Turkish Journal of Electrical Engineering and Computer Sciences

Volume 26 | Number 6 Article 37

1-1-2018

# Horizontal diversity in test generation for high fault coverage

ARBAB ALAMGIR

ABU KHARI BIN A'AIN

NORLINA PARAMAN

USMAN ULLAH SHEIKH

IAN GROUT

Follow this and additional works at: https://journals.tubitak.gov.tr/elektrik

Part of the Computer Engineering Commons, Computer Sciences Commons, and the Electrical and Computer Engineering Commons

## Recommended Citation

ALAMGIR, ARBAB; A'AIN, ABU KHARI BIN; PARAMAN, NORLINA; SHEIKH, USMAN ULLAH; and GROUT, IAN (2018) "Horizontal diversity in test generation for high fault coverage," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 26: No. 6, Article 37. https://doi.org/10.3906/elk-1805-212
Available at: https://journals.tubitak.gov.tr/elektrik/vol26/iss6/37

This Article is brought to you for free and open access by TÜBİTAK Academic Journals. It has been accepted for inclusion in Turkish Journal of Electrical Engineering and Computer Sciences by an authorized editor of TÜBİTAK Academic Journals. For more information, please contact academic.publications@tubitak.gov.tr.

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 3258 – 3273
© TÜBİTAK
doi:10.3906/elk-1805-212

# Horizontal diversity in test generation for high fault coverage

**Arbab ALAMGIR**[1*], **Abu Khari Bin A'AIN**[1,2], **Norlina PARAMAN**[1], **Usman Ullah SHEIKH**[1], **Ian GROUT**[3]

[1]School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia
[2]Institute of Integrated Engineering, Universiti Tun Hussein Onn, Batu Pahat, Malaysia
[3]Department of Electronic and Computer Engineering, Faculty of Science and Engineering, University of Limerick, Limerick, Ireland

**Received:** 31.05.2018 • **Accepted/Published Online:** 02.08.2018 • **Final Version:** 29.11.2018

**Abstract:** Determination of the most appropriate test set is critical for high fault coverage in testing of digital integrated circuits. Among black-box approaches, random testing is popular due to its simplicity and cost effectiveness. An extension to random testing is antirandom that improves fault detection by maximizing the distance of every subsequent test pattern from the set of previously applied test patterns. Antirandom testing uses total Hamming distance and total cartesian distance as distance metrics to maximize diversity in the testing sequence. However, the algorithm for the antirandom test set generation has two major issues. Firstly, there is no selection criteria defined when more than one test pattern candidates have the same maximum total Hamming distance and total cartesian distance. Secondly, determination of total Hamming distance and total Cartesian distance is computational intensive as it is a summation of individual Hamming distances and cartesian distances with all the previously selected test patterns. In this paper, two-dimensional Hamming distance is proposed to address the first issue. A novel concept of horizontal Hamming distance is introduced, which acts as a third criterion for test pattern selection. Fault simulations on ISCAS'85 and ISCAS'89 benchmark circuits have shown that employing horizontal Hamming distance improves the effectiveness of pure antirandom in terms of fault coverage. Additionally, an alternative method for total Hamming distance calculations is proposed to reduce the computational intensity. The proposed method avoids summation of individual Hamming distances by keeping track of number of 0s and 1s applied at each inputs. As a result, up to 90% of the computations are reduced.

**Key words:** Antirandom, test pattern generation, computations reduction, horizontal Hamming distance, vertical Hamming distance

## 1. Introduction

The race of innovation and technology development has shifted the trends from system on board to system on chip and system in package. Embedding millions of logical operations on a single platform with efficient utilization of resources results in extremely complex integrated circuits. On top of that, testing and verification is an essential step in formulation of VLSI realization process that increases production costs up to 40% [1–3]. Spurring from D-algorithm, FAN and PODEM, test generation has evolved over the past 50 years hitting the boundaries of quantum search algorithms and utilizing probabilistic correlations among primary inputs [4–7]. Speedy testing even throughout the production cycle is not sufficient to maintain modern reliability standards [8]. Therefore, high performance embedded systems are equipped with highly reliable built-in self-test (BIST) for

*Correspondence: arbab.alamgir@hotmail.com



This work is licensed under a Creative Commons Attribution 4.0 International License.

an on-chip testing during normal operations [9–14]. Comprising of test pattern generator and output response analyzer within the system, BIST automatically generates test patterns and compares the fault-free responses [10–13, 15]. However, the process is largely dependent on the quality of test patterns affecting the test length leading to delayed device operations.

BIST uses pseudorandom test pattern generation (PRTG) through linear feedback shift register (LFSR) because of its simplicity and cost effectiveness [3, 9, 11–13, 16–18]. PRTG outperforms other black-box test pattern generation approaches with its ability to generate large number of random test patterns irrespective of the structural implementation of circuit under test (CUT) [19]. Whereas, PRTG ignores the information of previously executed set of test patterns while generating subsequent test patterns. Lack of this information may generate a number of subsequent test patterns targeting the faults that have already been detected. This redundancy in test pattern generation increases the test length without any effect on fault coverage, therefore, long test time is consumed to achieve satisfactory levels of fault coverage.

Researchers have proposed several methods to overcome the inefficiency in PRTG [3, 8, 9, 14, 15, 18–24]. Weighted random testing uses various set of weights to increase probability of specific inputs [17, 25, 26]. A combinational logic is inserted between test pattern generator and CUT to increase the probability of required inputs. However, excessive time to reach the required input states increases the application time. Furthermore, combinational logic overhead of weighted random testing may be high for larger circuits [26, 27]. Mixed mode BIST uses seeding of deterministic test pattern to increase the performance of PRTG [9, 11, 18, 21, 28–30]. On the fly, reseeding inverts the logic values at the output of LFSR to modify next states targeting deterministic patterns. Experiments show high fault coverage but bit fixing and bit flipping for next state requires high area overhead. Circular self-test scheme connects the primary input and outputs through response analyzer forming a circular feedback [24, 27, 31]. However, fault coverage may be degraded if some required states are not accessible. In spite of dense research around weighted random and mixed mode BIST techniques, these methods are inefficient in terms of test application time and control logic overhead [32].

