Regeneration of Test Patterns for BIST by Using Artificial Neural Networks

Tsutomu Inamoto[1] and Yoshinobu Higami[1]

[1]Graduate School of Science and Engineering, Ehime University
3, Bunkyo-cho, Matsuyama, Ehime 790-8577, Japan
E-mail : [1]inamoto@ehime-u.ac.jp

Abstract: In this paper, we display an approach to detect circuit faults by the built-in self test (BIST) technology. In the BIST for a certain circuit, it is usual to generate test patterns by feeding their seed values to a test pattern generator (TPG), which is contained in a device together with the circuit. It is ideal but impractical to make the device to contain a digital memory that stores effective test patterns. The key idea of the presented approach is to use the artificial neural network (ANN) as such memory on the expectation that an ANN can be implemented as an analog circuit. In addition, this paper investigates the inaccuracy that is inevitable regarding analog components.

Keywords: LSI testing, fault detection, BIST, artificial neural network

1. Introduction

Nowadays people use highly condensed Large-Scale Integrated circuits (LSIs) in their electric appliances such as smartphones in their everyday life. Corrupted LSIs can easily break down the world, and thus it is necessary to make efforts to keep LSIs reliable. One of such efforts by LSI companies is to filter out corrupted LSIs before their shipment to customers. This production test is obviously important. In addition, it is going to become important to test LSIs operated in the field, since more and more LSIs are used outdoors in this era of IoT. The production test is useless for such in-field test, and another methodology is necessary.

One solution to keep LSIs in the field reliable is the built-in self test (BIST)[1]. In the BIST technology, a device is designed to contain a subject LSI and a test pattern generator (TPG). A test pattern for the LSI is a vector of values for primary inputs of the LSI. Given a test pattern, the output values of primary outputs of the LSI can be obtained before the in-field test by applying the test pattern to the simulated LSI. If observed output values are different from those have been obtained by simulating the fault-free LSI, we can say that at least one fault has occurred in the LSI. In this sense, it is important for a TPG how many faults can be detected by test patterns generated by it. The Linear Feedback Shift Register (LFSR)[1] can be used as the TPG. If the number of primary inputs is m, then an appropriate LFSR can generate all $2^m - 1$ patterns except for the zero vector. This is a strong point of using the LFSR as the TPG. However, the LFSR has such a weak point that it is unavoidable to generate useless bit vectors that cannot detect any fault and a long time is required to examine most of possible faults.

The ideal technique that yields most effective test patterns is to store test patterns known effective in a digital memory. However, this technique is impractical since the number of such test patterns is usually numerous and the cost of implementing such digital memory is high. Here, we can see that such a memory represents a mapping from an address to a test pattern. Such mapping is expected to be represented by the artificial neural network (ANN), which is widely applied nowadays owing to the development of the deep learning technology [2].

In this paper, we focus on an approach [3] to improve fault coverages by using the ANN. This approach is based on the expectation that the ANN can be implemented as an analog circuit. In the approach, effective test patterns are prepared beforehand, and each test pattern is associated with a binary vector that represents the address of that pattern. The mapping from such an address to a test pattern is learned in an ANN in software. The ANN is expected to be implemented as an analog circuit. Effective test patterns will be regenerated by using such ANN circuit and a counter circuit which covers all addresses of the test patterns. This design is sketched in Figure 1. In computational experiments, the approach is evaluated by using two circuits of the typical benchmark sets [4,5]. In addition, the inaccuracy, which is inevitable regarding analog components, is investigated by disturbing weights of ANNs.

