Research Article

Single stuck-at-faults detection using test generation vector and deep stacked-sparse-autoencoder

Leila Malihi[1] · Razieh Malihi[2]

Received: 27 November 2019 / Accepted: 28 August 2020 / Published online: 22 September 2020

Abstract
This paper proposed a new method for testing digital circuits without hardware implementation. This data-based method detects hundreds of single stuck-at faults in the ALU circuits, utilizing deep stacked-sparse-autoencoder (SSAE). ATALANTA software is one of the free automatic test pattern generation tools which cover faults in high accuracy. Test vectors which are extracted from bench circuits via ATALANTA software are the key point of the paper. Fault detection is introduced as a two-class problem. SSAE network is trained using the test vectors. Dimension reduction is done automatically in SSAE. Network performance is tested by changing sparse coefficients, number of stacked autoencoder and data augmentation. The results of this step are compared with the traditional multilayer perceptron classification. In this method, unlike SSAE, a manual method of reducing the dimension and extracting the feature is used. Fault coverage of ATALANTA software is over than 94%. Finally, the results obtained from the deep neural network show its significant performance in the circuit faults detection automatically.

Keywords Stacked-sparse-autoencoder · Test vectors · Softmax classifier · MLP

1 Introduction

Home designers are always looking for ways to test digital circuits. Therefore, the students are required to learn the professional skills and software available in this field. Universities can't always use advanced ATPG tools because of the cost of expensive licenses, so using free ATPG tools like ATALANTA can be an effective solution. The advantage of this software is its flexibility related to the test sets, fault dictionary and fault coverage. The input format of this software is named "bench". The use of this format is in benchmark circuit- ISCAS'85 [4] and ISCAS'89 [5]. The most complex circuit of ISCAS'85 is arithmetic-logic unit (ALU). Furthermore, fault detection is an indispensable part of up-to-date industries to guarantee safety and merchandise quality. In the automobile industry, maximum of the fault detection is completed by expert specialists and their conclusion is vastly influenced by their training, skill and differs with time of the day. Since the visual inspection in diagnosing the faults is time consuming and difficult; moreover, it causes opposite observation among specialists, the automatic detection can have a significant help in saving time, human force and reducing errors.

The control unit advises the ALU what task should be performed on that information then the ALU stores the outcome in an output register [1]. Some instances of number arithmetic tasks are addition, subtraction, multiplication, and division. An instance of rationale activities is an examination of qualities, such as NOT, AND or potential [2].

Every one of the faults progressively makes the processors defenseless and increase their structure complexities. Mainly, ALU is really powerless to these faults at inserted processors (i.e., scalar processors), since they have a single

✉ Leila Malihi, Malihi2@yahoo.co.uk | [1]Department of Electrical Engineering, Faculty of Engineering, Shahid Chamran University of Ahvaz, Ahvaz, Iran. [2]Department of Electrical Engineering, Faculty of Engineering, Urmia University, Urmia, Iran.

ALU and a SSAF in the ALU could make the whole processors archaic.

There are some basis automatic test pattern generation (ATPG) algorithms, such as D-algorithm [3], path oriented decision making (PODEM) [4] and FAN [5, 6] to distinguish stuck-at faults. Although these algorithms are efficient, they are not enough to test the system alone and must be combined with a fault-grading mechanism. To automatically produce test vectors in large circuit, these algorithms can be utilized.

Kajihara et al. [7, 8], proposed a parallel method for generating multiple test vector. The analyses of test vectors continued till all of the faults covered or reaches the limitation. If the size of the circuit is large, this process may take too long. Agrawal et al. [9] distinguish a part of faults with generating test vectors and for the remaining faults usages branch and bound technique. These remaining faults may result in a long process time. The combination of genetic algorithm (GA) and ATPG method is presented in paper [10]. This method is composed of repeatedly performing selection, crossover and mutation actions. To receive the optimum solution, it needs to evaluate numerous test pattern techniques.

In the last century, the use of artificial intelligence methods feature extraction and fault detection has expanded. These methods extract feature from the output voltage and input signal of the circuit and compare it with the target to detect the faults. For instance, Fault detection using support vector machine is purposed in [11, 12]. Local minimum, nonlinear and sample size problem can be lost with this classifier. The usage of multi-class relevance vector machine and random forest is presented in [13, 14]. These methods takes a long time to achieve acceptable results.

An artificial neural network is a system of neurons, which adapts complex functions through a progression of nonlinear transformation, with the appearance of deep learning methods [15]. It has been conclusively connected to complex classification errands, such as speech recognition [16]. Artificial neural networks could be adopted additionally to address the fault diagnosis problem [17, 18]. In any case, a large amount of the research works uses shallow neural networks or neural networks with various levelled structures.

One of the newest methods in the field of automatic feature extraction and fault detection is deep learning. High-level feature can be extracted from big data by deep learning method and the performance of the system will be higher than the traditional methods, as in those techniques the features had to be designed individually [19–21]. Mathematical technique can be employed to control the safety and the quality of service for note connection [22]. In [23], the artificial neural network is applied to model the fault detection process and the fault correction process. Testing effort is utilized in this method since it has a high effect on fault detection and correction process. The cryptographic service was chosen to perform privacy defense for messages sent over dispersed cyber physical systems, and organize fault detection within trusted algorithm to oppose fault injection assaults [24].

Stacked sparse autoencoder (SSAE) is a special model in deep learning which can learn set of features. In this way, time can be saved and computational complexity can be reduced.