PRTG is not the only feasible solution for test sequence generation [16, 20, 32]. Various enhancements have been introduced to improve black-box testing using maximization of Hamming distance (HD) and cartesian distance (CD). Total HD (THD) and total CD (TCD) are two distance metrics used to place every subsequent test pattern maximally apart from the set of previously chosen test patterns. Maximization of THD and TCD leads to an optimum test sequence termed as antirandom (AR) or orderly random testing sequence [33–36]. However, the procedure is computational intensive as it requires determination of THD and TCD of all the test pattern candidates for selection of a test pattern with maximum THD and maximum TCD [37, 38]. Random Like Testing Sequence (RLTS) proposes a test pattern generation method that selects a test pattern randomly and maximizes the THD for next test patterns [39]. This type of test pattern generation leads to maximization of THD only and a random selection is carried out instead of TCD maximization. Fast-antirandom (FAR) suggests a test pattern generation technique based on centralizing method and orthogonal selection [40]. FAR centralizes the previously chosen test patterns by taking the average of each input and finds an orthogonal test pattern to the centralized test pattern. FAR is the best applicable in generating a test sequence for an existing random set of test patterns. However, quantity and quality of random seed patterns are still puzzle. Adaptive random testing (ART) randomly selects a number of test patterns from the test pattern candidates and computes only those test patterns for maximum TCD and maximum THD [41]. Restricted random testing and normalized random testing are improvements to ART [42, 43]. Unfortunately, even with high computational overhead it

resulted in production of low quality test patterns. Scalable test pattern generation (STPG) addresses the issue of scalability of testing sequences by a fixed distance approach [44]. STPG avoids TCD calculations by using an adding factor to generate subsequent test patterns. However, there are no guidelines for selection of adding factor in order to increase fault detection. Shiyi Xu proposes a quasi-best distance approach that uses a predetermined distance value to generate subsequent test patterns instead of maximizing TCD [45]. However, according to sphere-packing bound or Hamming bound, a small number of test patterns are available if the predetermined distance is high and vice versa. Scalable antirandom testing (SAT) proposes a bit swapping technique after every $2^n$ cycles of each input [46]. Iterative antirandom (IAR) amplifies the fault detection by proposing a localized distance metric maximal minimal Hamming distance [47]. IAR suggests maximization of maximal minimal Hamming distance for a given length of testing sequence. Following this method controlled random testing generates short test sequences using predetermined lengths of $q = 2, 3$ and 4 test patterns [48]. Recently, optimal controlled random tests (OCRT) with short length $q = 2(log_2 N + 1)$ are proposed for an N-input CUT [49]. OCRTs are repeatedly generated for random test patterns to form a complete test set. All the above approaches show an effort in minimizing the computational overhead either by compromising on distance metrics or restricting test sequence to a localized maximization.

This paper proposes two-dimensional HD to enhance arrangement of test sequence. Moreover, an alternative THD determination procedure is proposed that reduces computational complexity up to 90%. The proposed two-dimensional test pattern generation algorithm is implemented using high level programming and test sequences are subjected to combinational (ISCAS'85) [50] as well as sequential (ISCAS'89) [51] benchmark circuits. Stuck-at fault coverage is compared between the proposed and previous approaches to test its effectiveness. It is observed that the proposed two-dimensional diversity enhancement in testing sequence achieves high fault coverage.

The remainder of this paper is organized as follows: Section 2 provides definitions of basic terms in accordance with previous literature. These definitions are important for understanding of AR concept. In Section 3, test pattern generation of 4-bit CUT is carried out as an example to highlight the issues with AR algorithm. This section brings up two major issues in AR: Ambiguity in selection procedure and computations complexity in THD calculations. Section 4 proposes two-dimensional HD concept to overcome the ambiguity in test pattern selection. Section 5 addresses the issue of computational complexity by introducing an alternative method of THD determination that does not require individual HD calculations. Section 6 presents the comparison of computations between conventional and the proposed method of THD determination. Section 7 comprises of fault simulation on ISCAS'85 and ISCAS'89 benchmark circuits. This section compares the effectiveness of the proposed approach with previous test generation methods. Finally, conclusion of this paper is presented in Section 8.

## 2. Definitions of critical terms

This section introduces definitions of critical terms being used throughout this research paper. The definitions are in accordance with the previous literature [33–40, 44–49, 52]. All definitions are true for a test pattern $t_i = \{t_{i,0}, t_{i,1}, t_{i,2}...t_{i,N-1}\}$, where $t_{i,j} \in \{0, 1\}$.

**Definition 1** *HD between two test patterns $t_1$ and $t_2$ is given as:*

$$HD(t_1, t_2) = \sum_{i=0}^{N-1} (t_{1,i} \oplus t_{2,i}) \tag{1}$$

**Example 1** *Consider* $t_1 = \{0100\}$ *and* $t_2 = \{1101\}$,
$HD(t_1, t_2) = (t_{1,0} \oplus t_{2,0}) + (t_{1,1} \oplus t_{2,1}) + (t_{1,2} \oplus t_{2,2}) + (t_{1,3} \oplus t_{2,3})$
$HD(t_1, t_2) = (0 \oplus 1) + (1 \oplus 1) + (0 \oplus 0) + (0 \oplus 1) = 1 + 0 + 0 + 1 = 2$

**Definition 2** *THD of a test pattern* $t_i$ *with test set* $T = \{t_0, t_1, t_2, t_3...t_{i-1}\}$ *is given as:*

$$THD(T, t_i) = \sum_{k=0}^{i-1} HD(t_k, t_i) \tag{2}$$

**Example 2** *Consider* $T = \{t_0, t_1, t_2\} = \{0001, 0010, 0100\}$ *and* $t_3 = \{0000\}$,
$THD\{T, t_3\} = \sum_{k=0}^{2} HD(t_k, t_3) = HD(t_0, t_3) + HD(t_1, t_3) + HD(t_2, t_3) = 1 + 1 + 1 = 3$

**Definition 3** *CD between two test patterns* $t_1$ *and* $t_2$ *is given as:*

$$CD(t_1, t_2) = \sum_{i=0}^{N-1} \sqrt{(t_{1,i} \oplus t_{2,i})} = \sqrt{HD(t_1, t_2)} \tag{3}$$

**Example 3** *Consider* $t_1 = \{0110\}$ *and* $t_2 = \{1101\}$,
$CD(t_1, t_2) = \sqrt{(t_{1,0} \oplus t_{2,0}) + (t_{1,1} \oplus t_{2,1}) + (t_{1,2} \oplus t_{2,2}) + (t_{1,3} \oplus t_{2,3})}$
$CD(t_1, t_2) = \sqrt{(0 \oplus 1) + (1 \oplus 1) + (1 \oplus 0) + (0 \oplus 1)} = \sqrt{1 + 0 + 1 + 1} = \sqrt{3}$

**Definition 4** *TCD of a test pattern* $t_i$ *with test set* $T = \{t_0, t_1, t_2, t_3...t_{i-1}\}$ *is given as:*

$$TCD(T, t_i) = \sum_{k=0}^{i-1} CD(t_k, t_i) \tag{4}$$

**Example 4** *Consider* $T = \{t_0, t_1, t_2\} = \{0101, 0010, 1010\}$ *and* $t_3 = \{0000\}$,
$TCD\{T, t_3\} = \sum_{k=0}^{2} CD(t_k, t_3) = CD(t_0, t_3) + CD(t_1, t_3) + CD(t_2, t_3) = \sqrt{2} + 1 + \sqrt{2} = 3.8284$

## 3. Problem formulation

The efficiency of random testing is greatly improved by introducing divergent test patterns with every subsequent test pattern selection. AR maximizes THD and TCD between preceding and subsequent test patterns. Maximizing the total distance raises the chance of targeting faults that have not been explored by previously selected test patterns. This implies that, large number of faults can be detected if selection of every subsequent test pattern is carried out such that it has maximum THD and maximum TCD with respect to previously applied set of test patterns. However, there is no selection criteria defined when more than one candidate patterns have maximum THD and maximum TCD.

