphical representation is not available. The PCA is also useful in compressing data by reducing the number of dimensions without losing necessary information once the patterns are found [4], [5]. Before this work, PCA finds significant application in image processing and recently in the form of unsupervised learning in ATPG [26], but this statistical tool has not been explored as a pre-processing step of ANN training in ATPG or MI-based ATPG (also known as *PCA-assisted ANN-guided ATPG*).

III. Methodology

This study demonstrates PCA-based pre-processing of training data obtained from circuits, c6288, c3540, and b05, chosen due to their large logic depths. The use of deep circuits in training ANN for ATPG is empirically found to be effective [18]. The ANN training data is obtained from successful and failed backtraces in a COP-based ATPG, as illustrated in the recent work [17]. The backtrace histories are stored. Backtraces that lead to backtrack are classified as "failure", and those that lead to fault detection as "success". In the present research,

TABLE I
EXAMPLE OF 8-DIMENSIONAL FEATURE (SIGNAL CHARACTERISTIC) DATA FOR FIRST 5 SIGNALS OF TRAINING CIRCUIT C6288.

Fan-out	*Gate type*	*COP CC*	*COP CO*	*SCOAP SC0*	*SCOAP SC1*	*SCOAP SCO*	*Dist.*
0.000	0.000	0.063	1.000	0.013	0.058	0.000	0.237
0.000	0.000	0.063	1.000	0.013	0.058	0.004	0.211
0.000	0.000	0.938	1.000	0.034	0.016	0.004	0.184
0.000	0.000	0.938	0.063	0.034	0.016	0.045	0.158
1.000	0.330	0.500	0.125	0.007	0.011	0.034	0.132

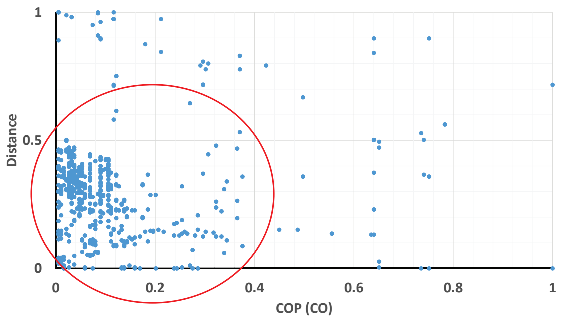

Fig. 2. A two-dimensional scatter plot of "distance" and "COP CO" data for all signals in training circuits c6288, c3540 and b05. A nearly circular concentration indicates a weak correlation between two features.

the ANN recognizes eight features (characteristics) for each signal line (PI, gate output, or fanout branch). The feature values are normalized in the range [0, 1]. They are specified below, with examples shown in Table I:

1) **Fanout** - Its value is 0 for a signal (line) with single destination, and 1 for multiple destinations.
2) **Gate type** - The type of a signal is specified numerically. PI, fanout branch, and inverter output are type 0.0. A multiple-input gate output signal, which can be a non-fanout signal or a fanout stem, is type 1 through 6 corresponding to AND, NAND, OR, NOR, XOR, or XNOR gate, respectively. After normalization the gate-type becomes 0.0, 0.167, 0.33, 0.5, 0.67, 0.83, or 1.0.
3) **COP CC** - Combinational controllability computed by COP [25] as probability of setting the signal to 1.
4) **COP CO** - Combinational observability computed by COP [25] as probability of observing the signal at primary outputs (POs).
5) **SCOAP SC0** - Effort of setting the signal to 0 as computed by SCOAP [27].
6) **SCOAP SC1** - Effort of setting the signal to 1 as computed by SCOAP [27].
7) **SCOAP SCO** - Effort of observing the signal at POs as computed by SCOAP [27].
8) **Distance** - Number of lines on the shortest path between the signal and PIs, normalized with respect to the maximum PI to PO depth of the circuit.

A. Principal Component Analysis

This section performs PCA on a set of data and also attempts to provide elementary mathematical background required to understand PCA's mechanisms, such as calculating mean, covariance matrix, eigenvectors, and eigenvalues of a covariance matrix, choosing the components that form a feature vector, and finally deriving a new data set based on feature vectors.