This paper devises a new plan, in which no necessity of mathematical model for the data exists neither any threshold specification. Another improvement is to propose a methodology that has low complexity with high performance in fault discovery and separation. The purpose of the paper is to identify the single stack-at faults based on the automatic test patterns generation (ATPG). This technique has two stages; in the initial step, test vectors are removed from the ALU circuit utilizing ATALANTA programming. These vectors are utilized in the classification step. In this progress, the dimension of the feature vector reduced utilizing an encoding portion of the autoencoder at the second stage.

The other part of this paper is as follows. Section 2, the methodology has been explained in details. Section 3 shows the results of fault detection and isolation. In this section comparison with other methods are performed. Finally, Sect. 4 presents remarks and conclusion.

2 Materials and methods

To detect SSAF, the first important step is feature extraction. We used ATALANTA software to extract test vectors. After that, SSAE used to detect single stuck-at-faults.

2.1 Alu architecture

The ALU is a digital circuit in the CPU, abbreviated as the arithmetic logic unit. The ALU has the task of performing mathematical operations (such as addition, multiplication, etc.) as well as logical operations on the data. The ALU unit is the last component to perform computation and processing on the data. The ALU unit may contain more than one number in the CPUs. There are three types for this benchmark in following that have explained.

2.1.1 8-bit ALU

This benchmark performs binary and BCD arithmetic, and besides logic and shift operations. Binary and BCD arithmetic functions can be done with this benchmark. Using *two's complement* adder, BCD addition can be done [25].

2.1.2 9-bit ALU

This ALU is useful for arithmetic and logic activities. The parity of outcomes is processed too [25].

2.1.3 12-bit ALU and controller

The components of this benchmark are consist of: an ALU, a comparator, a uniformity checker, and a few equality trees. The inputs of this comparator are 12-bit, and Y > X is computed utilizing a carry-lookahead adder (CLA). 4, 6, 8 or 12-bit comparison of its inputs can be made by programming this comparator. An attractive component of the comparator is that it utilizes two identical CLAs that have identical inputs; an excess method usually utilized in shortcoming tolerant frameworks. The CLAs have a genuinely standard structure with 3, 4 and 5-bit blocks [25].

2.2 Features extraction

2.2.1 Atalanta

ATALANTA programming environment is utilized to do coding test pattern generator for stuck-at fault detection in combinational circuits. The prominent scholastic tools ATALANTA was created at the Virginia Tech University. There are executable files with the required libraries and manual pages [26].

They keep running on Microsoft Windows XP and in advanced versions of working frameworks Windows, an advanced circuit portrayed is utilized in arrangement "bench" as input.

2.2.2 ATPG tool ATALANTA

ATALANTA is suitable just for combinational circuits, and it depends on the FAN calculation [26]. It is utilized to detect and localize the stuck-at-0 and stuck-at-1 faults.

The name of one output file is "<circuit_nam>.test", which is generated and contains test vectors and fault-free output. Test vectors for both of stuck-at faults were generated by ATALANTA. Furthermore, one of the essential parts of ATALANTA is a random test pattern generator (RTPG).

2.3 Classification

Deep learning has been widely studied in the field of computer vision, and for this reason, a large number of related methods have been developed. They are derived into four categories, including convolutional neural networks, restricted Boltzmann machines (RBMS), autoencoders, and sparse coding at last.

To compare the result, the following standard is used [25]:

$$\mathrm{ACC} = \mathrm{TP} + \mathrm{TN} / (\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}) \tag{1}$$

In our case, real positive (TP) and true negative (TN) are defined as correctly predicted abnormal (fault) and normal (No-fault) equipment, respectively. In contrast, false-positive (FP) and false-negative (FN) are defined as incorrectly predicted abnormal and wrong predicted normal equipment correspondingly.

ROC analysis is an evaluation method which is based on statistical decision theory for measuring the classification performance. It manifests the relationship between the fraction of true-positive value (sensibility) and the fraction

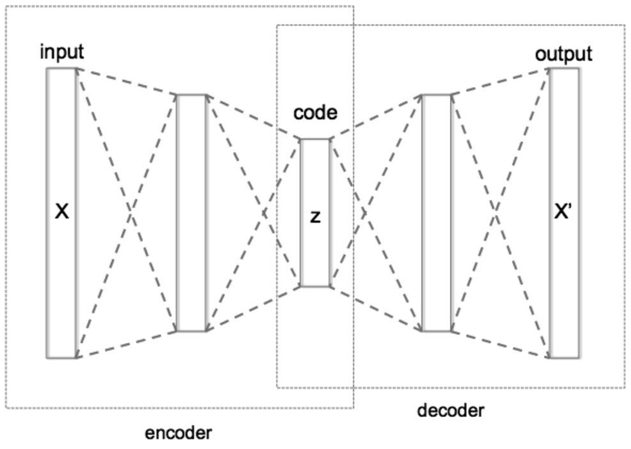


Fig. 1 Autoencoder [29]