**Example 5** *Let us take an example of a 4-bit CUT. Table 1 shows the THDs and TCDs of test pattern candidates with respect to previously applied test set* $\{0000, 1111\}$. *All input combinations are considered as candidates for next test pattern selection. The test pattern candidates with maximum THD and maximum TCD are shown in bold. It can be observed that six test pattern candidates have maximum THD and maximum TCD*

*of 4 and 2.82, respectively. AR selection criteria signifies that all the six test pattern candidates are eligible for next selection. This scenario initiates a random selection among eligible test pattern candidates, which may lead to an unoptimized testing sequence. Therefore, another distance criterion guiding towards higher fault coverage is required. This paper proposes a novel distance criterion in Section 4 and proves its effectiveness in Section 7.*

**Table 1**. THD and TCD for candidate test patterns.

| Candidate | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| THD | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| TCD | 2.73 | 2.73 | 2.82 | 2.73 | 2.82 | 2.82 | 2.73 | 2.73 | 2.82 | 2.82 | 2.73 | 2.82 | 2.73 | 2.73 |

Secondly, selection of a test pattern candidate with maximum THD requires THDs of all the candidate patterns. Moreover, THD of a candidate pattern is determined by adding up individual HDs with respect to all previously selected test patterns (Definition 2). Figure 1 gives a pictorial view of HD calculations required for a maximum THD test pattern selection. It is mathematically represented as "$m * n$" HD calculations for "$n$" number of previously applied test patterns and "$m$" number of test pattern candidates.

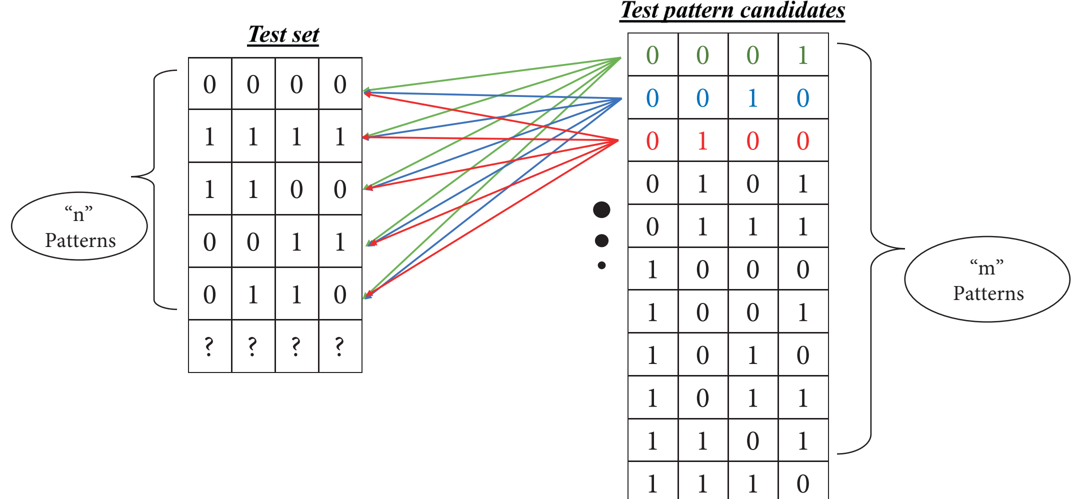


**Figure 1**. HD calculations for an maximum THD selection.

**Example 6** *Let us select a test pattern with maximum THD for 4-bit CUT. Having three test patterns* $\{0101, 1001, 1100\}$ *in a test set, a new selection is required to be carried out. In order to select a test pattern with maximum THD, individual THD of each candidate is required with respect to all of the previously selected test patterns. Each candidate pattern needs 3 HD calculations and a total number of 39* ($m = 13$ *and* $n = 3$) *HD calculations are carried out to identify a candidate pattern with maximum THD.*

Moreover, AR test pattern selection signifies choosing of a candidate pattern and putting it in the set of previously selected test patterns. This causes an increase in the number of previously applied test patterns (n) and a decrease in the number of patterns in the candidate set (m). Thus, the number of HD calculations continues to rise until the number of previously applied test patterns (n) equals to the remaining number of test pattern candidates (m). However, a fall in HD calculations is observed when the test set length exceeds

half of possible input combinations. The number of HD calculations required to sort all the input combinations according to THD is mathematically represented as:

$$HDs = \sum_{i=1}^{L} n_i m_{L-i} \quad (5)$$

where "L" represents the total number of input combinations. Hence, sorting of 16 (L $= 2^N$ for $N = 4$) input combinations according to maximum THD requires 680 HD calculations. Moreover, the total number of HD calculations required for sorting 10-bit (L $= 2^{10}$) sequence is 178,956,800. The number of HD calculations shows an exponential increase as the number of inputs for CUT increases. Consequently, AR selection procedure becomes highly computational intensive. The above discussion concludes that there are two issues with AR algorithm, which are:

- No criterion is defined when more than one test patterns have maximum THD and maximum TCD,
- Test pattern selection procedure is highly computational intensive with summation of individual HDs.

There are two basic requirements to improve AR. Firstly, an additional distance based metric should be introduced to guide selection procedure when more than one test patterns have maximum THD and maximum TCD. Secondly, THD calculation procedure needs to be revised such that it does not need individual HD calculation. This paper proposes a two-dimensional (horizontal and vertical) HD to optimize the test pattern selection procedure. Furthermore, an alternative THD calculation procedure is also proposed in Section 5 that determines THD without calculating individual HDs.

## 4. Two dimensional Hamming distance

The previous section explains that the concept of AR is vague when more than one test pattern candidates reach possible maxima of THD and TCD. Table 1 shows that six test pattern candidates have maximum of THD and TCD. This section defines an absolute criteria to compare pseudoexhaustive properties of test pattern candidates having maximum THD and maximum TCD. The objective of this section is to deduce another distance metric based on absolute criteria to guide test pattern selection process. Table 2 shows a test set $T = \{00000, 11111\}$ and the test pattern candidates $t_{c1} = \{01111\}$, $t_{c2} = \{00111\}$ and $t_{c3} = \{10101\}$. The following definitions are presented in order to evaluate the candidates according to their vertical and horizontal absolute criteria.

**Definition 5** *Vertical absolute criteria for* $t_c$ *test pattern candidate (VACT) is the number of* $k$*-bit binary combinations newly generated by* $t_c$ *compared to respective* $k$*-bit binary combinations generated by previously selected patterns in the test set [47, 48]. (Vertical comparison in Table 2)*

**Definition 6** *Horizontal absolute criteria for* $t_c$ *test pattern candidate (HACT) is the number of unique combinations for an arbitrary* $k$ *out of* $N$ *bits generated by the pattern* $t_c$*.*

Here, "$k$" is an integer ranging $0 < k < N$ and it is equal to 3 in case of Table 2. All the 3-bit binary combinations that are newly generated by test pattern candidates compared to respective 3-bit binary combinations generated by $\{00000, 11111\}$ are in bold. Table 2 shows that VACT of both $t_{c2}$ and $t_{c3}$ is 9 which is greater than VACT of $t_{c1}$. Therefore, $t_{c2} = \{00111\}$ and $t_{c3} = \{10101\}$ are better candidates for next