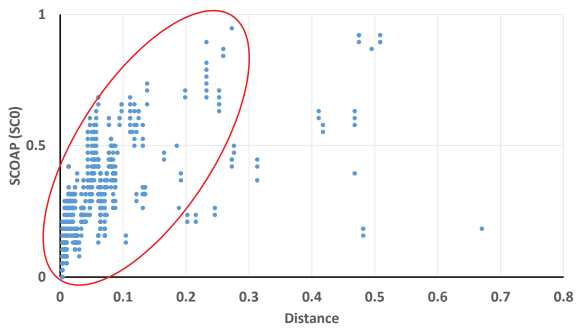

Fig. 3. A two-dimensional scatter plot of "SCOAP SC0" and "distance" data for all signals in training circuits c6288, c3540 and b05. The elliptical concentration indicates significant correlation between two features.

TABLE II
EXAMPLE OF FEATURES x = COP CO AND y = DISTANCE IN TABLE I, AND MEAN-ADJUSTED VALUES FOR FIRST 5 SIGNALS OF C6288. MEANS $< x >$ AND $< y >$ ARE COMPUTED FOR ALL SIGNALS OF TRAINING CIRCUITS C6288, C3540 AND B05.

COP (CO) *x*	*Distance* *y*	$x_{adjust} = x - < x >$	$y_{adjust} = y - < y >$
1	0.237	0.795	-0.021
1	0.211	0.795	-0.048
1	0.184	0.795	-0.074
0.063	0.158	-0.143	0.100
0.125	0.132	-0.080	-0.127

*1) **Step 1: Getting data**:* Examples of 8 input features of the ANN are given in Table I. For simplicity, 2-dimensional data for "COP CO" and "Dist." are illustrated in Table II. Figures 2 and 3 show the combined training data from circuits c6288, c3540, and b05 for two pairs of features. These are scatter plots of raw data and show how some features can have stronger correlation. The circular or elliptical concentration with minimum outliers indicates uncorrelated or correlated data. Although not empirically proven here, this issue will be revisited in later sections.

*2) **Step 2: Subtracting mean**:* The mean of each data types in Table I is calculated and subtracted from the respective data as shown in Table II. The well known procedure for computation of mean is [28]:

$$< a >= \frac{1}{N} \sum_{i=1}^{N} a_i = \frac{a_1 + a_2 + \cdots + a_N}{N} \tag{3}$$

where $a = x$ or y, and N is the total number of signals in training circuits.

*3) **Step 3: Calculating covariance matrix**:* Data sets can be either single or multi-dimensional, and depending on which various statistical tools are involved in calculating the effect of respective dimensions on each other. Standard deviation and variance are such statistical tools that can be used for one-dimensional data sets to calculate the standard deviation for each dimension of the data independent of the other dimensions. However, it is essential to have an evaluation metric to find how much the dimensions vary from the mean concerning each other, known as *covariance*. The mathematical formula for covariance is similar to variance [28]:

$$Var_x = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - <x>)^2 \tag{4}$$

$$Cov_{x,y} = \frac{\sum_{i=1}^{N}(x_i - <x>)(y_i - <y>)}{N-1} \tag{5}$$

The total number of covariances for 2 or more data dimensions is $\frac{n^2-n}{2}$, where n is the dimension of data set. As this study deals with 8-dimensional data sets, $n = 8$, and the total number of covariances (i.e., the number of elements in either the lower triangle or the upper triangle of a symmetric matrix) is 28.