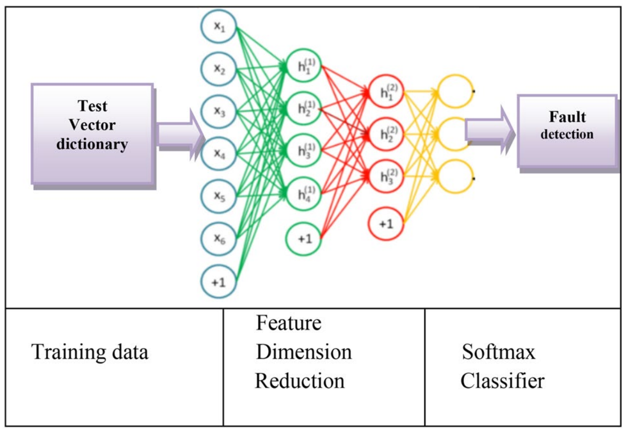


Fig. 2 Fault detection based on SSAE

```
# c3540
# 50 inputs
# 22 outputs
# 490 inverters
# 1179 gates

INPUT(1)
INPUT(13)
INPUT(20)
INPUT(33)
INPUT(41)
INPUT(45)
INPUT(50)
INPUT(58)
INPUT(68)
INPUT(77)
INPUT(87)
INPUT(97)
INPUT(107)
INPUT(116)
```

```
OUTPUT(1713)
OUTPUT(1947)
OUTPUT(3195)
OUTPUT(3833)
OUTPUT(3987)
OUTPUT(4028)
OUTPUT(4145)
OUTPUT(4589)
OUTPUT(4667)
OUTPUT(4815)
OUTPUT(4944)
OUTPUT(5002)
OUTPUT(5045)
OUTPUT(5047)
OUTPUT(5078)
OUTPUT(5102)
OUTPUT(5120)
OUTPUT(5121)
OUTPUT(5192)
OUTPUT(5231)
```

```
763 = NOT(116)
768 = OR(257, 264)
769 = NOT(1)
772 = BUFF(1)
779 = NOT(1)
782 = BUFF(13)
786 = NOT(13)
793 = AND(13, 20)
794 = NOT(20)
798 = BUFF(20)
803 = NOT(20)
820 = NOT(33)
821 = BUFF(33)
825 = NOT(33)
829 = AND(33, 41)
832 = NOT(41)
835 = OR(41, 45)
836 = BUFF(45)
839 = NOT(45)
842 = NOT(50)
```

Fig. 3 Bench format of 8 bit ALU

```
# c5315
# 178 inputs
# 123 outputs
# 581 inverters
# 1726 gates

INPUT(1)
INPUT(4)
INPUT(11)
INPUT(14)
INPUT(17)
INPUT(20)
INPUT(23)
INPUT(24)
INPUT(25)
INPUT(26)
INPUT(27)
INPUT(31)
INPUT(34)
INPUT(37)
INPUT(40)
INPUT(43)
INPUT(46)
INPUT(49)
INPUT(52)
INPUT(53)
INPUT(54)
```

```
OUTPUT(709)
OUTPUT(816)
OUTPUT(1066)
OUTPUT(1137)
OUTPUT(1138)
OUTPUT(1139)
OUTPUT(1140)
OUTPUT(1141)
OUTPUT(1142)
OUTPUT(1143)
OUTPUT(1144)
OUTPUT(1145)
OUTPUT(1147)
OUTPUT(1152)
OUTPUT(1153)
OUTPUT(1154)
OUTPUT(1155)
OUTPUT(1972)
OUTPUT(2054)
OUTPUT(2060)
OUTPUT(2061)
OUTPUT(2139)
OUTPUT(2142)
OUTPUT(2309)
OUTPUT(2387)
OUTPUT(2527)
OUTPUT(2584)
```

```
1161 = BUFF(571)
1173 = BUFF(574)
1185 = BUFF(571)
1197 = BUFF(574)
1209 = BUFF(137)
1213 = BUFF(137)
1216 = BUFF(141)
1219 = NOT(583)
1223 = BUFF(577)
1235 = BUFF(580)
1247 = BUFF(577)
1259 = BUFF(580)
1271 = BUFF(254)
1280 = BUFF(251)
1292 = BUFF(251)
1303 = BUFF(248)
1315 = BUFF(248)
1327 = BUFF(610)
1339 = BUFF(607)
1351 = BUFF(613)
1363 = BUFF(616)
1375 = BUFF(210)
1378 = BUFF(210)
1381 = BUFF(218)
1384 = BUFF(218)
1387 = BUFF(226)
1390 = BUFF(226)
```

Fig. 4 Bench format of 9 bit ALU

of false-positive value (1-specificity) with the variations of the decision threshold [27].

2.3.1 Data augmentation

Deep networks need a large amount of training data to have high accuracy. If the training data is very little, image augmentation will require to increase the accuracy of deep networks. Image augmentation artificially can be done by random rotation, shifts, shear and flips, etc.

2.3.2 Autoencoder

Autoencoder is a specific type of artificial neural network used to encode learning, and then uncompressed that code into something which closely matches the original data. Instead of training the network and predicting the target value of Y, in return for the X input, an autoencoder will be trained to rebuild the X input [28]. In Fig. 1, the architecture of the autoencoder consist of 3 layers: encoders, code and decoder is presented. The encoder comperes the input data and produces the code, after that the decoder reconstruct the input using this code.

This autoencoder is frequently trained via one of the many back propagation methods. One problem of getting back propagated to the first few layers is being tiny and ineffective. This reasons the network to rebuild the average of all the training data.

3 Sparse auto-encoder

By applying sparse limitations on hidden units, autoencoder detects fascinating structures in the data, even if the number of hidden layers is high. Informally, if the output of a neuron is close to 0, that neuron will be "active" (or as "firing"), and if its output value is close to 1, that neuron will be "inactive". A_j^2 indicates the activation of hidden unit j in the autoencoder. However, this symbol does not indicate that the input x is activated. Further, let [28]:

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^2 x^i\right] \tag{2}$$

We would like to (approximately) impose restrictions $\hat{\rho}_j = \rho$. Sparsity parameter is shown as ρ. It is better to set hidden neuron j to be close to 0.05 (say). To meet this limitation, the activities of the hidden unit are mostly close to 0 [28].