2. LSI Testing Problem

2.1 Fault model

Regarding the logic circuit, there are many fault models such as the bridging fault, the open fault, the delay fault, and the stuck-at fault [1]. In this paper, the stuck-at fault model is

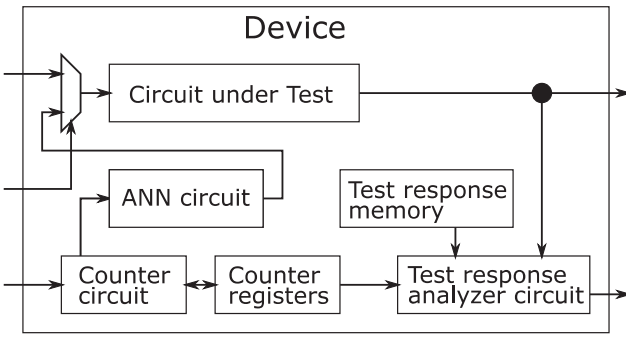

Figure 1. Sketch of BIST device that contain ANN circuit as TPG

considered since it is most basic. In this fault model, a faulty line in a circuit has the logic value "0" (stuck-at-0, sa0) or "1" (stuck-at-1, sa1) no matter what input signals are applied to primary inputs of the circuit. In addition, it is assumed that the number of faults occurred in a circuit is one (single fault model) since a circuit that has multiple faults can be labeled faulty by detecting one of them. The authors also have interested in the transition delay fault model. However, the presented research is too preliminary to apply to such a more complicated fault model, so the application to other delay models is left as one of future tasks. This paper considers only the combinational logic circuit and the scan circuit, since the former is most basic and the latter is widely used.

2.2 Fault detection problem

Two representative technologies for LSI testing are the fault detection and the fault diagnosis [1]. The objective of the fault detection is to judge whether a subject circuit (Circuit under Test, CUT) has faults or not. The objective of the fault diagnosis is to distinguish which faults occurred in the CUT. This paper focuses on the fault detection.

In the fault detection, we can say that a CUT is faulty by using a test pattern if its observed output vector differs from the fault-free output vector. Let us denote by $\mathbb{t} \in \{0,1\}^{N^{\mathrm{I}}}$ a test pattern that is a bit vector for primary inputs of the CUT, where N^{I} denotes the number of primary inputs of the CUT. Let us denote by $G_k(\mathbb{t}) \in \{0,1\}^{N^{\mathrm{O}}}$ the output vector of the CUT when it has fault $k \in \hat{\mathcal{F}} \coloneqq \{-N^{\mathrm{L}}, -N^{\mathrm{L}}+1, \dots, -1, 0, 1, \dots, N^{\mathrm{L}}-1, N^{\mathrm{L}}\}$ and is applied test pattern $\mathbb{t}$, where N^{O} and N^{L} denote the number of primary outputs and the number of lines of the CUT, respectively. Here, it should be noted that $k = 0$ means there is no fault, and $-k$ or k represents an sa0 or sa1 fault has occurred at line k, respectively. By using these symbols, the ideal goal of the test pattern generation is said to find such $\mathbb{t}$ for all faults $k \in \mathcal{F} \coloneqq \hat{\mathcal{F}} \setminus \{0\}$ that satisfies $G_0(\mathbb{t}) \neq G_k(\mathbb{t})$. The fault detection is usually conducted by using a test pattern set $\mathcal{T}$ comprised of test patterns. The fault coverage $f(\mathcal{T})$ of test pattern set $\mathcal{T}$ is the fraction of the number of detected faults to the number of all faults; that is,

$$f(\mathcal{T}) \coloneqq \frac{|\{k \in \mathcal{F} \mid \sum_{\mathbb{t} \in \mathcal{T}} d_k(\mathbb{t}) \geq 1\}|}{|\mathcal{F}|} . \quad (1)$$

Here, $d_k(\mathbb{t})$ denotes whether test pattern $\mathbb{t}$ can detect fault k or not and is defined as follows:

$$d_k(\mathbb{t}) \coloneqq \begin{cases} 1 & \text{if } G_0(\mathbb{t}) \neq G_k(\mathbb{t}), \\ 0 & \text{otherwise}. \end{cases} \quad (2)$$