**Table 2**. Test pattern comparison based on VACT and HACT.

| $\{b_0b_1b_2b_3b_4\}$ | $b_0b_1b_2$ | $b_0b_1b_3$ | $b_0b_1b_4$ | $b_0b_2b_3$ | $b_0b_2b_4$ | $b_0b_3b_4$ | $b_1b_2b_3$ | $b_1b_2b_4$ | $b_1b_3b_4$ | $b_2b_3b_4$ | VACT | HACT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| {00000} | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 10 | 1 |
| {11111} | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 10 | 1 |
| $t_{c1} = \{01111\}$ | 011 | 011 | 011 | 011 | 011 | 011 | 111 | 111 | 111 | 111 | 6 | 2 |
| $t_{c2} = \{00111\}$ | 001 | 001 | 001 | 001 | 011 | 011 | 011 | 011 | 011 | 111 | 9 | 3 |
| $t_{c3} = \{10101\}$ | 101 | 100 | 101 | 110 | 111 | 101 | 010 | 011 | 001 | 101 | 9 | 7 |

selection as compared to $t_{c1} = \{01111\}$. In order to choose between $t_{c2}$ and $t_{c3}$, HACT requires a horizontal observation of rows in Table 2. All the unique 3-bit combinations formed by each test pattern candidate are underlined. It is observable that $t_{c3} = \{10101\}$ generates 7 unique combinations as compared to 3 unique combinations by $t_{c2}$. Consequently, $t_{c3} = \{10101\}$ has a greater HACT and it is capable of introducing higher diversity in the test sequence.

Due to high complexity of VACT and HACT, it is not practical to use them for real applications [47, 48]. However, a closer observation to test pattern candidates $t_{c2} = \{00111\}$ and $t_{c3} = \{10101\}$ divulges that there are more bit transitions in $t_{c3} = \{10101\}$ leading to a greater HACT. This ignites the idea of maximizing bit transitions while selecting subsequent test patterns.

Motivated from the above discussion, this research proposes two-dimensional HD calculations. Hereafter, HD and THD (Definitions 1 and 2) defined by AR are referred as vertical Hamming distance (VHD) and vertical total Hamming distance (VTHD), respectively. Moreover, the count of bit transitions in a test pattern is termed as horizontal total Hamming distance (HTHD). HTHD is proposed as a third distance criterion to select between test pattern candidates. The following definition is true for a test pattern $t_i = \{t_{i,0}, t_{i,1}, t_{i,2}...t_{i,N-1}\}$, where $t_{i,j} \in \{0, 1\}$.

**Definition 7** *HTHD of a test pattern $t_i$ is given as:*

$$HTHD(t_i) = \sum_{l=0}^{N-2} (t_{i,l} \oplus t_{i,l+1}) \quad (6)$$

**Example 7** *Consider $t_1 = \{0010\}$,*
$HTHD\{t_1\} = (t_{1,0} \oplus t_{1,1}) + (t_{1,1} \oplus t_{1,2}) + (t_{1,2} \oplus t_{1,3})$
$HTHD\{t_1\} = (0 \oplus 0) + (0 \oplus 1) + (1 \oplus 0) = 0 + 1 + 1 = 2$

Having introduced the definition 7, let us again consider example 5. Table 1 shows all candidates for the previously selected test set $T = \{0000, 1111\}$. As maximum THD and maximum TCD measures are the same for six test pattern candidates, an additional distance criteria named HTHD is introduced in this section. Table 3 shows HTHD measure of all the test pattern candidates using definition 7. It can be seen that only two test pattern candidates $\{0101, 1010\}$ have the highest transition count of 3. Without any loss of generality any one of these two test pattern candidates can be chosen as subsequent test pattern because one will follow the other. Consequently, maximizing HTHD enables selection procedure to decide among test pattern candidates having equal maximum VTHD and maximum TCD. As the proposed approach is an enhancement to pure AR [34], this paper terms it as AR with HTHD (ARHTHD). Moreover, Section 7 shows that HTHD, as the third criterion, improves fault coverage as compared to pure AR.

**Table 3**. Example 5 with VTHD and HTHD.

| Candidate | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VTHD | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| TCD | 2.73 | 2.73 | 2.82 | 2.73 | 2.82 | 2.82 | 2.73 | 2.73 | 2.82 | 2.82 | 2.73 | 2.82 | 2.73 | 2.73 |
| HTHD | 1 | 2 | 1 | 2 | 3 | 2 | 1 | 1 | 2 | 3 | 2 | 1 | 2 | 1 |

## 5. Avoiding individual VHD for VTHD

Calculating VTHD by summing individual VHDs with respect to previously selected test patterns results in a computation intensive procedure. Section 3 shows that AR selection procedure becomes more complex in CUTs with large number of inputs. This section proposes an alternative method to calculate VTHD that avoids the need to calculate individual VHDs. The proposed procedure counts the total number of 0s and 1s applied on each input by the previous test patterns. The proposed calculations for VTHD are mathematically represented as follows:

**Definition 8** *Proposed method for VTHD calculations:*

$$VTHD(t_i) = \sum_{j=0}^{N-1} [t_{i,j} \oplus 0] * Z_N[j] + [t_{i,j} \oplus 1] * O_N[j], \tag{7}$$

where "$t_i$" represents the test pattern for which VTHD is to be calculated. "N" is the number of inputs of CUT. $Z_N$ and $O_N$ are row vectors of length equal to N representing the total number of 0s and 1s applied previously at a particular CUT input.

**Example 8** *Consider* $t_{15} = \{0111\}$ *test pattern candidate with list of previously applied test patterns is shown in Table 4. The first step is to form* $Z_N$ *and* $O_N$ *vectors. There are seven 0s applied previously on input* $N_0$*. Therefore, the first element of* $Z_N$ *is 7. Similarly, all the columns of* $Z_N$ *and* $O_N$ *are filled. Hence,* $Z_N = [7, 8, 8, 8]$ *and* $O_N = [8, 7, 7, 7]$*. Using definition 8,*

*for j=0;* $[t_{15,0} \oplus 0] * Z_N[0] + [t_{15,0} \oplus 1] * O_N[0] = [0 \oplus 0] * 7 + [0 \oplus 1] * 8 = 8$

*for j=1;* $[t_{15,1} \oplus 0] * Z_N[1] + [t_{15,1} \oplus 1] * O_N[1] = [1 \oplus 0] * 8 + [1 \oplus 1] * 7 = 8$

*for j=2;* $[t_{15,2} \oplus 0] * Z_N[2] + [t_{15,2} \oplus 1] * O_N[2] = [1 \oplus 0] * 8 + [1 \oplus 1] * 7 = 8$

*for j=3;* $[t_{15,3} \oplus 0] * Z_N[3] + [t_{15,3} \oplus 1] * O_N[3] = [1 \oplus 0] * 8 + [1 \oplus 1] * 7 = 8$

$VTHD(t_{15}) = 8 + 8 + 8 + 8 = 32$

Example 8 shows that there is no need to calculate individual VHDs for VTHD with the proposed method. Instead, each test pattern candidate requires only "N" repetitions of definition 8. As a result, "$m * n$" VHD calculations required to determine maximum VTHD are reduced to only "$m * N$". The elimination of VHD calculations causes a high decrease in computations for VTHD calculations. Moreover, the number of computations are decreased successively because of the reduction in the number of test pattern candidates with every selection of test pattern. The next section shows a comparison of VHD calculations between the proposed approach and the conventional method.