3.1 Stacked sparse auto-encoder

One kind of deep neural network is stacked sparse auto-encoder (SSAE). The main components of SSAE are many sparse auto-encoders (SAE) and a softmax classifier. The output of each SAE linked to the next SAE's input. First of all, each SAE and the softmax layer are trained. After training the model, feature extraction and fault classification can be done [28]. More details about the layers are as follows:

3.1.1 The first SAE

The main layers of an auto-encoder (AE) are the input layer, hidden layer and reconstruction layer, respectively [28]. First, the input data is encoded using a hidden layer, and then reconstruction is done. In Fig. 2, the architecture of purposed model is shown.

3.1.2 The second SAE

The output of the first SAE is the second SAE input, and the training process is like the first one. For other SAE is as the second one.

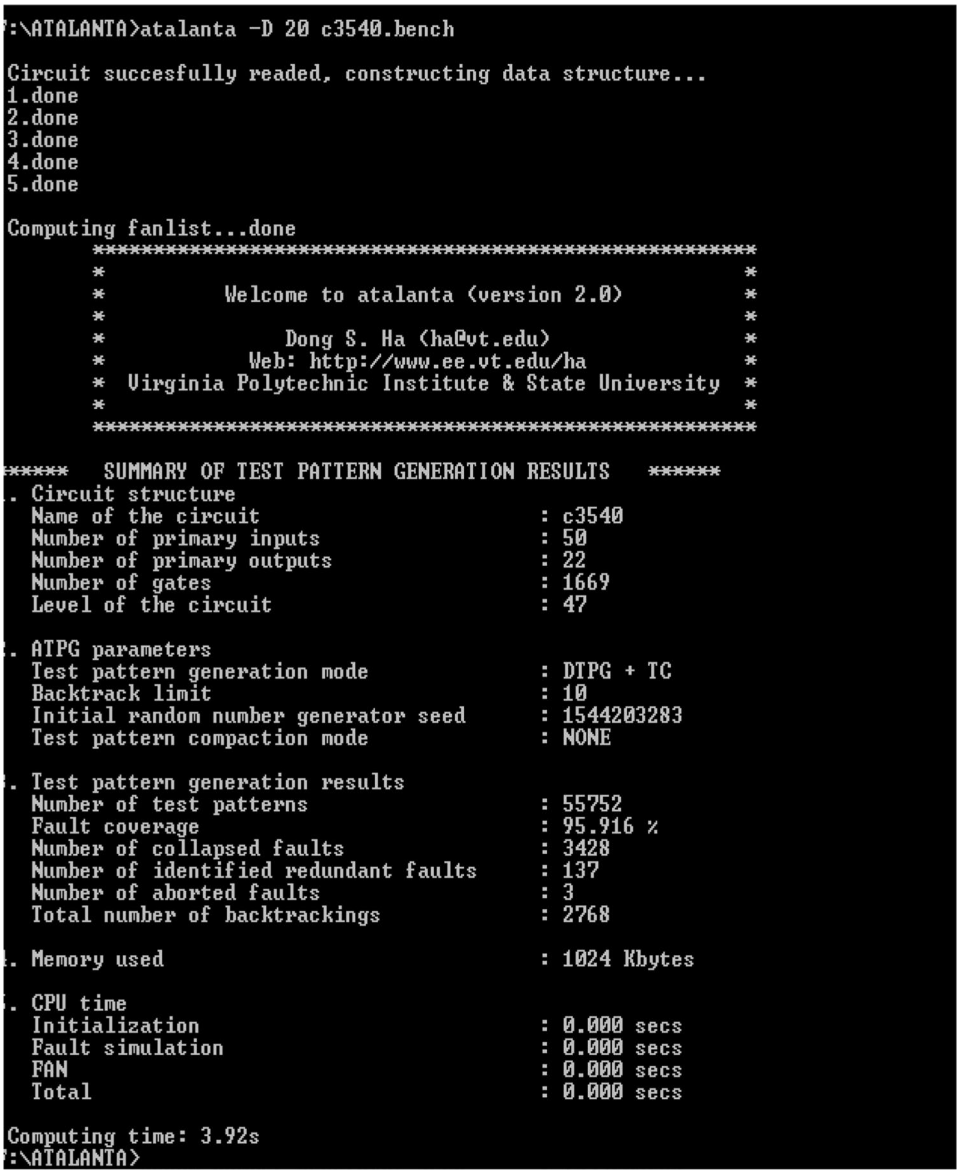
3.1.3 The softmax layer

In machine learning, the Softmax function is used to classify. For example, if there are "n" different classes and the machine learning algorithm does not directly produce the

```
# c2670                  OUTPUT(143)          659 = NOT(266)
# 233 inputs             OUTPUT(144)          663 = NOT(269)
# 140 outputs            OUTPUT(145)          667 = NOT(272)
# 321 inverters          OUTPUT(146)          671 = NOT(275)
# 872 gates              OUTPUT(147)          675 = NOT(278)
                         OUTPUT(148)          679 = NOT(281)
INPUT(1)                 OUTPUT(149)          683 = NOT(284)
INPUT(2)                 OUTPUT(150)          687 = NOT(287)
INPUT(3)                 OUTPUT(151)          693 = BUFF(29)
INPUT(4)                 OUTPUT(152)          699 = BUFF(29)
INPUT(5)                 OUTPUT(153)          705 = NOT(294)
INPUT(6)                 OUTPUT(154)          711 = NOT(297)
INPUT(7)                 OUTPUT(155)          715 = NOT(301)
INPUT(8)                 OUTPUT(156)          719 = NOT(305)
INPUT(11)                OUTPUT(157)          723 = NOT(309)
INPUT(14)                OUTPUT(158)          727 = NOT(313)
INPUT(15)                OUTPUT(159)          730 = NOT(316)
INPUT(16)                OUTPUT(160)          733 = NOT(346)
INPUT(19)                OUTPUT(161)          734 = NOT(349)
INPUT(20)                OUTPUT(162)          735 = BUFF(259)
INPUT(21)                OUTPUT(163)          738 = BUFF(256)
INPUT(22)                OUTPUT(164)          741 = BUFF(263)
INPUT(23)                OUTPUT(165)          744 = BUFF(269)
INPUT(24)                OUTPUT(166)          747 = BUFF(266)
INPUT(25)                OUTPUT(167)          750 = BUFF(275)
INPUT(26)                OUTPUT(168)          753 = BUFF(272)
INPUT(27)                OUTPUT(169)          756 = BUFF(281)
```