3. Regenerating Deterministic Test Patterns by Artificial Neural Networks

3.1 Focused approach

The approach [3] focused in this paper assumes that an effective test pattern set is given beforehand and expects that the ANN can be implemented as an analog circuit. The latter assumption is too optimistic and some detailed comparisons between analog and digital architectures as like [6] are necessary in implementing circuits. However, this paper is too preliminary to consider how to implement, and such comparisons are left as future tasks. We envisage the implementation is easier than usual cases, since an ANN circuit in this approach abandons the learning ability. The approach is comprised of the following three stages: (i) to associate binary vectors with test patterns of the given test pattern set, (ii) to train an ANN that represents the mapping from binary vectors to test patterns, and (iii) to implement the trained ANN as an analog circuit. The stages (i) and (ii) are considered in this paper. The investigation on stage (iii) is left as a future task. We think it is a unique point of the approach that an ANN is deployed as a mapping function, whereas an ANN circuit is usually deployed to represent a scalar function. An example of such function is a non-linear classifier that decides whether a circuit is faulty or not [7].

Table 1. Characteristics of circuits and test patterns

Circuit	Number of inputs	Number of lines	Number of faults	Number of test patterns
c7552	207	7,660	15,104	108
s38584	1,464	40,162	36,303	205

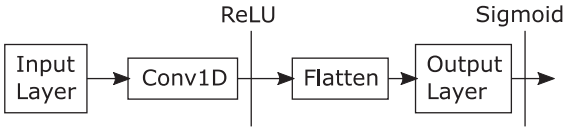

Figure 2. Network design of ANNs

Let us describe the approach by taking circuit c7552, which characteristics are displayed in Table 1, as an example. That circuit has 207 primary inputs, and 108 test patterns are given. These test patterns are said complete since they can detect all of detectable stuck-at faults in that circuit. In stage (i), [0, 107] integers are represented as 7-bit vectors then associated with 108 test patterns. In stage (ii), an ANN is constructed to obey the design that uses as few layers as possible. This design is displayed in Figure 2. The ANN is trained by using 7-bit vectors and test patterns as its inputs and outputs, respectively. After training, a test pattern is generated by feeding a 7-bit vector to the ANN.

Let us compare the amount of hardware resources required in holding test patterns in a digital memory and those required in implementing an ANN circuit. If R^{b} transistors are necessary to hold one bit, then the digital memory requires $R^{\mathrm{b}} N^{\mathrm{I}} |\mathcal{T}|$ transistors. The number of neurons in the input layer is $\mathrm{m} \coloneqq \lceil \log_2 |\mathcal{T}| \rceil$. In this paper, it is designed according to our experience that the number of filters in the Conv1D layer and the size of each filter are $2m$ and m, respectively. The number of weights from the input layer to the Conv1D layer is $(m+1) \cdot 2m$, where "+1" represents the bias term. The number of weights from the Conv1D layer to the output layer is $(2m \cdot m + 1)N^{\mathrm{I}}$. If a weight in the ANN is implemented as an op-amp comprised of αR^{b} transistors and N^{I} is sufficiently larger than m^2, then the fraction of the number of transistors for the digital memory to the number of those for

the ANN circuit is roughly estimated as follows:

$$\frac{R^{\mathrm{b}}N^{\mathrm{I}}2^m}{\alpha R^{\mathrm{b}}(2m^2+2m+(2m^2+1)N^{\mathrm{I}})} \simeq \frac{2^m}{\alpha(2m^2+1)}. \quad (3)$$

3.2 Consideration on inaccuracy

An analog circuit is inaccurate since the signal is disturbed by noise and it is impossible to produce analog components that characteristics are identical to those in theories. The inaccuracy has to be considered in order to realize the focused approach. This paper considers the inaccuracy as disturbances on weights of ANNs. Let us assume that an ANN was trained to regenerate test patterns. Before we obtain test patterns by using the ANN, the ANN is disturbed by adding a random value to each weight of the ANN. If the weight is w, then the random value is designated to obey $N(0, r|w|)$, where $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ, and $r\ (>0)$ is a parameter that represents the magnitude of disturbances. Of course, this treatment of the inaccuracy is tentative. Future tasks include using the SPICE simulator [6] and disturbing parameters of SPICE models.