## 6. Computations comparison

In order to prove the effectiveness of the proposed method for VTHD determination, MATLAB programming is used to calculate computations required for sorting all input combinations according to maximum VTHD selections. Using the proposed method, VTHD of each test pattern candidate is calculated directly without any prior VHD calculations. Only "N" repetitions of definition 8 are carried out to calculate VTHD. Furthermore, an observation to the definition 8 shows that it can be implemented using only one 2-input multiplexer. Setting $t_{i,j}$ as a control input with $Z_N$ and $O_N$ at inputs allows $Z_N$ or $O_N$ to appear at multiplexer output depending on

**Table 4**. List of previously applied test patterns in example 8.

| Test | $N_0$ | $N_1$ | $N_2$ | $N_3$ |
|---|---|---|---|---|
| $t_0$ | 0 | 0 | 0 | 0 |
| $t_1$ | 1 | 1 | 1 | 1 |
| $t_2$ | 0 | 0 | 1 | 1 |
| $t_3$ | 1 | 1 | 0 | 0 |
| $t_4$ | 0 | 1 | 0 | 1 |
| $t_5$ | 1 | 0 | 1 | 0 |
| $t_6$ | 0 | 1 | 1 | 0 |
| $t_7$ | 1 | 0 | 0 | 1 |
| $t_8$ | 0 | 0 | 0 | 1 |
| $t_9$ | 1 | 1 | 1 | 0 |
| $t_{10}$ | 0 | 0 | 1 | 0 |
| $t_{11}$ | 1 | 1 | 0 | 1 |
| $t_{12}$ | 0 | 1 | 0 | 0 |
| $t_{13}$ | 1 | 0 | 1 | 1 |
| $t_{14}$ | 1 | 0 | 0 | 0 |

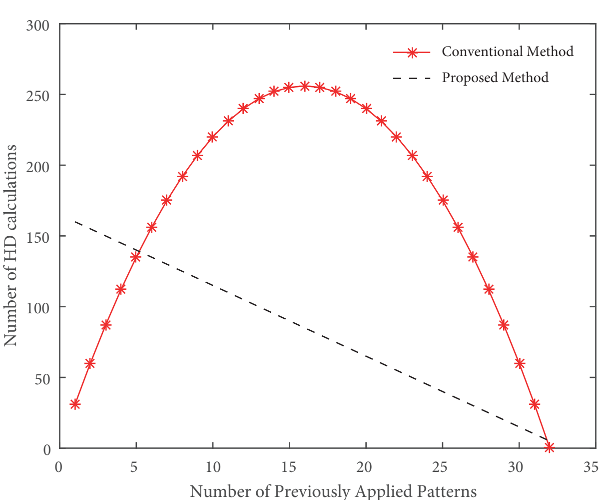


**Figure 2**. VHD calculations for 5-bit CUT conventional vs proposed method.

value of $t_{i,j}$. On the other hand, conventional method (Definition 2) requires large number of XORs depending on the inputs of CUT.

Section 5 shows that every new test pattern selection requires "$m * N$" repetitions of definition 8 for N-input CUT. Furthermore, N-input CUT translates to $2^N$ possible input combinations that either exist in set of previously applied test patterns (n) or in candidate list of test patterns (m). Mathematically, $2^N = m + n$ or $m = (2^N - n)$. Therefore, the number of calculations required to select a test pattern are "$(m*N) = (2^N - n)*N$" with "n" number of previously selected test patterns. Figure 2 shows a comparison of VHD computations between the proposed and conventional methods for the sorting of 5-input combinations. The red curve in the graph shows the computations of conventional method that sum up to 5456 VHD calculations, whereas the total number of repetitions of definition 8 (proposed) are only 2640 with a reduction of 51.613%.

**Table 5**. List of previously applied test patterns in example 8.

| Inputs of CUT | Conventional method [34, 47] | Proposed method | Percentage reduction |
|---|---|---|---|
| 5 | 5,456 | 2,640 | 51.613% |
| 8 | 2,796,160 | 263,168 | 90.588% |
| 10 | 178,956,800 | 5,248,000 | 97.067% |
| 12 | 11,453,245,440 | 100,687,872 | 99.121% |
| 15 | 5,864,062,009,344 | 8,053,309,440 | 99.863% |

Table 5 shows a comparison of VHD computations between the proposed and conventional method. It can be observed that the computations using the proposed method are always less than the computations through conventional method. The reduction of computations is due to the avoiding of individual VHD calculations with all the previously selected test patterns. The last column in the Table 5 shows that the percentage reduction in VHD calculations increases with the increase in number of inputs of CUT. Additionally, the reduction of computations with the proposed procedure has no influence on the fault coverage because the resulted VTHD values are same as those calculated by the conventional method.

## 7. Fault simulation and fault coverage comparison

Finally, this section compares the fault coverage of several random testing strategies to show the effectiveness of the proposed test pattern generation. High level MATLAB programming is used to generate test patterns for all approaches. The generated test patterns are used to test stuck-at faults in combinational (ISCAS'85) and sequential (ISCAS'89) benchmark circuits. ATLANTA is used as a fault simulator to compute fault coverage on benchmark circuits by different test pattern generation strategies. Percentage of stuck-at fault is compared among the testing strategies to observe the effectiveness of each algorithm.

Firstly, a comparison between pure AR and ARHTHD (proposed) is carried out because proposed method is an enhancement of pure AR. Figures 3 and 4 show the comparison of fault coverage between pure AR and ARHTHD. Figure 3 shows fault coverage on c5315 benchmark circuit with 178 primary inputs. Dashed curve represents ARHTHD, whereas plain curve represents the fault coverage with pure AR. It can be seen that fault coverage with 50 test patterns of ARHTHD and pure AR is 87.28% and 65.45%, respectively. Similarly, c2670 benchmark circuit has 233 primary inputs and a comparison of fault coverage is shown in Figure 4. The proposed algorithm shows higher fault coverage as compared to pure AR throughout the test application. After 50 test patterns ARHTHD exposes 82.73% of the faults, whereas pure AR exposes only 78.06% of the faults.

ISCAS'89 benchmark circuits are also tested to observe the effectiveness of the proposed approach on sequential circuits. In design for testability, sequential circuit testing is a challenging task addressed by scan chain implementation. Scan testing is carried out by considering flip flops as pseudoinputs to the combinational block. Scan files of ISCAS'89 are used to test the performance of proposed approach as compared to the previous approaches. Figures 5 and 6 show fault coverage comparison on s35932 and s38417 benchmark circuits, respectively. Dashed curve represents fault coverage with ARHTHD test sequence and plain curve represents fault coverage by pure AR. Figure 5 shows that ARHTHD is able to achieve higher fault coverage as compared to pure AR on s35932 benchmark circuit. Moreover, an increase in difference of fault coverage is observed as test application gets mature. After 100 test patterns, pure AR is able to discover only 46.33% of the faults,

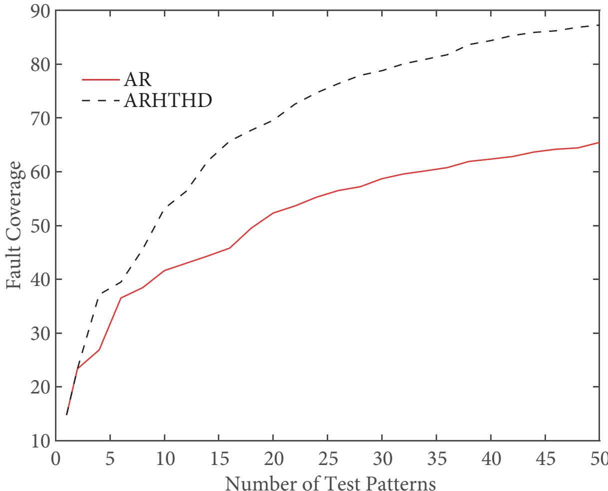


**Figure 3**. Fault coverage of c5315 benchmark circuit.

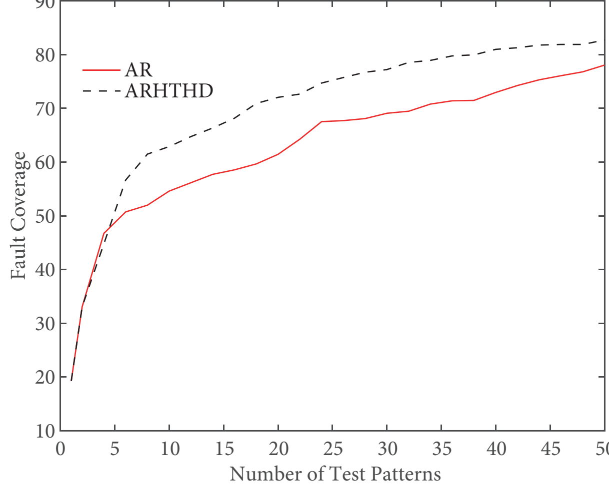


**Figure 4**. Fault coverage of c2670 benchmark circuit.

whereas ARHTHD exposes 51.01% of the faults. Similar comparison on s38417 benchmark circuit in Figure 6 shows that pure AR is able to discover only 71.48% stuck-at faults whereas proposed method with global diversity is able to discover 75.53% of faults. Clearly globalizing the distance aspect has effect on fault coverage of sequential circuits.

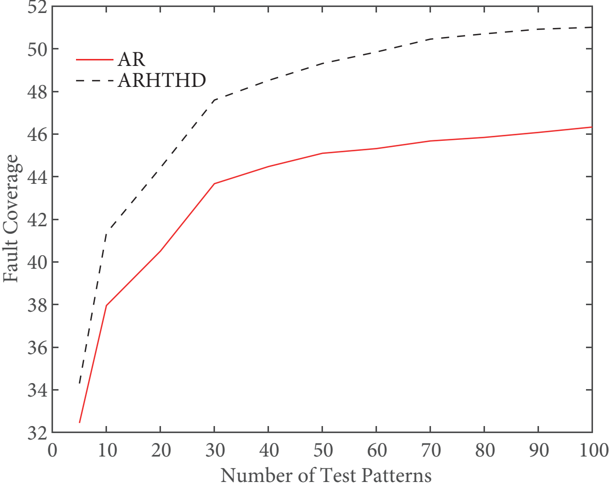


**Figure 5**. Fault coverage of s35932 benchmark circuit.

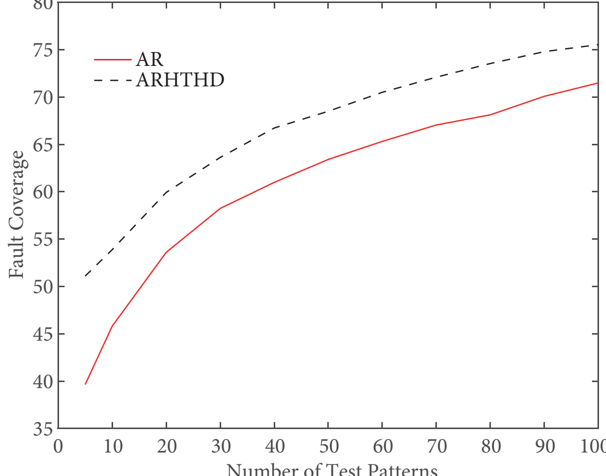


**Figure 6**. Fault coverage of s38417 benchmark circuit.

Furthermore, Table 6 lists ISCAS'85 and ISCAS'89 benchmark circuits with the number of inputs, outputs, gates and possible faults. All the circuits are tested using test sequences from pure AR and ARHTHD. The last two columns of Table 6 show stuck-at fault coverage by each algorithm. It can be observed that pure AR is able to detect less faults as compared to ARHTHD having equal length of testing sequence. This is due to random selection by pure AR among test pattern candidates having maximum VTHD and maximum TCD. However, ARHTHD addresses this scenario as an opportunity to enhance diversity in testing sequence. The proposed approach uses HTHD maximization as a third criteria when more than two test patterns have maximum VTHD and maximum TCD.

**Table 6**. Simulation results for ISCAS'89 and ISCAS'89 benchmark circuits.

| Benchmark circuit | Number of inputs | Number of outputs | Number of gates | Number of faults | Pure AR [33, 34] | Proposed ARHTHD |
|---|---|---|---|---|---|---|
| c1908 | 33 | 25 | 880 | 1879 | 67.53% | 72.11% |
| c432 | 36 | 7 | 160 | 524 | 85.68% | 90.64% |
| c499 | 41 | 32 | 202 | 758 | 88.65% | 90.50% |
| c1355 | 41 | 32 | 546 | 1574 | 83.48% | 85.19% |
| c3540 | 50 | 22 | 1669 | 3428 | 62.48% | 66.59% |
| c880 | 60 | 26 | 383 | 942 | 84.50% | 87.68% |
| c5315 | 178 | 123 | 2307 | 5350 | 64.72% | 86.31% |
| c7552 | 207 | 108 | 3512 | 7550 | 79.62% | 81.93% |
| c2670 | 233 | 140 | 1193 | 2747 | 74.73% | 79.21% |
| s344 | 24 | 26 | 101 | 342 | 99.12% | 99.70% |
| s420 | 34 | 17 | 140 | 455 | 70.32% | 71.64% |
| s510 | 25 | 13 | 179 | 564 | 93.26% | 94.32% |
| s820 | 23 | 24 | 256 | 850 | 59.17% | 61.41% |
| s832 | 23 | 24 | 262 | 870 | 58.16% | 60.11% |
| s953 | 45 | 52 | 311 | 1079 | 61.72% | 66.35% |
| s1238 | 32 | 32 | 428 | 1355 | 66.78% | 67.52% |
| s1423 | 91 | 79 | 490 | 1515 | 88.79% | 92.41% |
| s5378 | 214 | 213 | 1004 | 4551 | 76.20% | 78.92% |
| s35932 | 1763 | 2048 | 12204 | 39094 | 46.33% | 51.01% |
| s38417 | 1664 | 1742 | 8709 | 31180 | 71.48% | 75.53% |
| s38584 | 1464 | 1730 | 11448 | 36303 | 78.79% | 79.70% |

Table 7 summarizes the fault coverage comparison of proposed method with previous random testing methods on ISCAS'85 and ISCAS'89 benchmark circuits. In order to have a comparison on uniform ground, each test generation algorithm generates 50 test patterns sparking with all ones' seed pattern (first test pattern). Moreover, pseudorandom generation is carried out for testing strategies requiring random patterns. First column of Table 7 gives the name of benchmark circuit and first row gives the name of test generation method. Following rows report fault coverage on each benchmark circuit. Large input combinational and sequential benchmark circuits are subjected to all testing methods. It can be observed that proposed method with HTHD enhancement in AR is able to provide highest fault coverage compared to all other testing strategies. The closest fault coverage is observed by OCRT that generates $2(log_2N+1)$ series of distant patterns for every random selection. OCRT provides significant improvement in computational complexity but with a large compromise on test sequence diversity. On the other hand, ARHTHD enhances the distance aspect by introducing two-dimensional HD that helps targeting different faults with every subsequent test pattern.

## 8. Conclusion

This research expands the ideology of diversity in random testing sequence by proposing HTHD as the third criteria for test pattern selection. Moreover, an alternate method for VTHD determination is proposed, which

**Table 7**. Fault coverage comparison with previous methods

| Circuit Name | PRTG [19] | RLTS [39] | FAR [40] | ART [41] | STPG [44] | AR [34] | SAT [46] | IAR [47] | OCRT [49] | ARHTHD (proposed) |
|---|---|---|---|---|---|---|---|---|---|---|
| c5315 | 23.75% | 43.48% | 43.62% | 23.75% | 63.71% | 65.45% | 23.81% | 56.77% | 84.46% | 87.28% |
| c7552 | 40.51% | 49.47% | 49.08% | 40.05% | 53.90% | 81.04% | 40.25% | 58.08% | 83.12% | 83.39% |
| c2670 | 33.19% | 45.51% | 46.65% | 33.19% | 54.75% | 78.06% | 33.15% | 56.08% | 80.76% | 82.73% |
| s510 | 85.46% | 84.04% | 85.10% | 85.10% | 31.38% | 93.26% | 93.97% | 86.88% | 88.47% | 94.32% |
| s641 | 58.31% | 57.88% | 57.01% | 56.80% | 69.54% | 79.48% | 57.23% | 72.57% | 78.83% | 81.42% |
| s713 | 57.14% | 58.17% | 56.62% | 54.73% | 68.50% | 77.10% | 56.97% | 70.56% | 76.24% | 78.57% |
| s953 | 55.79% | 45.02% | 58.85% | 55.79% | 29.56% | 61.72% | 63.29% | 40.77% | 59.31% | 66.35% |
| s1423 | 73.59% | 69.17% | 70.69% | 73.00% | 41.06% | 88.97% | 83.33% | 80.99% | 86.53% | 92.40% |
| s5378 | 35.97% | 39.88% | 40.10% | 35.66% | 38.10% | 76.20% | 46.67% | 49.00% | 75.10% | 78.92% |
| s13207 | 23.29% | 42.53% | 42.53% | 23.29% | 43.13% | 66.53% | 40.08% | 50.36% | 65.32% | 66.54% |

reduces computations up to 90%. Fault simulations demonstrate that high fault coverage is achieved by ARHTHD at much faster rate than other random test generation methods. ARHTHD is more effective due to the fact that it maximizes the global diversity with every subsequent test pattern. One possible way to utilize ARHTHD is to replace it with deterministic testing after suitable high coverage is achieved. In this paper, only black-box testing is considered, which is also being used in software testing and testing of HDL descriptions [2, 52]. However, additional research is required to exploit structural information for further increase in fault coverage.

**Acknowledgment**

We would like to thank Virginia Tech University for allowing us to have access to Atlanta which made this research work possible.

## References

[1] Venkatasubramanian M. Failure evasion: Statistically solving the NP complete problem of testing difficult-to-detect faults. PhD, Auburn University, Auburn, Alabama, USA, 2016.

[2] Anand S, Burke EK, Chen TY, Clark J, Cohen MB, Grieskamp W, Harman M, Harrold MJ, Mcminn P, Bertolino A, Li JJ. An orchestrated survey of methodologies for automated software test case generation. J Syst Software 2013; 86: 1978-2001.

[3] Rinitha R, Ponni R. Testing in VLSI: A survey. In: International Conference on Emerging Trends in Engineering, Technology and Science; 24 Feb 2016; Pudukottai, India: IEEE. pp. 1-6.

[4] Venkatasubramanian M, Agrawal VD. Database Search and ATPG–Interdisciplinary Domains and Algorithms. In: 29th International Conference on VLSI Design; 4-8 Jan. 2016; Kolkata, India: IEEE. pp. 38-43.

[5] Venkatasubramanian M, Agrawal VD. Quest for a quantum search algorithm for testing stuck-at faults in digital circuits. In: International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems; 12-14 Oct. 2015; Amherst, MA, USA: IEEE. pp. 128-133.

[6] Venkatasubramanian M, Agrawal VD. Failures guide probabilistic search for a hard-to-find test. In: North Atlantic Test Workshop; 9-11 May, 2016; Providence, RI, USA: IEEE. pp. 18-23.

[7] Venkatasubramanian M, Agrawal VD. A new test vector search algorithm for a single stuck-at fault using probabilistic correlation. In: North Atlantic Test Workshop; 14-16 May, 2014; NY, USA: IEEE. pp. 57-60.

[8] Agrawal VD, Kime CR, Saluja KK. A tutorial on built-in self-test. I. Principles. IEEE Des Test Comput 1993; 10: 73-82.

[9] Shiao CM, Lien WC, Lee KJ. A Test-per-cycle BIST architecture with low area overhead and no storage requirement. In: International Symposium on VLSI Design, Automation and Test; 25-27 April 2016; Hsinchu: IEEE. pp. 1-4.

[10] Jutman A, Aleksejev I, Raik J. Design and Test Technology for Dependable Systems-on-Chip. Hershey, Pennsylvania, USA: IGI Global, 2011.

[11] Joice DN, Saravanan S. Efficient Test Sequence Generator for Area Optimization in LFSR Reseeding. Ind J Sci Technol 2016; 9: 29.

[12] Devika KN, Bhakthavatchalu R. Programmable MISR modules for logic BIST based VLSI testing. In: International Conference on Control, Instrumentation, Communication and Computational Technologies; 16-17 Dec. 2016; Kumaracoil, India: IEEE. pp. 699-703.

[13] Novák O, Jenícek J, Rozkovec M. LFSR Reseeding Based Test Compression Respecting Different Controllability of Decompressor Outputs. In: International Symposium on Design and Diagnostics of Electronic Circuits & Systems; 22-24 April, 2015; Belgrade, Serbia: IEEE. pp. 9-14.

[14] Mrugalski G, Rajski J, Rybak L, Solecki J, Tyszer J. Deterministic built-in self-test based on compressed test patterns stored on chip and their derivatives. United States Patent US 9933485, 2018.

[15] McCluskey EJ. Built-in self-test techniques. IEEE Des Test Comput 1985; 2: 21-28.

[16] Acharya S. Cellular automata pseudorandom sequence generation. MSc, University of Victoria, Victoria, Canada, 2017.

[17] Xiang D, Wen X, Wang LT. Low-Power Scan-Based Built-In Self-Test Based on Weighted Pseudorandom Test Pattern Generation and Reseeding. IEEE T Vlsi Syst, 2017; 25: 942-953.

[18] Lin X, Rajski J. Logic built-in self-test with high test coverage and low switching activity. United States patent US 9568552, 2017.

[19] Wagner KD, Chin CK, McCluskey EJ. Pseudorandom testing. IEEE T Comput 1987; C-36: 332-343.

[20] Rajski J, Tyszer J, Kassab M, Mukherjee N. Embedded deterministic test. IEEE T Comput Aid D 2004; 23: 776-792.

[21] Krishna CV, Jas A, Touba NA. Test vector encoding using partial LFSR reseeding. In: Proceedings International Test Conference 2001; 1-1 Nov. 2001; Baltimore, MD, USA: IEEE. pp. 885-893.

[22] K.Chakrabarty, B.T.Murray, V.Iyengar. Deterministic built-in test pattern generation for high-performance circuits using twisted-ring counters. IEEE T Vlsi Syst 2000; 8: 633-636.

[23] Koneman B. LFSR-coded test patterns for scan designs. In: European Test Conference; 19-22 April 1993; Rotterdam, Netherlands: IEEE. pp. 237-242.

[24] Lien WC, Lee KJ, Hsieh TY, Chakrabarty K. A new LFSR reseeding scheme via internal response feedback. In: Asian Test Symposium; 18-21 Nov. 2013; Yilan County, Taiwan: IEEE. pp. 97-102.

[25] Rani DG, Meenakshi MM, Marina SA. Low hardware overhead implementation of 3-weight pattern generation technique for VLSI testing. In: International Conference on Devices, Circuits and Systems; 6-8 March 2014; Combiatore, India: IEEE. pp. 1-5.

[26] Paschalis A, Voyiatzis I, Gizopoulos D. Accumulator based 3-weight pattern generation. IEEE T Vlsi Syst 2012; 20: 357-361.

[27] Lien WC, Lee KJ, Hsieh TY, Chakrabarty K. Efficient LFSR reseeding based on internal-response feedback. J Electron Test 2014; 30: 673-685.

[28] Liu T, Kuang J, Cai S, You Z. An effective logic BIST scheme based on LFSR-reseeding and TVAC. Int J Electron 2014; 101: 1217-1229.

[29] Lien WC, Lee KJ, Hsieh TY. A test-per-clock LFSR reseeding algorithm for concurrent reduction on test sequence length and test data volume. In: Asian Test Symposium; 19-22 Nov. 2012; Nigata, Japan: IEEE. pp. 278-283.

[30] Eggersglüß S, Krenz-Bååth R, Glowatz A, Hapke F, Drechsler R. A new SAT-based ATPG for generating highly compacted test sets. In: International symposium on Design and Diagnostics of Electronic Circuits & Systems; 18-20 Apr. 2012; Tallinn, Estonia: IEEE. pp. 230-235.

[31] Wen K, Hu Y, Li X. Deterministic circular self test path. Tsinghua Sci Technol 2007; 12: 20-25.

[32] Lien WC, Lee KJ, Hsieh TY, Ang WL. An efficient on-chip test generation scheme based on programmable and multiple twisted-ring counters. IEEE T Comput Aid D 2013; 32: 1254-1264.

[33] Xu S. Orderly random testing for both hardware and software. In: Pacific Rim International Symposium on Dependable Computing; 15-17 Dec. 2008; Taipei: IEEE. pp. 160-167.

[34] Wu SH, Jandhyala S, Malaiya YK, Jayasumana AP. Antirandom testing: a distance-based approach. VLSI Des 2008; 2008: 2.

[35] Wu S, Malaiya YK, Jayasumana AP. Antirandom vs. pseudorandom testing. In: International Conference on Computer Design: VLSI in Computers and Processors; 5-7 Oct. 1998; Austin, Texas: IEEE. pp. 221-223.

[36] Malaiya YK. Antirandom testing: getting the most out of black-box testing. In: International Symposium on Software Reliability Engineering; 24-27 Oct. 1995; Toulose, France: IEEE. pp. 86-95.

[37] Mrozek I, Yarmolik V. Multiple Controlled Random Testing. Fund Inform 2016; 144: 23-43.

[38] Mrozek I, Yarmolik V. Methods of Synthesis of Controlled Random Tests. In: International Conference on Computer Information systems and Industrial management; 14-16 Sep. 2016; Lithuania: Springer. pp. 429-440.

[39] Xu S. Random-like testing of very large scale integration circuit. J Shanghai Univ 1998; 2: 279-283.

[40] Chen T, Bai A, Hajjar A, Andrews AK, Anderson C. Fast anti-random (FAR) test generation to improve the quality of behavioral model verification. J Electron Test 2002; 18: 583-594.

[41] Chen TY, Leung H, Mak IK. Adaptive random testing. In: Annual Asian Computing Science Conference; 8-10 Dec. 2004; Chiang Mai, Thailand: Springer. pp. 320-329.

[42] Chan KP, Chen TY, Towey D. Normalized restricted random testing. In: International Conference on Reliable Software Technologies; 16-20 June 2003; Toulouse, France: Springer. pp. 368-381.

[43] Chan KP, Chen TY, Towey D. Restricted random testing. In: European Confernece on Software Quality; 9-13 June 2002; Helsinki, Finland: Springer. pp. 321-330.

[44] Yiunn DB, A'ain AK, Ghee J. Scalable test pattern generation (STPG). In: IEEE Symposium on Industrial Electronics & Applications; 3-5 Oct. 2010; Penang, Malaysia: IEEE. pp. 433-435.

[45] Xu S, Xu P. A Quasi-best Random Testing. In: Asian Test Symposium; 1-4 Dec. 2010; Shanghai: IEEE. pp. 21-26.

[46] Sahari MS, A'ain AK, Grout IA. Scalable antirandom testing (SAT). International Journal of Innovative Science and Modern Engineering (IJISME) 2015; 3: 33-35.

[47] Mrozek I, Yarmolik VN. Iterative antirandom testing. J Electron Test 2012; 28: 301-315.

[48] Yarmolik SV, Yarmolik VN. Controlled random tests. Automat Rem Contr+ 2012; 73: 1704-1714.

[49] Mrozek I, Yarmolik V. Optimal Controlled Random Tests. In: International Conference on Computer Information systems and industrial management; 16-18 June 2017; Bialystok, Poland: Springer. pp. 27-38.

[50] Bryan D. The ISCAS'85 benchmark circuits and netlist format. North Carolina State University 1985; 25.

[51] Brglez F, Bryan D, Kozminski K. Combinational profiles of sequential benchmark circuits. In: IEEE International Symposium on Circuits and Systems; 8-11 May 1989; Portland, OR, USA: IEEE. pp. 1929-1934.

[52] Mrozek I, Yarmolik V. Antirandom test vectors for BIST in hardware/software systems. Journal Fundamenta Informaticae 2012; 119: 163-185.