Fig. 5 Bench format of 12 bit ALU

Fig. 6 20 test vectors, selected from 8 bit ALU



```
:\ATALANTA>atalanta -D 20 c3540.bench

Circuit succesfully readed, constructing data structure...
1.done
2.done
3.done
4.done
5.done

Computing fanlist...done
       ******************************************************
       *                                                    *
       *          Welcome to atalanta (version 2.0)         *
       *                                                    *
       *               Dong S. Ha (ha@vt.edu)               *
       *            Web: http://www.ee.vt.edu/ha            *
       *  Virginia Polytechnic Institute & State University *
       *                                                    *
       ******************************************************

*****   SUMMARY OF TEST PATTERN GENERATION RESULTS   ******
. Circuit structure
  Name of the circuit                       : c3540
  Number of primary inputs                  : 50
  Number of primary outputs                 : 22
  Number of gates                           : 1669
  Level of the circuit                      : 47

. ATPG parameters
  Test pattern generation mode              : DTPG + TC
  Backtrack limit                           : 10
  Initial random number generator seed      : 1544203283
  Test pattern compaction mode              : NONE

. Test pattern generation results
  Number of test patterns                   : 55752
  Fault coverage                            : 95.916 %
  Number of collapsed faults                : 3428
  Number of identified redundant faults     : 137
  Number of aborted faults                  : 3
  Total number of backtrackings             : 2768

. Memory used                               : 1024 Kbytes

. CPU time
  Initialization                            : 0.000 secs
  Fault simulation                          : 0.000 secs
  FAN                                       : 0.000 secs
  Total                                     : 0.000 secs

Computing time: 3.92s
:\ATALANTA>
```

probability of input in each of these classes, instead, take n real numbers corresponding to the input position of each class. The following function can be used to convert real points to probability so that the conditions for defining a probability mass function are fulfilled [28]:

$$P(Y=j\,|\,x_i) = \frac{e^{x_i\theta_j^r}}{\sum_{l=0}^{k-1} e^{x_i\theta_l^r}} \tag{3}$$

The total number of classes is shown as K; furthermore, θ_j is the parameter vector of class j [30].

$$y_i^{pred} = \arg\max P(Y=j\,|\,x_i) \tag{4}$$

y_i^{pred} is the predicted class for sample x_i [31].

4 Experimental results

4.1 Data collection

The academic ATALANTA software was developed at the Virginia Tech University. The "bench" circuit is the input format of this software. In the first step, the bench format of the ALU circuits are programed in the ATALANTA. The bench coding of 8 bit ALU is as Fig. 3. The 9 and 12 bit ALU bench coding is shown in Figs. 4 and 5 respectively. Comment is shown as symbol # at the beginning of the line. Inputs and outputs are defined after the comments and then each line stipulates the function of one logical gate in the circuit.

The character # in the Figs. 3, 4, 5 is defined as comment. After input and output describing, the logical model of ALU circuit has been defined. The stuck-at-0 and stuck-at-1 faults are recognized in the ATALANTA. The output of ATLANTA on ALU circuits is shown as follows:

In Fig. 6, 20 test vectors were selected for each fault. 55,752 test vectors were extracted from 8 bit ALU totally.

According to Table 1, the number of faults for 8 bit, 9 bit and 12 bit ALU are 3288, 5285 and 2584 respectively. 20 test vectors were randomly selected for each fault, so the total number of test vectors are 55,752, 88,978 and 39,725 respectively. ATALANTA software has a high fault coverage. The minimum fault coverage is 94.06% for 12 bit ALU and the maximum fault coverage is 98.78% for 9 bit ALU.

ATALANTA is based on the FAN algorithm. After running the code, test vector in the form of a "<circuit name>.test" file is generated. For instance a set test vector for the C3540 circuit is generated as C3540.test. A part of these test vectors is shown as Table 2.

4.2 Parameter setting and results

Network parameters affect network performance, therefore selection the most optimized ones is essential.

The number of classes and neurons are the number of faults. According to Table 1, the number of faults in 8 bit ALU are 3288, in 9 bit ALU are 5285 and in 12 bit ALU are 2584. So the number of classes in 8 bit ALU are 3288, in 9 bit ALU are 5285 and in 12 bit ALU are 2584. The input data for 8 bit ALU has 55,752 × 72 dimensions, for 9 bit ALU has 94,328 × 72 dimensions and for 12 bit ALU has 42,472 × 373 dimensions.