4. Computational Experiments

4.1 Experimental setup

The focused approach was evaluated by calculating fault coverages of c7552 of the ISCAS '85 benchmark [4] and s38584 of the ISCAS '89 benchmark [5]. s38584, which is originally a sequential circuit, was converted to a scan circuit and treated as a combinational circuit. A test pattern set was used for each circuit that had been prepared in the authors' laboratory. Each test pattern set is complete in the sense that it can detect all detectable stuck-at faults. The characteristics of those circuits and test pattern sets are displayed in Table 1.

To obtain a baseline of the fault coverage, the LFSR was used as a typical TPG. The LFSR has N^{I} registers and they are connected in an internal-XOR form. The initial values of registers in the LFSR were randomly set for 10 times. For each circuit and for each initial set of register values, a test pattern set was generated. The numbers of generated test patterns were equal to those displayed in Table 1.

Table 2. Parameter setting for ANNs

Parameter	Value
Optimizer	Adam with default parameter setting
Activation function	ReLU, Sigmoid
Number of filters in Conv1D	$2N^{\mathrm{I}}$
Window size of filters in Conv1D	N^{I}
Training epoch	100,000
Batch size	256
Loss function	binary cross-entropy
Disturbance magnitude (r)	0.05

Table 3. Fault coverages by LFSRs [%]

Circuit	Min.	Avg.	Max.
c7552	91.45	92.49	93.70
s38584	89.65	90.38	91.32

The network design of used ANNs is displayed in Figure 2. This design is brought by such a consideration that a shallow network seems appropriate for decreasing the time in generating test patterns. Other parameters are displayed in Table 2. The ANNs were built, trained, then used for 10 times with different pseudo-random numbers.

In implementing programs, Python 3.6.8 was used with packages lfsr 0.2, tensorflow 2.0.0, and Keras 2.3.0. All computations were conducted on a computer that is equipped with dual Xeon E5-2640 v2 2.0 [GHz] and 64 [GB] main memory, and is operated by CentOS 7.7.

4.2 Results on LFSRs

Ten test pattern sets were generated by LFSRs. Their statistics on fault coverages are displayed in Table 3. In calculating fault coverages, undetectable (redundant) faults were removed from $\mathcal{F}$ so as to make possible to attain 100% fault coverage. Table 3 displays that fault coverages are not high since the numbers of test patterns are few.

4.3 Results on noise-free ANNs

For each circuit, 10 ANNs were built. Each of them was trained by using the test pattern set corresponding to the circuit. By feeding counter values to a resultant ANN, a test pattern set was generated. Characteristics of ANNs on the accuracy are displayed in Table 4. This table also contains averaged computational times required in training ANNs in the desktop computer. This table displays, when focusing on the row 'c7552,' the deterministic test pattern set was completely regenerated for all times by using trained ANNs. It is omitted to display that fault coverages were almost 100%. Here, it should be noted that the ANN circuit will require many transistors whereas the LFSR requires few ones. It is one of future tasks to compare the LFSR with the ANN circuit in terms of required resources by using the SPICE simulator.

4.4 Results on disturbed ANNs

Each ANN evaluated in Sect. 4.3 was disturbed as described in Sect. 3.2. The accuracies of regenerated test patterns are displayed in Table 5. This table indicates that even when an ANN achieved a high level of accuracy in software, its analog

Table 4. Accuracies of noise-free ANNs and averaged computational times.

Circuit	Min.[%]	Avg.[%]	Comp. time [sec]
c7552	100	100	466.73
s38584	99.07	99.77	3470.0

Table 5. Accuracies of disturbed ANNs [%]

Circuit	Min.	Avg.	Max.
c7552	80.43	83.56	86.31
s38584	68.24	70.43	71.83

circuit may not achieve that accuracy due to the inaccuracy. In addition, Table 5 indicates that the degradation becomes more serious if the CUT is larger, since the accuracies on s38584, which is larger than c7552, are smaller than c7552.