For a simple illustration, let us consider 2-dimensional data of Figure 2. There is only one covariance. Higher the covariance, stronger is correlation between features. In this case, covariance is extremely low, suggesting that these features are almost uncorrelated. The covariance matrix is computed using equation 5 as:

$$C = \begin{pmatrix} 0.057 & 0.008 \\ 0.008 & 0.026 \end{pmatrix} \tag{6}$$

*4) **Step 4: Calculating eigenvectors and eigenvalues:*** Eigenvectors and eigenvalues [28] are calculated from the covariance matrix C using the python standard library. These signify the relative strengths of data components, which help fetch significant PCs. Eigenvectors are orthogonal to each other and provide useful reorganization of data. Eigenvectors represent a rotation matrix, the eigenvalues correspond to the square of the scaling factor in each dimension. Here,

$$eigenvectors = \begin{pmatrix} 0.973 & -0.232 \\ 0.232 & 0.973 \end{pmatrix} \tag{7}$$

$$eigenvalues = \begin{pmatrix} 0.059 & 0.024 \end{pmatrix} \tag{8}$$

*5) **Step 5: Forming a feature vector:*** This step illustrates compression and reduced dimensionality of data set. All eigenvectors are different and have different eigenvalues. The eigenvector with highest eigenvalue is the major principal component of the data set. It carries maximum significance among data dimensions. Eigenvectors are obtained from the covariance matrix and ordered according to decreasing eigenvalues. One may choose the significant eigenvectors based on their high eigenvalues and discard the rest of the eigenvectors without losing much information, as shown below. In the present situation, the two variables have rather low correlation and hence none would be dropped. However, just for illustration if we were to drop the second variable,

$$selected\ eigenvector = \begin{pmatrix} 0.973 \\ 0.232 \end{pmatrix} \tag{9}$$

Finally, n dimensional data may produce at most n eigenvectors and corresponding eigenvalues. A subset of p eigenvectors may be chosen eliminating those with relatively small eigenvalues. Finally, a data set of dimension p ($p \leq n$) is created in the next step.

*6) **Step 6: Reconstructing a new data set:*** A transform T is an $n \times n$ square matrix containing eigenvectors as rows. The mean-adjusted feature data for each line is an n-dimensional vector. This vector when multiplied by T, produces a new n-dimensional vector of principal components (PC) for the corresponding line. A subset of p data elements is selected as described in the previous step. Similar transformation is applied to all lines of the circuits.

B. Selecting Major Principal Components

This section highlights details on generation of six major PCs for each line, as discussed in Section III-A. There are various avenues to fix the number of significant PCs, but the Pearson correlation coefficient (PCC)-based technique is chosen to compress/extract the final dimensions of PC-based data [29]. PCA is well-suited to compressing data that are highly correlated with each other, and therefore, dimensionality reduction is a key benefit [4], [5]. However, datasets contain a mix of correlated and uncorrelated items. The neural network training becomes more efficient when all its input features are strictly orthogonal or, in other words, un-correlated. Therefore, PCC is a handy technique by which one can compress unnecessary correlated data, keeping the uncorrelated data intact. The well-known equation for PCC [28] is as follows:

$$r = \frac{\sum_{i=1}^{N}(x_i - <x>)(y_i - <y>)}{\sqrt{\sum_{i=1}^{N}(x_i - <x>)^2}\sqrt{\sum_{i=1}^{N}(y_i - <y>)^2}} \tag{10}$$

where r is PCC, x and y are data samples of two-dimensional dataset, $<x>$ and $<y>$ are computed mean of data samples, and N is number of samples.

The entire data set from three training circuits, of which only a sample is shown in Table I, was analyzed to compute correlation coefficients as shown by the 8×8 matrix below. We observe that diagonal elements are self correlated. Because they are highly correlated, the PCC is 1 (highlighted in bold). Pair-wise correlation coefficients are off-diagonal elements and, as pointed out earlier, considering the diagonal symmetry there are 28 of them.

$$\begin{pmatrix}
\mathbf{1.000} & 0.246 & -0.099 & -0.086 & -0.145 & -0.005 & 0.031 & -0.029 \\
0.246 & \mathbf{1.000} & -0.107 & -0.100 & -0.142 & 0.025 & -0.049 & -0.091 \\
-0.099 & -0.107 & \mathbf{1.000} & 0.064 & 0.322 & -0.164 & 0.039 & 0.212 \\
-0.086 & -0.100 & 0.064 & \mathbf{1.000} & 0.130 & 0.214 & -0.364 & 0.201 \\
-0.145 & -0.142 & 0.322 & 0.130 & \mathbf{1.000} & \mathbf{0.408} & 0.286 & \mathbf{0.622} \\
-0.005 & 0.025 & -0.164 & 0.214 & \mathbf{0.408} & \mathbf{1.000} & 0.175 & \mathbf{0.559} \\
0.031 & -0.049 & 0.039 & -0.364 & 0.286 & 0.175 & \mathbf{1.000} & 0.258 \\
-0.029 & -0.091 & 0.212 & 0.201 & \mathbf{0.622} & \mathbf{0.559} & 0.258 & \mathbf{1.000}
\end{pmatrix}$$

Items 5, 6, and 8, i.e., SCOAP SC0, SCOAP SC1 and distance as shown in Table I example, display strong correlation as highlighted in bold-green color in the PCC matrix. The correlation of SCOAP SC0 and distance is also observed in Figure 3. Six major PCs in this study are based on PCC, as five are weakly correlated and are assumed uncorrelated, and of the remaining three two can be dropped. Thus, only six PCs are used to train the ANN, and to facilitate algorithmic decisions in ATPG.

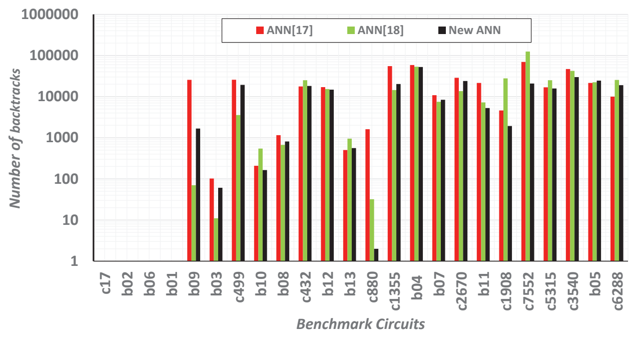

Fig. 4. Backtracks required to find a test or verify redundancy for all checkpoint stuck-at faults in benchmark circuits, arranged left to right in order of increasing logic depth. This study's ANN (black bars) shows reduction in backtracks compared to the previous ANNs shown as orange bars [17] or green bars [18].

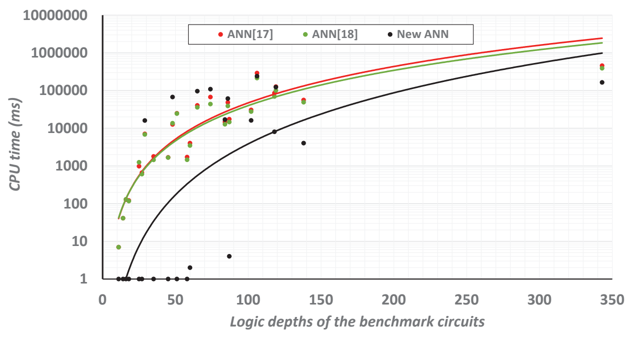

Fig. 5. CPU time to find a test or verify redundancy for all checkpoint stuck-at faults for circuits of Figure 4. The present ATPG (black) used reduced CPU time compared to those previously reported, shown as orange [17] or green [18]. Points indicate actual CPU time for a circuit. The curves are MATLAB fits giving CPU time as a function of logic depth.

C. Preprocessing and ATPG

We record 8 features listed at the beginning of this section for all signal lines in the three training circuits, c6288, c3540, and b05 in a $L \times 8$ matrix, where L is total number of lines in the three circuits. Next, follows the PCA leading to the 8×8 matrix appearing at the end of the last page. A COP-based ATPG is run on the training circuits to record training data along with major PCs for all backtraced lines. Each backtrace is also labeled either as success if it leads to a test, or failure if it is undone by a backtrack. An ANN is then trained, and replaces the conventional heuristics-based sub-routine of a PODEM ATPG. When backtracing through a gate with multiple inputs, the ANN rates the chance for success for each input, and the input with highest rating is chosen. To prepare a circuit under test (CUT) for ATPG, first, all six features are computed for each line, and the values are transformed into PCs, in a similar way as was done for training circuits.

IV. Experimental Results

Experiments were carried out on a workstation containing Intel-8700 processor and 8 GB RAM. The design for test (DFT) electronic design automation (EDA) tools were implemented in C++ using MSVC++14.15 compiler with performance optimizer. The ANN training and PCA analysis were executed using Python and MATLAB. PODEM ATPG [3] was implemented along with an event-driven fault simulator [2]. This PODEM is implemented in such a way that any heuristic, distance [3], COP [25], SCOAP [27], ANN [17], ANN [18] or new ANN, could be applied across ISCAS'85 [30] and ITC'99 [31] benchmark circuits. As an ATPG program is time-expensive to run exhaustively, some faults may be aborted. Nearly identical fault coverage was obtained with each heuristic by using a suitable per-fault time limit.

We hope this study will inspire EDA vendors to put in MI in their ATPG software. Understandably, EDA vendors were disinclined to pass on their program source code, and it was infeasible to perform research-oriented experiments using binary. Therefore, our experiments were restricted to in-house EDA tools.

The ATPG was applied to all testable and redundant single stuck-at faults in each circuit. Figures 4 and 5 show the number of backtracks and ATPG CPU time (ms) of three ANN-based heuristics: ANN [17] shown in orange, ANN [18] in green, and new ANN (PCA-assisted) in black. For each circuit three bars record total backtracks in Figure 4 and three points show CPU times in Figure 5, corresponding to the three versions of PODEM. Curves in Figure 5 are MATLAB power-law curve fits ($y = cx^b$) for the three PODEM versions. For circuits b10, b12, c880, b04, b11, c1908, c7552, c5315, and c3540 the new ANN outperforms the other two ANN-based heuristics [17], [18] in terms of backtracks and ATPG CPU time. Circuits c2670, b07, and c3540 show reduced backtracks but require more backtraces alleviating the CPU time benefit. Circuits c432, c2670, b07, b13, c6288, b09, b03, c499, b08, b13, c1355 show that quite often, the PCA-assisted ANN-guided PODEM ATPG gives the best guidance, but when does not it is never the worst performing. Circuits c17, b02, b01, b06 have zero reconvergent fanouts and provide no scope for reducing backtracks by the new ANN. Circuit b05 has no reduction in backtracks but a significant reduction in CPU time. This result is significant in terms of the ability to achieve the so-called "sweet-spot" [17], [18].

V. Discussion and Future Work

Minimizing test generation time has been the sole motive for many IC test researchers in the past decades. As the time to generate tests depends on the anatomy of the ATPG algorithm, innovations can improve algorithm's efficacy. The use of heuristics in backtrace is one such technique, which attempts to find a test with minimal bad decisions or "***backtracks***", and more successful "***backtraces***." These two ATPG activities play significant roles in the way the search space is explored for finding tests, trying to minimize the CPU time. The search for a test is terminated as soon as a suitable vector is found, making the exploration of the remaining space unnecessary. Conventional heuristics attempt to enhance ATPG performance, but with the introduction of MI this attempt can be further improved to an extent a single conventional heuristic can never achieve [10], [11]. Past attempts [17] to combine multiple heuristics used statically configured ANN and rudimentary training methodology. The results were promising but left scope to improve further. Hence, the breakthrough discovery of MI-based ATPG, though attractive to the test community, needed more attention in its shortcomings. A structured and formal training methodology potentially enhances backtrack performance and improves CPU time [18].

This article attempts to improve upon the previous MI-based test generation system's detriments by introducing an existing statistical method, known as PCA [4], [5]. The authors of this study think that the ANN of MI-based test generation systems may have many potential features, and their large number could have been a detriment in the past research. Also, finding a "sweet-spot" between "backtracks" and "CPU time" is a challenge, and PCA has the powerful ability to combine even larger-dimensional correlated data, reduce the data volume, and continue to improve the ANN training efficiency. This study went beyond expectation by showing an order of magnitude reduction in test generation time (except for c6288) through effectively combining multiple heuristics.