Table 1 Number of faults and fault coverage percentage and number of test vector

Circuit	8 bit	9 bit	12 bit
Number of faults	3288	5285	2584
Fault coverage %	95.91	98.78	94.06
Number of test vector	55,752	88,978	39,725

Table 2 Test vectors

Test vector 1	1001011X11001X1000…101X
Test vector 2	110X11001111001X11…0011
Test vector 3	111111001X100X0011…11X0
Test vector 4	100X0011X1110101X1…0101
Test vector 5	1X10X010010X110101…1010
Test vector 6	0100X1001001X11010…1100
Test vector 7	X10X1101010X0101X…101X
Test vector 8	01X110100101X10101…0010

The main parameters of the SSAE are the number of hidden layers and hidden neurons; the number of hidden layers is set from 1 to 3, and the number of hidden neurons are set to {180}, {100}, and {60} experimentally.

According to Fig. 7, the amount of ρ sets to 0.15 for the first autoencoder, sets to 0.3 for the second autoencoder and sets to 0.25 for the third autoencoder to get high accuracy. The parameters of MLP have been set in Table 3. The number of neurons in the output layer set to the number of faults. For instance, there are 3288 faults in 8 bit ALU, hence the number of output neurons are 3288.

4.3 Evaluation of the proposed method

Mean squared error used for evaluation of the training process. In Fig. 8, for 8 bit ALU, the error is 0.0007 at epoch 26, for 9 bit ALU the error is 0.0001 at epoch 26 and for 12 bit ALU, the error is 0.0004 at epoch 26. As it was shown, the training performance is going higher when the number of iterations increases. However, the model's training time is growing linearly. Therefore we should find a tradeoff between the accuracy and the spent time.

The best validation performances using MLP neural network were shown in Fig. 9. There are three plots in red, blue, green color. For the 8 bit ALU, the validation plot (blue) is higher than others. The best performance is 97.5% accuracy for 8 bit ALU. For 9 bit ALU, the validation performance is higher. The mean squared error is 46.2022 at epoch 52 in 9 bit ALU and for 12 bit ALU this value reduces to 18.947.

The comparison results of two fault detection methods are shown in Table 4. According to this table, SSAE in all of ALU circuit has the best accuracy. The results of 8 bit ALU are almost the same in both methods. In others, SSAE has the high accuracy.

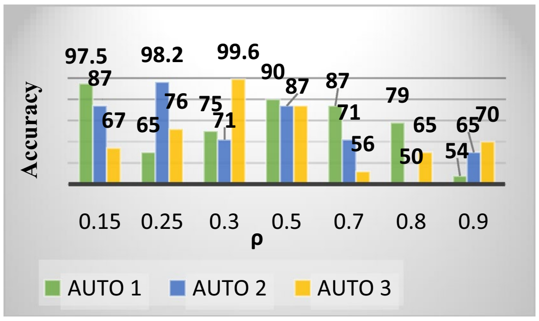


Fig. 7 The effect of ρ changes on accuracy

Table 3 Parameters and configurations for the MLP neural network

Network	Feedforward backpropagation
Number of training epoch	500
Error goal	1.00E_03
Learning rate	0.1
Training function	Levenberg–Marquardt back propagation
Hidden layer activation function	Logsig
Number of neurons in the output layer	Number of faults
Number of neurons in the hidden layer	60
Output layer activation function	Linear

5 Conclusion

This paper presents a new method for stuck-at fault detection in ALU circuits. ATALANTA software is utilized to extract test vectors from ALU circuits. By stacking three sparse autoencoders and using the softmax classifier as the output layer, the fault detection model is constructed. The SSAE is used in unsupervised learn high-level features from the dataset. Fault detection accuracy was 99.6% for 12 bit ALU. In other method MLP classifier is used. In this neural network number of neurons in output layer is the number of faults. There isn't any automatic feature reduction. Accuracy decrease to 81.1% for 12 bit ALU. By comparing the proposed method with MLP, the experimental results show that