We picked up ANNs that were minimum or maximum on the accuracy, and thus four ANNs were selected. By feeding counter values to those disturbed ANNs, a test pattern set was generated for each ANN. The fault coverages of those test pattern sets are displayed in Table 6. Let us focus on the cell of row 'c7552' and column 'Min. accuracy' in Table 6. The figure '91.07' displays that the test pattern set generated by the ANN having the lowest accuracy achieved 91.07% on fault coverage. This table displays with regard to c7552 that the fault coverage seems to positively correlate to the accuracy, and it is possible to generate a test pattern set that is more effective than the LFSR. However, the difference between most effective results ($1.16 = 94.86 - 93.70$) is slight. On the other hand, Table 6 displays with regard to s38584 that both of two generated test pattern sets were more effective than the most effective test pattern set by the LFSR, and the difference seems unneglectable ($2.82 = 94.41 - 91.32$). Besides, Table 6 displays that the fault coverage seems to negatively correlate to the accuracy; that is, a higher fault coverage was obtained by using an ANN having a lower accuracy. This result discourages us that higher fault coverages may be impossible even when ANNs are sufficiently trained. On the other hand, the result encourages us that relatively higher fault coverages may be obtainable even when ANNs are not sufficiently trained; in other words, an analog circuit that represents an ANN may generate effective test patterns regardless of the inaccuracy.

The treatment of the inaccuracy in this paper does not distinguish the productional noise and the operational noise. The productional noise means the analog circuit of an ANN cannot function as same as being in software. The operational noise means signal values in lines of an analog circuit are disturbed because of various causes, and thus output values of the circuit are also disturbed. The consideration on this distinction is left as a future task.

5. Conclusions

In this paper, we display a simple approach to regenerate deterministic test pattern sets by using ANNs. This approach is based on the expectation that the ANN can be implemented as an analog circuit and assumes that appropriate test pattern sets are prepared. That approach succeeded in achieving higher fault coverages than test pattern sets by LFSRs when it is possible to perfectly implement ANNs as circuits. In addition, the inaccuracy was investigated by disturbing weights of ANNs. 5% relative noise heavily degraded the accuracy of regenerating test pattern sets. However, fault coverages of regenerated test pattern sets were not so degraded. Moreover, such an interesting result was observed that an ANN having a lower accuracy had generated a test pattern set having a higher fault coverage.

Future tasks include to evaluate the approach on larger circuits, to treat the inaccuracy in a more realistic manner, to investigate the relation between the accuracy and the fault coverage, to consider other fault models such as transition faults or delay faults, and to consider the feasibility of implementing the ANN as an analog circuit by using the SPICE simulator.

Table 6. Fault coverages by disturbed ANNs [%]

Circuit	Min. accuracy	Max. accuracy
c7552	91.07	94.86
s38584	94.41	94.14

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 19K11877.

References

[1] N. K. Jha and S. Gupta, "Testing of Digital Systems," Cambridge University Press, June 2012.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436-444, 2015. https://doi.org/10.1038/nature14539

[3] T. Inamoto and Y. Higami, "Application of Convolutional Neural Networks to Regenerate Deterministic Test Patterns for BIST," Proc. of the 34th Int. Tech. Conf. on Circuits/Systems Computers and Communications (ITC-CSCC 2019), pp.523-524, June 23-26, 2019.

[4] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," Proc. of the IEEE Int'l Symposium Circuits and Systems (ISCAS '85), pp.677-692, 1985.

[5] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," Proc. of the International Symposium of Circuits and Systems (ISCAS '89), pp.1239-1934, 1989.

[6] L. Gatet, H. Tap-Beteille and F. Bony, "Comparison Between Analog and Digital Neural Network Implementations for Range-Finding Applications," in IEEE Transactions on Neural Networks, vol. 20, no. 3, pp. 460-470, March 2009, doi: 10.1109/TNN.2008.2009120.

[7] D. Maliuk, H. Stratigopoulos, H. Huang and Y. Makris, "Analog neural network design for RF built-in self-test," 2010 IEEE International Test Conference, Austin, TX, 2010, pp. 1-10, doi: 10.1109/TEST.2010.5699272.

[8] L.W. Nagel and D.O. Pederson, "Simulation Program with Integrated Circuit Emphasis (SPICE)," Technical Report, EECS Department, University of California, Berkeley, 1973.