This study opens up future avenues. First, backtraces of reconvergent fanout free circuits can be improved. Second, finding a "sweet-spot" that demands minimal or optimal backtracks in lowest CPU time. Third, identifying redundant/untestable faults quickly to expedite the ATPG process. Fourth, recent work on MI-based test point insertion (TPI) [13] demonstrates that random circuits could substitute for the use of third-party IP circuits and generate training data with no limits. Fifth, the new method was demonstrated only on academic benchmark circuits where the PC-trained ANN-guided ATPG gave inspiring performance, hence future applications to industrial circuits, yet to be tried, are likely show capabilities. Sixth, continued search for new and effective heuristics, and finally, the efficacy of ANN with more than a single-hidden layer [32] may have potential.

VI. Conclusion

To be successful, an ATPG algorithm must backtrack when necessary and then move forward again. Efficiency is derived by minimizing backtracks. PCA [4], [5] allows to effectively increase the amount of useful information fed through machine intelligence into the ATPG. It brings closer to the elusive goal of zero backtrack. The work presented here shows that PCA reduces the dimension of the training dataset and effectively trains the ANN. This is known as PCA assisted supervised learning in contrast with the supervised learning [17], [18], and can be employed to solve NP-complete problems like ATPG [1].

References

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

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

[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] 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.

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

[6] J. Olsson, C. B. Uvo, K. Jinno, A. Kawamura, K. Nishiyama, N. Koreeda, T. Nakashima, and O. Morita, "Neural Networks for Rainfall Forecasting by Atmospheric Downscaling," *Journal of Hydrologic Engineering*, vol. 9, no. 1, pp. 1–12, 2004.

[7] G. J. Bowden, "Forecasting Water Resources Variables using Artificial Neural Networks," Ph.D. dissertation, University of Adelaide, Australia, 2003.

[8] M. Gibbs, N. Morgan, H. R. Maier, G. C. Dandy, J. B. Nixon, and M. Holmes, "Investigation into the Relationship Between Chlorine Decay and Water Distribution Parameters Using Data Driven Methods," *Mathematical and Computer Modelling*, vol. 44, no. 5, pp. 485–498, 2006.

[9] E. Ranaee, G. Porta, M. Riva, and A. Guadagnini, "Investigation of Saturation Dependency of Oil Relative Permeability during WAG Process through Linear and Non-linear PCA," in *Proc. European Conference on the Mathematics of Oil Recovery*, no. 1. European Association of Geoscientists amp; Engineers, 2014, pp. 1–14.

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

[11] S. Patel and J. Patel, "Effectiveness of Heuristics Measures for Automatic Test Pattern Generation," in *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, 1986, pp. 547–552.

[12] 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.

[13] 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.

[14] Y. Sun, S. K. Millican, and V. D. Agrawal, "Special Session: Survey of Test Point Insertion for Logic Built-in Self-test," in *Proc. IEEE 38th VLSI Test Symposium (VTS)*, 2020, pp. 1–6.

[15] 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.

[16] 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. Test*, vol. 36, no. 1, p. 123–133, Feb. 2020.

[17] 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.

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

[19] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," *ArXiv*, vol. abs/1811.03378, 2018.

[20] 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.

[21] 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, https://doi.org/10.1002/widm.1360.

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

[23] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, *Neural Models and Algorithms for Digital Testing*. Springer, 1991.

[24] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature Cell Biology*, vol. 521, pp. 436–444, 2015.

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

[26] S. Roy, S. K. Millican, and V. D. Agrawal, "Unsupervised Learning in Test Generation for Digital Integrated Circuits," in *Proc. IEEE European Test Symposium (ETS'21)*, 2021.

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

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

[29] W. Kirch, Ed., *Pearson's Correlation Coefficient*. Dordrecht: Springer Netherlands, 2008, pp. 1090–1091.

[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] D. K. Hunter, H. Yu, M. S. P. III, J. Kolbusz, and B. M. Wilamowski, "Selection of Proper Neural Network Sizes and Architectures - A Comparative Study," *IEEE Trans. Industrial Informatics*, vol. 8, no. 2, pp. 228–240, 2012.