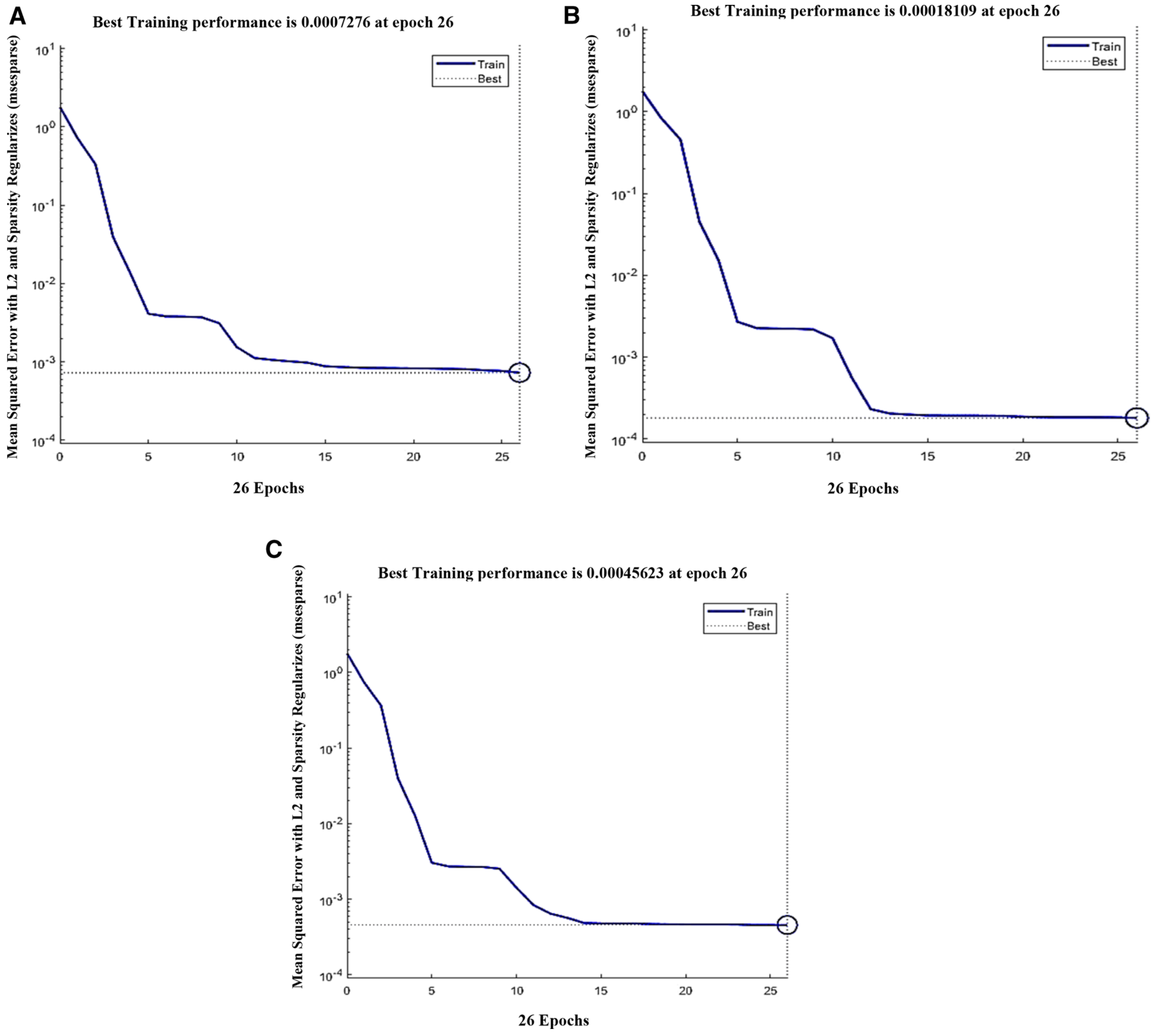


Fig. 8 Best training performance. **a** 8 bit ALU, **b** 9 bit ALU, **c** 12 bit ALU

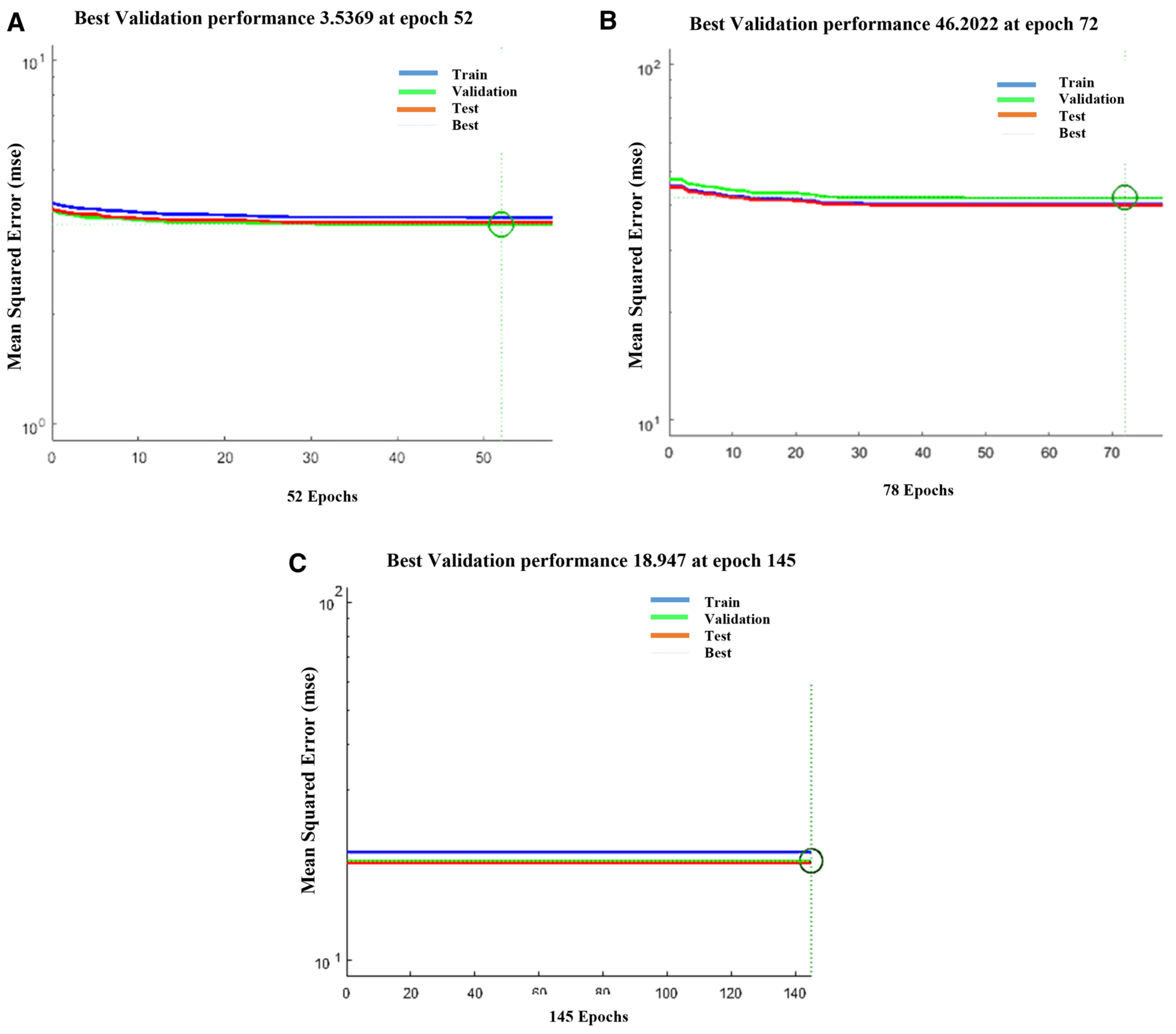


Fig. 9 Validation performance. **a** 8 bit ALU, **b** 9 bit ALU **c** 12 bit ALU

Table 4 Fault detection and location accuracy of different methods

Circuit accuracy %	8 bit	9 bit	12 bit
MLP	97.5	53.8	81.1
SSAE	97.9	98.2	99.6

the proposed method achieves high fault detection accuracy. The effect of network parameters on network results is noticeable. Increasing the Spars coefficient amount reduces the accuracy. The best accuracy for fault detection is related to 12 bit ALU which is 99.6%. Using test vectors as features due to high accuracy for fault detection.

Acknowledgments This study was funded by Shahid Chamran University (1000$).

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Takahashi I, Boateng KO, Takarnatsu Y (1999) A new method for diagnosing multiple stuck-at faults using multiple and single fault simulations. In: Proceedings of the IEEE VLSI test symposium, pp 64–69
2. https://study.com/academy/lesson/arithmetic-logic-unit-alu-definition-design-function.html
3. Roth JP et al (1968) Diagnosis of automata failures. IBM J Res Dev 10(4):278–291
4. Goel P (1981) An implicit enumeration algorithm to generate tests for combinational logic circuits. IEEE Trans Comput C-30(3):215–222
5. Fujiwara H, Shimono T (1983) On the acceleration of test generation algorithms. IEEE Trans Comput C-32(12):1137–1144
6. Fujiwara H (1985) Fan: a fanout-oriented test pattern generation algorithm. In: Proceedings of IEEE international symposium circuits system, pp 671–674
7. Kajihara S, Sumioka T, Kinoshita K (1993) Test generation for multiple faults based on parallel vector pair analysis. In: Proceedings of IEEE/ACM ICCAD, Santa Clara, CA, USA, pp 436–439
8. Kajihara S, Nishigaya R, Sumioka T, Kinoshita K (1994) Efficient techniques for multiple fault test generation. In: Proceedings of IEEE ATS, Nara, Japan, pp 52–56
9. Agrawal A, Saldanha A, Lavagno L, Sangiovanni A. L (1997) Compact and complete test set generation for multiplestuck-faults. In: Proceedings of IEEE/ACM ICCAD, San Jose, CA, USA, pp 212–219
10. Anita JP, Vanathi PT (2014) Genetic algorithm based test pattern generation for multiple stuck-at faults and test power reduction in VLSI circuits. In: Proceedings of 2014 international conference on electronics and communication systems (ICECS), pp 1–6
11. Qing HU, Wang RJ, Zhan YJ (2008) Fault diagnosis technology based on SVM in power electronics circuit. Proc Chin Soc Electr Eng 28(12):107–111
12. Cui J, Wang Y, Liu Q (2007) The technique of power electronic circuit fault diagnosis based on higher-order spectrum analysis and support vector machines. Proc Chin Soc Electr Eng 10(10):62–66
13. Wang T, Xu H, Han J et al (2015) Cascaded H-bridge multilevel inverter system fault diagnosis using a PCA and multi-class relevance vector machine approach. IEEE Trans Power Electr 30(12):1
14. Cai JD, Yan RW (2009) Fault diagnosis of power electronic circuit based on random forests algorithm. In: 2009 fifth international conference on natural computation, pp 214–217
15. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
16. Hinton G, Deng L, Yu D, Dahl G, Mohamed A, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath T et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Proc Mag 29(6):82–97
17. Eslamloueyan R (2011) Designing a hierarchical neural network based on fuzzy clustering for fault diagnosis of the tennessee-eastman process. Appl Soft Comput 11(1):1407–1415
18. Zhang Z, Zhao J (2017) A deep belief network based fault diagnosis model for complex chemical processes. Comput Chem Eng 107:395–407
19. Dahl GE, Yu D, Deng L et al (2012) Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans Audio Speech Lang Process 20(1):30–42
20. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527
21. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems. Curran Associates Inc. pp 1097–1105
22. Xie Y, Zeng H, Hu C, Ji Z (2019) Security/timming-aware design space exploration of CAN FD for automative cyber-phsical systems. IEEE Trans Ind Inf 15(2):1094–1104
23. Xiao H, Cao M, Peng R (2020) Artificial neural network based software fault detection and correction prediction models considering testing effort. Appl Soft Comput 94:106491
24. Jiang W, Wen L, Zhan J, Jiang K (2020) Design optimization of confidentiality-critical cyber physical systems with fault detection. J Syst Archit 107:101739
25. http://web.eecs.umich.edu/~jhayes/iscas.restore/
26. User's Guide for ATALANTA (1991) Virginia Polytechnic & State University. https://github.com/hsluoyz/Atalanta
27. Fawcett T (2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874
28. Xu J, Xiang L, Liu Q et al (2016) Stacked sparse autoencoder (SSAE) for nuclei detection of breast cancer histopatholog images. IEEE Trans Med Imaging 35(1):119–130
29. Fujiwara H, Shimono T (1983) On the acceleration of test generation algorithms. IEEE Trans Comput 32(12):1137–1144
30. Qiaoxuan Y, Bin D et al (2017) Stacked sparse autoencoder based fault detection and location method for modular five level converters. In: IECON 2017—43rd annual conference of the IEEE industrial electronics society
31. Goodfellow L, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge