J Electron Test (2012) 28:301–315
DOI 10.1007/s10836-011-5272-1

Iterative Antirandom Testing

Ireneusz Mrozek · Vyacheslav N. Yarmolik

Received: 11 August 2011 / Accepted: 9 December 2011 / Published online: 11 January 2012
© The Author(s) 2012. This article is published with open access at Springerlink.com

Abstract Antirandom testing is a variation of pure random testing, which is the process of generating random patterns and applying it to a system under test (both software systems and hardware systems). However, research studies have shown that pure random testing is relatively less effective at fault detection than other testing techniques. Antirandom testing improves the fault-detection capability of random testing by employing the location information of previously executed test cases. In antirandom testing we select test case such that it is as different as possible from all the previous executed test cases. Unfortunately, this method essentially requires enumeration of the input space and computation of each input pattern when used on an arbitrary set of existing test data. This avoids scale-up to large test sets and (or) long input vectors. The objective of this paper is to find a more efficient method of the test generation which does not need any computation. The key idea of proposed approach is an iterative application of the short antirandom tests where the first test vector in each iteration is generated randomly. Moreover, we propose a new metric the Maximal Minimal Hamming Distance (MMHD) which allows us to define an optimal antirandom test with restricted number of patterns. Experimental results are given to evaluate the performance of the new approach.

Keywords Antirandom test · Random tests · Fault coverage · Maximal minimal hamming distance

Responsible Editor: V. D. Agrawal

This paper was supported by grant S/WI/5/08 from Faculty of Computer Science at Bialystok University of Technology, Poland

I. Mrozek (✉) · V. N. Yarmolik
Faculty of Computer Science, Bialystok University of Technology, Bialystok Poland
e-mail: i.mrozek@pb.edu.pl

V. N. Yarmolik
e-mail: v.yarmolik@pb.edu.pl

1 Introduction

Efficient test patterns and methods of their generation play still crucial role in both hardware and software testing. In case of systems with limited number of inputs we can use exhaustive testing. Exhaustive testing generally means to verify the functionality of the circuit under test [1]. In case of combinational circuits it needs 2^N different N-bit patterns, where N is the number of inputs. The exponential growth of the test length restricts the concept of the exhaustive testing to circuits with a limited number of inputs [1, 23]. Locally exhaustive [4] or pseudo exhaustive [7] testing is a concept to avoid the restricted number of inputs of the circuit under test. These approaches are the real alternatives to exhaustive testing. They allow for sufficiently reducing the number of the test vectors. It is possible by taking advantage of the fact that often many or all output variables depend only on a small subset of input variables [4, 10, 11]. A set $T(N, k)$ of pseudo exhaustive test patterns in a binary N-space, exhaustively covers all specified k-subspaces if the projection of $T(N, k)$ onto those k-subspaces contains all 2^k distinct binary vectors in each specified k-subspace [23]. The main advantage of pseudo exhaustive testing is the sufficiently lower test complexity $O(T(N, k))$ (the

number of test patterns). It can be bounded by the following inequality $2^k \leq O(T(N,k)) \leq 2^N$. For example, the pseudo exhaustive test $T(6,2) = \{000000, 000011, 011100, 101101, 110110, 111011\}$ contains only six test patterns. Therefore the complexity of this test equals to $O(T(N,k)) = O(T(6,2)) = 6$ and is much lower than the upper bound $2^N = 2^6 = 64$ [10].

Further development and active research of the pseudo exhaustive testing have been done for complex computer systems and software applications within the framework of so called Combination Testing Strategies [6]. This strategy can be applied for a general case when black box testing environment is used. In this case the set of parameters is divided into subsets for which all combinations of input values are generated. To construct all combination the Covering Arrays may be used [9].

As a good approximation of exhaustive and pseudo exhaustive testing the random testing have been widely used [16, 21, 22]. In this case by the definition the each test pattern is selected randomly, regardless of the previous test patterns applied. The advantages of random testing include its low cost of implementation, ability to generate numerous test cases automatically, generation of test cases in the absence of the object specification and apart from these, it brings randomness into the testing process. Moreover random testing typically uses test patterns which are produced by a pseudo-random number generator with a specific seed value so that any system failures can be reproduced [18]. Random testing and its variations have been extensively used and studied for both hardware testing and software testing [15, 18, 20, 27, 28]. Available evidence suggests that random testing may be reasonable choice for obtaining a moderate degree of confidence.

Random testing does not exploit some information that is available in black box testing environment. This information consists of the previous tests applied [14]. Therefore the new approach called Antirandom Testing (AT) was proposed. In this case each test pattern is chosen such that its total distance from all previous patterns is maximum [14, 25, 26]. Distance is a measure of how different two test patterns T_i and T_j are [14]. So, to achieve higher fault coverage, one should choose test patterns that are as different as possible from the patterns previously generated. The assumption is that similar input to a system will expose similar types of faults, and therefore an input set that contains values that are very different from each other has a higher likelihood of exposing faults in a object under test.

For antirandom testing the Hamming Distance (HD) and Cartesian Distance (CD) have been used to measure the differences between two patterns [14, 25]. Therefore the new test pattern is chosen such that maximise these distances. This approach has proved more efficient than random testing [14, 26]. Unfortunately the basic antirandom method essentially requires enumeration of the input space and computation of distances for each potential input pattern [14]. Even for improved version of the method, computations become too expensive for real dimension N of the test patterns [17]. Therefore many modification of antirandom tests have been proposed to simplify the process of generating of antirandom test patterns. As an example Partial Antirandom Testing [18], Adaptive Random Testing [2, 3] Orderly Random testinh [29] or Scalable Test Pattern Generation [33] can be mentioned.

The objective of this paper is to find a more efficient method of the test patterns generation which does not need any computation. A new solution is based on iterative application of antirandom tests with a small number of patterns. As more efficient the new metric the *Maximal Minimal Hamming Distance* (MMHD) between two test patterns T_i and T_j is used instead of Hamming Distance or Cartesian Distance. This allows to construct optimal antirandom test with restricted number of patterns.

In the next Section 2 we will introduce the basic concepts of the antirandom tests. The formal definitions of antirandom tests and main measures are presented there. In this section one can find definitions of two new metrics called absolute criteria for the next test pattern (ACT) and absolute criteria for the set of patterns (ACS) too. In the Section 3 an optimal short antirandom test with $q = 2, 3$ and 4 patterns will be generated. The optimality of the antirandom tests with a small number of patterns will be examined with respect to different metrics. The new metric the Maximal Minimal Hamming Distance (MMHD) will be proposed too. Section 4 presents the analyzes of the short antirandom tests coverage. These analyzes have been done with respect to absolute criteria ACS. The main concepts of iterative antirandom testing are proposed and analyzed in the Section 5. Experimental results to evaluate coverage of the iterative testing procedure are given in the Section 6. Finally, the conclusion of this paper is presented in the Section 7.

2 Antirandom Testing

Antirandom testing technique is a variant of random testing. It was proposed by Malayia [14]. As in the case of random testing it is a black-box strategy. This means that it assumes no information about the internal implementation of the object under test. Antirandom

testing is based on the idea that test cases have to be selected to have maximum *distance* from each other. In this approach, there is using the hypothesis that if two test patterns have only a small distance between them, then the sets of faults encountered by the two are likely to have a number of faults in common. Conversely, if the distance between two test patterns is large, then the set of faults detected by one is likely to contain only a few of the faults detected by the other [14]. As a measure of distance Malaiya proposes to use Hamming distance or Cartesian distance. Therefore the inputs of the object under test are encoded by a binary pattern and each value from the input domain is represented by one or more binary patterns. So, antirandom means that each new test pattern in a test sequence lies as far from all previous test patterns in the sequence as possible. The set of test patterns is generated in such a way that each new test pattern added to the test set is the test case which is the most different from the test cases currently in the test set.

Now we give the formal definitions of the crucial terms relating to the antirandom testing.

Definition 1 Antirandom test $T = \{T_0, T_1, T_2, ..., T_{q-1}\}$ (AT) is a test containing a test pattern T_i, $i \in \{0, 1, 2, ..., q-1\}$, where T_i is chosen such that it satisfies to some criterion with respect to all test patterns T_0, T_1, T_2, ... , T_{i-1} have been obtained before and $T_i = t_{i,N-1}, t_{i,N-2}, ..., t_{i,2}, t_{i,1}, t_{i,0}$ (for binary case $t_{i,j} \in \{0, 1\}$).

Definition 2 The Hamming Distance $\mathrm{HD}(T_i, T_j)$ (HD) between two binary test patterns T_i and T_j is calculated as a weight $\mathrm{w}(T_i \oplus T_j)$ (number of ones) of pattern $T_i \oplus T_j$.

Definition 3 The Cartesian Distance $\mathrm{CD}(T_i, T_j)$ (CD) between two binary patterns T_i and T_j is given by:

$$\begin{aligned}\mathrm{CD}(T_i, T_j) &= \sqrt{(t_{i,0}-t_{j,0})^2+(t_{i,1}-t_{j,1})^2+...+(t_{i,N-1}-t_{j,N-1})^2} \\ &= \sqrt{|t_{i,0}-t_{j,0}|+|t_{i,1}-t_{j,1}|+...+|t_{i,N-1}-t_{j,N-1}|} \\ &= \sqrt{\mathrm{HD}(T_i, T_j)}. \qquad (1)\end{aligned}$$

As an example consider a pair of patterns: $A = (0000)$ and $B = (1010)$. Then $\mathrm{HD}(A, B) = 2$ and $\mathrm{CD}(A, B) = \sqrt{2}$

Definition 4 Total Hamming distance (THD) for any pattern is the sum of its Hamming distances with respect to all previous patterns. We refer to the total distance when Hamming and Cartesian distances are used by $\mathrm{THD}(T_i)$ and $\mathrm{TCD}(T_i)$, respectively.

Definition 5 Maximal Distance Antirandom Test (MDAT) is a test with maximal value of some function F with the distances as arguments.

For example, in [14, 17, 26] HD and CD have been used to construct the functions $\mathrm{F_1(HD)}$ and $\mathrm{F_2(CD)}$. Both functions are used as a fitness functions for consecutive test patterns generations. The next T_i pattern is generated to make the total distance between T_i and each of T_0, T_1, ..., T_{i-1} maximal one. The total distances are calculated as

$$\mathrm{F_1(HD)} = \sum_{j=0}^{i-1} \mathrm{HD}(T_i, T_j), \qquad (2)$$

$$\mathrm{F_2(CD)} = \sum_{j=0}^{i-1} \mathrm{CD}(T_i, T_j). \qquad (3)$$

Definition 6 Maximal Hamming Distance Antirandom Test MDAT(HD) is the MDAT that uses Hamming distance as the distance measure for consecutive test patterns according to the Eq. 2. Maximal Cartesian Distance Antirandom Test MDAT(CD) is the MDAT that uses Cartesian distance as the distance measure for consecutive test patterns according to the Eq. 3 [14].

Definition 7 Absolute Criteria for the next T_i test pattern (ACT) is the maximal number $F_3(T_i) = maxC(T_i(N, k))$ of additional binary combinations (with respect to the test patterns already chosen) for arbitrary k out of N bits generated by the pattern T_i.

Definition 8 Absolute Criteria for the set $T = \{T_0, T_1, T_2, ..., T_{i-1}, T_i\}$ of the test patterns (ACS) is the maximal number $F_4(T) = maxC(T(N, k))$ of binary combinations for arbitrary k out of N bits generated by the set $T_0, T_1, T_2,... ,T_{i-1}, T_i$ including the pattern T_i.

To illustrate the *Absolute Criteria* let us consider the test T which contains two patterns: $T = \{T_0, T_1\} = \{00000, 11111\}$. Assume that two other patterns {01111} and {00111} are the candidates for the third pattern T_2 of the test T and we have to evaluate them in respect of *Absolute Criteria* with $k = 3$. All the details of the evaluation are shown in the Table 1. All the 3-bit patterns generated by {01111} and {00111} which are new

Table 1 Absolute criteria example

$T_i = \{b_0b_1b_2b_3b_4\}$	$b_0b_1b_2$	$b_0b_1b_3$	$b_0b_1b_4$	$b_0b_2b_3$	$b_0b_2b_4$	$b_0b_3b_4$	$b_1b_2b_3$	$b_1b_2b_4$	$b_1b_3b_4$	$b_2b_3b_4$	$F_3(T_i)$	$F_4(T)$
$T_0 = 00000$	000	000	000	000	000	000	000	000	000	000	10	10
$T_1 = 11111$	111	111	111	111	111	111	111	111	111	111	10	20
$T_2 = 01111$	**011**	**011**	**011**	**011**	**011**	**011**	111	111	111	111	6	26
$T_2 = 00111$	**001**	**001**	**001**	**001**	**011**	**011**	**011**	**011**	**011**	111	9	29

compared to 3-bits patterns generated by T_0 and T_1 are in bold. We observe that {00111} allows us to generate nine new 3-bit patterns compared to six new 3-bit patterns generated by {01111}. Therefore $T_2 = \{00111\}$ is better candidate for the third pattern of the test T in respect of *Absolute Criteria*.

It should be noted that due to the high complexity of the ACT and ACS estimation it is not practical to use them for real application with real value of N and q.

To construct of a MDAT a procedure based on exhaustive search is proposed in [14].

1. For each of N input variables, assign an arbitrarily chosen value to obtain the first test pattern.
2. To obtain each new pattern, evaluate the THD (TCD) for each of the remaining combinations with respect to the combinations already chosen and choose one that gives maximal distance. Add it to the set of selected patterns.
3. Repeat Step 2 until all 2^N combinations have been used, or until the desired number of test patterns have been generated.

To illustrate the process of generating MDAT, the generation of a test set which contains four antirandom 3-bit inputs will be considered in an Example 1. Graphically this process is illustrated in Fig. 1 using a cube with each node representing one pattern.

Example 1 MDAT(HD) generating process in case of system with three binary inputs

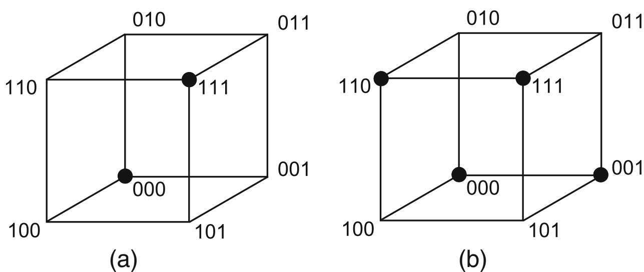

Fig. 1 Construction of 3-bit MDAT(HD) [25]

The complete input domain for a system with three binary inputs is:

0 : 000 4 : 100
1 : 001 5 : 101
2 : 010 6 : 110
3 : 011 7 : 111

Let us start with the first test pattern {0, 0, 0} and add it to the test set. The initial antirandom test set was empty and first pattern could be arbitrarily selected from the input domain and added to the test set. Arbitrarily selection does not result in any loss of generality as the procedure can easily be used for sequences starting with any pattern [14]. The next input to be added to the antirandom test set is the pattern from the domain space which is most different from the current inputs in the test set. In case of MDAT(HD) the next pattern T_1 is obviously {1, 1, 1} with $\mathrm{THD}(T_1) = 3$. It is shown in Fig. 1a, where the input combinations already chosen are marked. Now a symmetrical situation exists. Any pattern chosen would have HD = 1 from one of the past chosen patterns and HD = 2 from the others. Let us assume that value {0,0,1} is picked from the input domain and added to the antirandom test set so the set contains {0,0,0}, {1,1,1}, and {0,0,1}. Which pattern is chosen as the fourth member of the test set depends upon the difference function used. If we use Hamming distance as the difference function then the sum of the Hamming differences between each of the three current members of the antirandom test set and the five other patterns from the input domain are:

	000		111		001	
010	1	+	2	+	2	= 5
011	2	+	1	+	1	= 4
100	1	+	2	+	2	= 5
101	2	+	1	+	1	= 4
110	2	+	1	+	3	= 6

Base upon the above results we can say that {1,1,0} is the most different from the existing members of the antirandom test set and it should be added to the test set. This fact is illustrated in the Fig. 1b too. The selected pattern lies at the opposite corner of the cube to the corner with {0,0,1} pattern.

The Example 1 points out that pure antirandom test patterns generation requires enumeration of the input space and computation of distances for each potential input pattern. Even in case of improved procedure of generating of the antirandom test patterns [14] we need computations which are possible for systems with a relatively small input domain.

Adaptive Random Testing is one of the another testing method that uses a related concept of "distance" to generate test cases. The first algorithm of this class, the Fixed Size Candidate Set ART algorithm (FSCS-ART) was published in [3]. There are many enhancements of this approach too [5, 13, 24, 34]. The idea of FSCS-ART is presented in Algorithm 1. The algorithm of choosing a new test case can be divided into two steps:

- First, a set of k candidates c_i is randomly generated
- Second, one test case from the set of candidates is selected and the other are discarded. Selection is based on the distance between previously executed tests cases T and candidates. For each candidate c_i we find the minimal distance $d_{\min}$ between c_i and previously executed tests T. The candidate c_i with the largest $d_{\min}$ is selected, executed and added to T.

The algorithm of choosing a new test case in FSCS-ART for $k = 4$ is illustrated in Fig. 2 [2]. Previously executed test cases, T_1, T_2, T_3 denoted by dots and randomly generated candidates c_1, c_2, c_3 and c_4 denoted by squares are in the Fig. 2a. Figure 2b shows the process of calculating distances between the candidate c_1 and all previously executed test cases. We must do it for each test candidate c_i. The minimal distances $d_{\min}$ between candidates and test cases previously executed are depicted in Fig. 2c. Then we choose the test case c_i with the largest $d_{\min}$ and treat it as a new test case. In the example c_1 is candidate with the largest $d_{\min}$. It is shown in Fig. 2d.

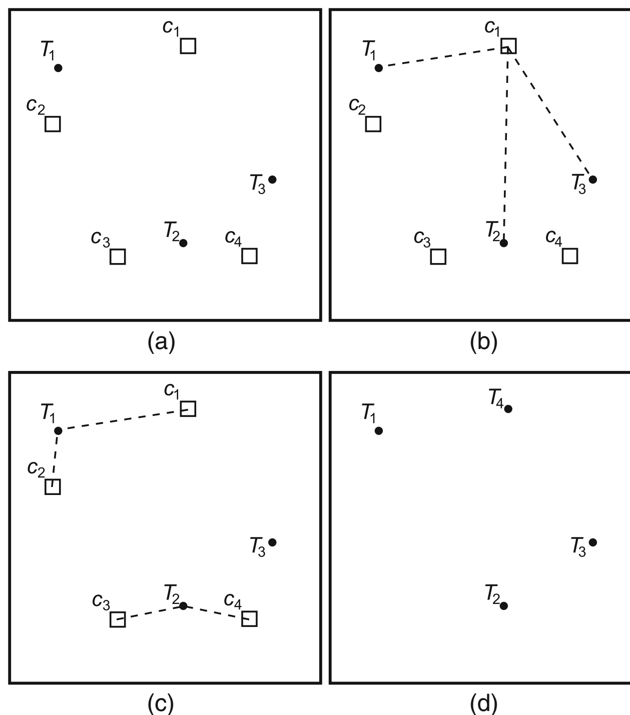

Fig. 2 Choosing a new test case in FSCS-ART

Algorithm 1 FSCS-ART algorithm [2]

$T = \{\}$ /* T is the set of previously executed test cases */
randomly generate an input t
add t to T
while stopping criteria not reached **do**
  $D = 0$
  randomly generate next k candidates $c_1, c_2, ..., c_k$
  for each candidate c_i **do**
    calculate the minimum distance d_i from T
    if $d_i > D$ **then**
      $D = d_i$
      $t = c_i$
    end if
  end for
  add t to T
  test the system using t as a test case
end while

From the above example we can note, that Adaptive Random Testing allows us to skip the enumeration of the whole input space. Unfortunately this method needs still a lot of computations of distances for each potential test pattern.

Another approach to generate antirandom tests was presented in [33]. It is called as Scalable Test Pattern Generation (STPG). Unlike pure antirandom method of the test generations authors in [33] introduced scalable generation method of antirandom tests. The test pattern generation in this algorithm is as follows [33]:

1. Initialize the input value and assign an *adding factor*
2. To obtain a maximum distance between two test patterns, T_0 and T_1, complement the first pattern, T_0, to become the second test pattern, T_1.
3. To obtain third test pattern, T_2, use the assigned *adding factor* and add it to the first test pattern, T_0, to generate third sequence, T_2.
4. Repeat step 2 and step 3 to generate T_3, T_4, etc.

The illustration of this algorithm is given in the Table 2. Let us initialize the input $T_0 = \{000\}$. The best way to obtain the next pattern T_1 with maximum distance between T_0 and T_1, obviously is by complementing $T_0 = \{000\}$. According to Table 2, $T_2 = \{010\}$ is generated by adding an *adding factor* {010} to T_0. By using adding operation we obtain new test pattern so that the complementing process can be continued. The complete test sequences is given in Table 2.

The proposed STPG method is easier to scale up then standard antirandom algorithm. Very important role in this algorithm plays the add factor. Unfortunately the authors of the algorithm don't give instructions on how to determine this factor in order to achieve maximum fault coverage.

Based on random testing the concept of orderly random testing has been proposed in [29]. The first step in direction of ordering the random patterns was *Semi-Maximum Distance Testing Sequences* (SMDTS) in which every test pattern has its own complement version. For example, $T = \{000, 111, 010, 101\}$ is an SMDTS, since {000, 111} and {010, 101} are complement each other. For semi-maximum distance testing new metrics, compare with THD and TCD, have been proposed [30].

Definition 9 *Total Hamming Distance* for the Set of Patterns T (THD) and *Total Cartesian Distance* for the Set of Patterns T (TCD) for any $T = \{T_0, T_1, T_2, \ldots, T_{q-1}\}$ is the sum of its Hamming $HD(T_i, T_j)$ and Cartesian $CD(T_i, T_j)$ distances for all $i \neq j; i, j \in \{0, 1, \ldots, q-1\}$.

$$THD(T) = \sum_{i=1}^{q-1} \sum_{j=0}^{i-1} HD(T_i, T_j),$$

$$TCD(T) = \sum_{i=1}^{q-1} \sum_{j=0}^{i-1} CD(T_i, T_j) \tag{4}$$

Table 2 STPG 3-bit test sequence generation example

T_i	Test pattern	Add	Complement
T_0	000		
T_1	111		T_0^*
T_2	010	$T_0 + 010$	
T_3	101		T_2^*
T_4	100	$T_2 + 010$	
T_5	011		T_4^*
T_6	110	$T_4 + 010$	
T_7	001		T_6^*

The procedure of generation of SMDTS with $2k, k = 1, 2, 3, \ldots$ patterns consists of randomly generated all even number of patterns, i.e. $T_0, T_2, \ldots$ is selected randomly, while each odd number of pattern is obtained by simply bit-wise complementing its previous one. According to this procedure the orderly random test can be obtained. The total distance THD for the set of patterns $T = T_0, T_1, T_2, \ldots, T_{2k-2}, T_{2k-1}$, selected by the procedure, has been proved to be k^2N [30]. For above presented example of SMDTS $T = \{000, 111, 010, 101\}$, $k = 2$ and $N = 3$ the total distance $THD = k^2N = 2^2 3 = 12$, what satisfies to the Eq. 4. Due to the only half of test patterns can reach their maximal distance such a type of distances has been called as Semi-Maximum Distance Testing Sequences.

As the further development of orderly random testing the Total Maximum Distance Test Sequences (TMDTS) have been proposed and analyzed in [29]. According to the definition of TMDTS both THD and TCD must be taking into account for the test construction. Then TMDTS is generated according to the Algorithm 2 [29].

To simplify the step 3 due to their computational complexity in [29] *Pre-Determined Distance Test Sequences* (PDDTS) have been proposed. The procedure for PDDTS generation is different with previous one only in step 3, which now can be formulated as follows.

> To obtain each new i-th pattern $T_0, T_2, T_4, \ldots, T_i, \ldots, T_{2k-2}$, with even number of subscript $i \in \{0, 2, 4, \ldots, 2k-2\}$ from the remaining combinations, chose T_i with $CD(T_i, T_j)$ distances for all $j \in \{0, 2, 4, \ldots, i-2\}$ equals at least any preset value.

Algorithm 2 Construction of TMDTS for $q = 2k$

1. Randomly chosen N bits pattern to be the first pattern termed as T_0
2. To obtained each new i-th pattern $T_1, T_3, T_5, \ldots, T_{2k-1}$ with odd number of subscript, simply bit-wise complementing the previous even pattern should be done, then THD takes the maximum value equals to $i \times N$.
3. To obtain each new pattern with even number of subscript $T_2, T_4, T_6, \ldots, T_{2k-2}$ from the remaining combinations chose one that gives the maximum TCD with respect to the combinations already chosen.
4. Repeat steps 2 and 3 until all q patterns have been chosen.

The main disadvantageous of TMDTS and PDDTS is related to the length of test sequence. This is because the larger the minimal distance $min\mathrm{CD}(T_i, T_j)$ between two patterns T_i and T_j is, the smaller the number of patterns Q could be available, what follows from the sphere-packing bound or Hamming bound [8]. This bound for $min\mathrm{HD}(T_i, T_j) = 2r + 1$ and $min\mathrm{CD}(T_i, T_j) = \sqrt{2r+1}$ can be expressed as the next inequality:

$$Q \le \frac{2^N}{\sum_{l=0}^{r} \binom{N}{l}} \tag{5}$$

For example, in a case of $N = 15$ and $min\mathrm{HD}(T_i, T_j) = 5 = 2 \times 2 + 1$ the Cartesian distance should satisfies to inequality $\mathrm{CD}(T_i, T_j) \ge \sqrt{5}$. Therefore, at most

$$Q \le \frac{2^{15}}{\sum_{l=0}^{2} \binom{15}{l}} = \frac{2^{15}}{\binom{15}{0} + \binom{15}{1} + \binom{15}{2}} = \frac{2^{15}}{1 + 15 + 105} \approx 270 \approx 2^8$$

test patterns is available with Cartesian distance $\mathrm{CD}(T_i, T_j)$ greater than or equal to $\sqrt{5}$. Then, calculating the complement patterns, as well, the PDDTS of length $q = 2^9$ can be generated. However, the raising Cartesian distance for example up to $\sqrt{7}$ drastically reduces the length q of PDDTS to $2^6 = 64$ patterns.

To sum up, we can say that there are many methods allowing for generating antirandom like patterns, but most of them need strong pre-calculation process.

3 Generation of Antirandom Tests with Restricted Number of Patterns

For optimal AT generation with a small number q of patterns $T_0, T_1, T_2, ..., T_{q-1}$, let us step by step generate sets of patterns for $q = 2, 3, 4$. Set of patterns should cover maximal number of patterns out of all possible binary patterns in k out of N bits. Optimally it should exhaustively covers all k-subspaces simultaneously i.e., the projections of N-dimensional patterns in the test set onto any input subset of a specified size k should contain all possible patterns of k-tuples. In this context 100% coverage means all 2^k binary combinations for all k-subspace in N dimensional space for some k. For example, pseudo exhaustive test $T(6, 2) = \{000000, 000011, 011100, 101101, 110110, 111011\}$ exhaustively covers all possible k=2-bit subspaces because the projection of $T(6, 2)$ test patterns onto 2-bit subsets contains all possible patterns of 2-tuples (00, 01, 10 and 11). That is why, for the following investigation, the metrics ACT (Definition 7) and ACS (Definition 8) will play the crucial role. For different applications of exhaustive, pseudo exhaustive, random, pseudo random and antirandom tests there is the restriction that $k << N$ [1, 11]. In further analyzes we will keep it what would allow us to simplify the numerical evaluations.

It should be noted that in [14, 17, 25, 26] for the next T_i pattern generation two metrics HD and CD have been chosen. These quite general metrics were used as arguments for the fitness functions $F_1(HD)$ and $F_2(CD)$ for estimation how T_i was different from previously generated patterns $T_0, T_1, T_2,... ,T_{i-1}$. Later it will be shown that this approach is not sufficient in terms of maximal total number of binary combination generated in k out of N bits.

As the first pattern T_0 of antirandom test any N-bit pattern can be used. For example, let us start with $T_0 = \{000...0\}$. This does not result in any loss of generality. The same procedure can easily be used in the case of the test starting with any pattern out of 2^N possible patterns [14]. As an example, let us consider the case of the set of $N = 6$ bits test patterns $T = T_0, T_1, T_2,...,$ with $T_0 = \{000000\}$.

As the next pattern T_1 of the $MDAT(HD)$ and $MDAT(CD)$ we should choose the one with maximal values of fitness functions $F_1(HD)$ and $F_2(CD)$. In case of $T_0 = \{000...0\}$ the maximal values of fitness functions obviously we achieve for the pattern $T_1 = \{111...1\}$ with $F_1(HD) = N$ and $F_2(CD) = \sqrt{N}$. For our example $T_1 = \{111111\}$ and $F_1(HD) = 6$, $F_2(CD) = 2.449$. It should be emphasised that for any k out of N bits the pattern T_1 generates new binary code, namely the k-bit code with all ones. So, two patterns AT generate for any k position exactly two binary codes. That is why regardless on any metrics, the optimal second antirandom pattern T_1 is the negation of the first pattern T_0. Then, for the general case an optimal antirandom test with two patterns (q=2) is a test:

$$\begin{aligned} T_0 &= t_{i,N-1}, t_{i,N-2}, ..., t_{i,2}, t_{i,1}, t_{i,0} \\ T_1 &= \overline{t_{i,N-1}}, \overline{t_{i,N-2}}, ..., \overline{t_{i,2}}, \overline{t_{i,1}}, \overline{t_{i,0}}. \end{aligned} \tag{6}$$

As the first pattern T_0 any random N-bit pattern can be used with $t_{i,c} \in \{0, 1\}$, $c \in \{0, 1, 2, ..., N-1\}$, then the second one T_1 is the negation of T_0.

On the third step the pattern T_2 should be generated. As can be seen, symmetrical situation exists now. Any next pattern T_2 with z zeros (ones) and $N - z$ ones

(zeros), or vice versa, will be optimal in terms of fitness function $F_1(HD)$. Really, for any pattern T_2 $F_1(HD) = HD(T_2, T_0) + HD(T_2, T_1) = N - z + z = N$. Our aim is to cover maximal number of binary patterns out of all possible patterns in k out of N bits. If we use $F_1(HD)$ as the fitness function then the set of three test patterns for $N = 6$ and $T_0 = \{000000\}$ can be generated as {000000, 111111, 100000} or {000000, 111111, 110000} or {000000, 111111, 111000} etc. In all these cases $F_1(HD)$ for the third pattern equals 6. At the same time, the coverage of $k - subspaces$ in $N = 6$-dimensional space by these sets of patterns is different. For $k = 3$ in respect of ACT $F_3(100000) = 10$, $F_3(110000) = 16$ and $F_3(111000) = 18$. Therefore, at this step of analyzes of the third pattern T_2, we can delete the metric $F_1(HD)$ in our further investigations. It does not allow to choose the most optimal (in terms of maximal coverage of k-subspaces in binary N-dimensional space) set of three patterns.

In a case of function $F_2(CD)$ for the next pattern T_2 with z zeros (ones) and $N - z$ ones (zeros) $F_2(CD) = CD(T_2, T_0) + CD(T_2, T_1) = \sqrt{N - z} + \sqrt{z}$. Then, $max F_2(CD)$ can be achieved as the solution $z = N/2$ of the next equation: $\delta(\sqrt{N - z} + \sqrt{z})/\delta(z) = 0$. For further investigation suppose that N is even number and divisible by 3. In our example $N = 6$ satisfies to above mentioned conditions. Now we can evaluate the pattern T_2 on the bases of absolute criteria ACT.

Two previous patterns $T_0 = \{000...0\}$ and $T_1 = \{111...1\}$ generate two distinct patterns for any arbitrary k out of N bits. They are consisting of all zero k-bit code and all ones k-bit code. According to the absolute criteria ACT the total number $F_3(T_0)$ and $F_3(T_1)$ of different combinations generated by two patterns T_0 and T_1 (taking into account that N is a big integer number, for which $k << N$ and $N - k \approx N$) is calculated as:

$$F_3(T_0) = F_3(T_1) = \binom{N}{k} = \frac{N!}{(N-k)!k!} \approx \frac{N^k}{k!}. \quad (7)$$

Only the pattern with $N/2$ zeros and $N/2$ ones as the third pattern T_2 allows getting maximal number of new k-bit patterns calculated according to the Eq. 8, what agrees with the maximising of the Cartesian distance $F_2(CD)$.

$$F_3(T_2) = \sum_{i=1}^{k-1} \binom{N/2}{k-i} \binom{N/2}{i} \approx \frac{1}{2^k} \sum_{i=1}^{k-1} \frac{N^k}{(k-i)!i!}. \quad (8)$$

For our previous example with $N = 6$ and $k = 2$, according to Eq. 7 $F_3(T_0) = F_3(T_1) = N!/((N-k)!k!) = 6!/(4! \times 2!) = 15$. The pattern T_2 with $N/2 = 3$ zeros and $N/2 = 3$ ones generates additional new binary combinations. The number of these combinations is calculated according to Eq. 8. For the same example ($N = 6$ and $k = 2$) it gives us $F_3(T_2) = (3!/((3-1)! \times 1!) \times (3!/(1! \times (3-1)!)) = 9$ new combinations. For our example the entire amount of two-bit binary codes within the $N = 6$ bits patterns $T_0 = \{000000\}$, $T_1 = \{111111\}$ and $T_2 = \{000111\}$ is $F_3(T_0) + F_3(T_1) + F_3(T_0) = 15 + 15 + 9 = 39$.

It is easy to show that for the same example $N = 6$ bits, the maximal number of two bit patterns ($k = 2$) can be obtained for the next three patterns $T_0 = \{000000\}$, $T_1 = \{111100\}$ and $T_2 = \{001111\}$. Really, the entire number of 2−bit combinations will be calculated as $F_4(T_0, T_1, T_2) = F_3(T_0) + F_3(T_1) + F_3(T_2) = 15 + 14 + 13 = 42$. This value satisfies to the absolute criteria ACS for the set of patterns. For this criteria the number q of antirandom patterns is the key input information for the optimal set of patterns generation, rather than previously generated patterns $T_0, T_1, T_2, ..., T_{i-1}$, like in approaches have been used and described in [14, 17, 25, 26]. It should be noted that for the patterns {000000, 111111, 000111} the minimal HD between any two patterns is $HD(000000, 000111) = HD(111111, 000111) = 3$ and for the last set of patterns {000000, 111100, 001111} minimal HD is $HD(000000, 111100) = HD(000000, 001111) = HD(111100, 001111) = 4$.

These examples can be generalized for three patterns ($q = 3$) antirandom tests. The first test can be constructed as:

$$\begin{aligned} T_0 &= t_{i,N-1}, t_{i,N-2}, ..., t_{i,N/2}, t_{i,N/2-1}, ..., t_{i,2}, t_{i,1}, t_{i,0} \\ T_1 &= \overline{t_{i,N-1}}, \overline{t_{i,N-2}}, ..., \overline{t_{i,N/2}}, \overline{t_{i,N/2-1}}, ..., \overline{t_{i,2}}, \overline{t_{i,1}}, \overline{t_{i,0}} \\ T_2 &= \overline{t_{i,N-1}}, \overline{t_{i,N-2}}, ..., \overline{t_{i,N/2}}, t_{i,N/2-1}, ..., t_{i,2}, t_{i,1}, t_{i,0}. \end{aligned} \quad (9)$$

Minimal HD for above presented antirandom test with $q = 3$ patterns equals to $N/2$. At the same time the second test can be constructed as:

$$\begin{aligned} T_0 &= t_{i,N-1}, ..., t_{i,2N/3}, t_{i,2N/3-1}, ..., t_{i,N/3}, t_{i,N/3-1}, ..., t_{i,1}, t_{i,0} \\ T_1 &= \overline{t_{i,N-1}}, ..., \overline{t_{i,2N/3}}, \overline{t_{i,2N/3-1}}, ..., \overline{t_{i,N/3}}, t_{i,N/3-1}, ..., t_{i,1}, t_{i,0} \\ T_2 &= t_{i,N-1}, ..., t_{i,2N/3}, \overline{t_{i,2N/3-1}}, ..., \overline{t_{i,N/3}}, \overline{t_{i,N/3-1}}, ..., \overline{t_{i,1}}, \overline{t_{i,0}}. \end{aligned} \quad (10)$$

Minimal HD for this test 10 equals to $2N/3$. Therefore as the more efficient metric the Maximal Minimal Hamming Distance (MMHD) can be used to construct antirandom test $T = \{T_0, T_1, T_2, ..., T_{q-1}\}$ with restricted

number q of patterns. As have been shown in [32] the following statement is true.

Statement 1 The test set $T = \{T_0, T_1, T_2, ..., T_{q-1}\}$ with maximal minimal Hamming distance $HD(T_i, T_j)$, where $i \neq j \in \{0, 1, 2, ..., q-1\}$, allows to get maximal value of C(T(N,k)) for any $k \ll N$ with given N and q.

This statement satisfies to absolute criteria ACS for the set of q test patterns, which allows getting maximal number of binary combinations for arbitrary k out of N bits.

In [19] the optimal set of test patterns have been obtained for $q = 3$ and $q = 4$. Both these sets can be regarded as the algorithm for antirandom tests generation with small number q of patterns. For three patterns ($q = 3$) this algorithm can be described by Eq. 9 with the random pattern T_0 at the input. The algorithm for $q = 4$ is shown below [19].

$$\begin{aligned}
T_0 &= t_{i,N-1}, ..., t_{i,2N/3}, t_{i,2N/3-1}, ..., t_{i,N/3}, t_{i,N/3-1}, ..., t_{i,1}, t_{i,0} \\
T_1 &= \overline{t_{i,N-1}}, ..., \overline{t_{i,2N/3}}, \overline{t_{i,2N/3-1}}, ..., \overline{t_{i,N/3}}, t_{i,N/3-1}, ..., t_{i,1}, t_{i,0} \\
T_2 &= t_{i,N-1}, ..., t_{i,2N/3}, \overline{t_{i,2N/3-1}}, ..., \overline{t_{i,N/3}}, \overline{t_{i,N/3-1}}, ..., \overline{t_{i,1}}, \overline{t_{i,0}} \\
T_3 &= \overline{t_{i,N-1}}, ..., \overline{t_{i,2N/3}}, t_{i,2N/3-1}, ..., t_{i,N/3}, \overline{t_{i,N/3-1}}, ..., \overline{t_{i,1}}, \overline{t_{i,0}}
\end{aligned} \tag{11}$$

Based on the above presented algorithm and starting from the pattern $T_0 = \{000000\}$, the antirandom test with four patterns will be obtained as $T = \{T_0, T_1, T_2, T_3\} = \{000000, 111100, 001111, 110011\}$.

To compare all previously used metrics for antirandom test generation, let us consider an example have been used in several papers [14, 17, 25, 26] for illustration of antirandom test generation with $N = 3$. For $q = 4$ this test includes patterns {000, 111, 010, 101}. As an alternative for comparison the optimal antirandom test with the patterns {000, 110, 011, 101} generated according to Eq. 11 is chosen. Both tests with their metrics for $k = 2$ are shown in Tables 3 and 4. In these examples it is easy to see that the Statement 1 allows formulating the condition for the optimal short antirandom test. For the maximal minimal hamming distance case the antirandom test generates maximal number $maxC(T(3, 2)) = 12$ of binary combinations for arbitrary $k = 2$ out of $N = 3$ bits.

Table 3 Standard three bit antirandom test

Test (T)	$t_{i,2}, t_{i,1}, t_{i,0}$	$F_1(HD)$	$F_2(CD)$	$F_3(T_i) = C(T_i(3,2))$	$F_4(T) = C(T(3,2))$
T_0	000	–	–	3	10
T_1	111	3	1.7320	3	
T_2	110	3	2.4142	2	
T_3	101	6	4.1460	2	

Table 4 Optimal three bit antirandom test

Test (T)	$t_{i,2}, t_{i,1}, t_{i,0}$	$F_1(HD)$	$F_2(CD)$	$F_3(T_i) = C(T_i(3,2))$	$F_4(T) = C(T(3,2))$
T_0	000	–	–	3	12
T_1	110	2	1.7320	3	
T_2	011	4	2.8284	3	
T_3	101	6	4.2426	3	

4 Short Antirandom Tests Coverage Analyzes

The previous part of the paper gives the clear evidence that for analysis of coverage of antirandom tests, the most important information is the amount $C(T(N,k))$ of binary combinations for arbitrary k out of N bits generated by the test. Further based on this characteristic the weighted value of combinations $P(T(N,k))$ will be used as result of division $C(T(N,k))$ by number $Q(N,k)$ of all possible combinations within arbitrary k out of N bits calculated as

$$Q(N,k) = 2^k \binom{N}{k} = 2^k \frac{N!}{(N-k)!k!)}. \tag{12}$$

The weighted value $P(T(N,k))$ for the first short antirandom test T_A^2 including two patterns T_0 and T_1 generated according to Eq. 6 can be calculated as

$$\begin{aligned}
P\left(T_A^2(N,k)\right) &= \frac{C\left(T_A^2(N,k)\right)}{Q(N,k)} = \frac{F_3(T_0) + F_3(T_1)}{Q(N,k)} \\
&= \frac{1}{2^{k-1}}.
\end{aligned} \tag{13}$$

For the next short antirandom test T_A^3, $C(T_A^3(N,k))$ is obtained as the sum of three values $F_3(T_0)$, $F_3(T_1)$ and $F_3(T_2)$, where T_0, T_1 and T_2 is generated according to the algorithm (10). $F_3(T_0)$ is presented as Eq. 7 and the rest is given below.

$$F_3(T_1) = \binom{2N/3}{k} + \sum_{i=1}^{k-1} \binom{2N/3}{k-i}\binom{N/3}{i}, \tag{14}$$

$$\begin{aligned}
F_3(T_2) &= \binom{N/3}{k} + \sum_{i=1}^{k-1} \binom{2N/3}{k-i}\binom{N/3}{i} \\
&\quad + \sum_{i=1}^{k-1} \binom{N/3}{k-i}\binom{N/3}{i}
\end{aligned} \tag{15}$$

With the same assumption as in Eq. 7 (N is a big integer number, for which $k \ll N$ and $N - k \approx N$) the $C(T_A^3(N,k))$ can be calculated as

$$C\left(T_A^3(N,k)\right) = F_3(T_0) + F_3(T_1) + F_3(T_2) \approx kN^k\Bigg(\frac{1}{k!} + \frac{1}{3^k k!} + \frac{2^k}{3^k k!} + \frac{1}{3^k}\sum_{i=1}^{k-1}\frac{1}{(k-i)! \times i!} + \frac{2}{3^k}\sum_{i=1}^{k-1}\frac{2^{k-i}}{(k-i)! \times i!}\Bigg). \quad (16)$$

For simplification of the last equation we will make use of the following equality:

$$\sum_{i-1}^{k-1}\frac{k!}{(k-i)!i!} = 2^k - 2. \quad (17)$$

Then

$$P\left(T_A^3(N,k)\right) = \frac{C\left(T_A^3(N,k)\right)}{Q(N,k)} = \frac{F_3(T_0) + F_3(T_1) + F_3(T_2)}{Q(N,k)} \approx \frac{1}{2^k} + \frac{2}{3^k} - \frac{1}{2^k 3^k} + \frac{2}{2^k 3^k}\sum_{i=1}^{k-1} 2^{k-i}\binom{k}{i}. \quad (18)$$

The value of $C(T_A^4(N,k))$ for the last short antirandom test can be obtained as the sum of four arguments: $F_3(T_0)$, $F_3(T_1)$, $F_3(T_2)$ and $F_3(T_3)$. The first three arguments have been estimated before (T_0, T_1 and T_2 for algorithm (10) and (11) are the same). The fourth argument is calculated as

$$F_3(T_3) = \left(\sum_{i=1}^{k-1}\binom{2N/3}{k-i}\binom{N/3}{i} + \sum_{i=1}^{k-1}\binom{N/3}{k-i}\binom{N/3}{i}\right). \quad (19)$$

Taking into account previously adopted assumption, the weighted value of $C(T_A^4(N,k))$ is equal to

$$P\left(T_A^4(N,k)\right) = \frac{C\left(T_A^4(N,k)\right)}{Q(N,k)} = \frac{F_3(T_0) + F_3(T_1) + F_3(T_2) + F_3(T_3)}{Q(N,k)} \approx \frac{1}{2^k} + \frac{1}{3^{k-1}} - \frac{1}{2^k 3^{k-1}} + \frac{1}{2^k 3^{k-1}} \times \sum_{i=1}^{k-1} 2^{k-i}\binom{k}{i}. \quad (20)$$

The weighted number of combinations 13, 18 and 20 within arbitrary k out of N bits of short antirandom test can be regarded as the measure of their coverage. It may be interpreted as the probability, due to the random value of T_0 for all above presented algorithms.

5 Iterative Antirandom Tests

Exhaustive and pseudo exhaustive testing of an object (hardware and software) have several attractive features. In addition to the fact that test patterns usually can be generated quite easily, the process and its fault (errors) coverage are no longer dependent directly on either the model of faulty behavior or on the specific object under test. The crucial problem, however, is how to provide exhaustive input patterns simultaneously with respect to many outputs associated with the same object under test. In the past several methods have been developed for this problem [1, 23]. While these methods give test sets which, due to simplicity of the mathematical structure, can be implemented quite easily, the coverage of such sets is not always high. This seems to imply that exhaustive pattern testing would be impractical unless sufficient object under test partitioning is exercised at the design stage [12].

One of the constructive solution of the problem is the near exhaustive and near pseudo exhaustive testing based on standard test patterns [11, 23, 31]. As the set of standard patterns, which satisfies the Statement 1, is $T_S(N,k) = \{T_0, T_1, T_2, ..., T_{q-1}\}$ with $HD(T_i, T_j) = N/2$, where $i \neq j \in \{0, 1, 2, ..., q-1\}$. The number q of patterns is constant and equals to $2(\lceil log_2 N\rceil + 1)$. Their structure and generation procedure are described in [31]. If $N = 2^m$ then $q = 2(m+1)$. As an example let us use the set of standard patterns $T_S(8,k)$ for $m = 3$ (see Table 5). The coverage of described set of patterns $T_S(N,k)$, consisting on $\lceil l = log_2 N\rceil + 1$ pairs of patterns $(T_0, T_1), (T_2, T_3), \dots, (T_{2l}, T_{2l+1})$, taking the

Table 5 Standard set of patterns $TS(8, k)$

T_i	$t_{i,0}$	$t_{i,1}$	$t_{i,2}$	$t_{i,3}$	$t_{i,4}$	$t_{i,5}$	$t_{i,6}$	$t_{i,7}$
T_0	0	0	0	0	0	0	0	0
T_1	1	1	1	1	1	1	1	1
T_2	0	0	0	0	1	1	1	1
T_3	1	1	1	1	0	0	0	0
T_4	0	0	1	1	0	0	1	1
T_5	1	1	0	0	1	1	0	0
T_6	0	1	0	1	0	1	0	1
T_7	1	0	1	0	1	0	1	0

same assumptions as before ($k \ll N$ and $N - k \approx N$), can be estimated according to the next equation [31]:

$$FC(T_S(N,k))) = \left(1 - \left(\frac{2^{k-1}-1}{2^{k-1}}\right)^{\lceil log_2 N \rceil + 1}\right) 100\%. \quad (21)$$

The last equation can be regarded as the weighted value in percent of all combinations within any k out of N bits generated by the standard test $T_S(N, k)$. For $N = 2^m$ and $k \in \{2, 3, 4\}$ the coverage $FC(T_S(N, k))$ of the $T_S(N, k)$ is given in Table 6 [31].

Brief analyzes of above presented data shows that with standard tests $T_S(N, k)$ we can achieve very high level of coverage. For example, with $q = 2(m + 1) = 2(30 + 1) = 62$ iterations of $T_S(N, k)$ and $k = 3$ the level of coverage is equal to $FC(T_S(N, k)) = 99.98\%$. Moreover the complexity $O(T_S(N, k))$ of $T_S(N, k)$ with $N = 2^m$ is $2(m + 1)$ and it is the minimal one for such type of tests [9, 23]. At the same time, $FC(T_S(N, k))$ is the decreasing function with the growing k. For limited values of N the limited values of $C(T_S(N, k))$ are achieved too. These values cannot be increased by the test $T_S(N, k)$.

As a good approximation of exhaustive and pseudo exhaustive testing the random testing have been widely used [16, 21]. In comparison with deterministic tests, like $T_S(N, k)$, the random tests $T_R(N, k)$ allow to achieve the unlimited value of $FC(T_R(N, k))$, as closed to 100%, as possible. It can be done by increasing complexity $O(T_R(N, k))$ of the test.

The weighted value in percent $FC(T_R(N, k))$ of all combinations within any k out of N bits generated by the random test $T_R(N, k)$ can be obtained from the Eq. 22 [16, 21, 31].

$$FC(T_R(N,k)) = \left(1 - \left(1 - \frac{1}{2^k}\right)^l\right) 100\%. \quad (22)$$

It is obviously that maximal value of $FC(T_R(N, k))$ equals to 100% can be achieved for $l = \infty$ only. Real values close to 100% for big values of $l \to \infty$, are shown in Table 7 [31]. All the data presented in Table 7 are obtained for the same values of N and the same $l = 2(m + 1)$ number of iterations as in case of results presented in Table 6. It allows to make the comparison and conclusions. The comparative analyzes shows that for fixed value of iterations l we achieve the higher coverage in case of the deterministic tests like $T_S(N, k)$ compare to random tests $T_R(N, k)$.

Random tests $T_R(N, k) = \{T_0, T_1, T_2, ..., T_{l-1}\}$, can be regarded as iterative tests, with one ($q = 1$) pattern per iteration. In all iterations the random pattern $T_i = (t_{i,N-1}, t_{i,N-2}, ..., t_{i,2}, t_{i,1}, t_{i,0})$, $i \in \{0, 1, 2, ..., l - 1\}$ with $t_{i,j} \in \{0, 1\}$ is selected regardless of the patterns previously applied.

The key idea of the short iterative antirandom tests $T_I^q(N, k)$ is to increase the size ($q > 1$) of the one iteration of the standard random tests. The first pattern for all iterations is a random vector, like in random tests, but the rest of iteration is constructed according to above presented algorithms (9), (10) and (11). The simplest solution is the iterative test with two patterns per iteration. The first pattern T_0 is the random one and the second pattern T_1 is the negation of the T_0.

The coverage $P(T_I^q(N, k))$ expressed as a weighted value of binary combinations generated during one iteration is presented in Section 4 as an coverage of optimal antirandom test $P(T_A^q(N, k))$. It should be noted that $0 < P(T_A^q(N, k)) \leq 1$. The coverage $FC(T_I^q(N, k))$ of arbitrary antirandom tests $T_I^q(N, k)$ with $l = rq$ iterations can be estimated as:

$$FC\left(T_I^q(N,k)\right) = \left(1 - \left(1 - P\left(T_A^q(N,k)\right)\right)^r\right) 100\%. \quad (23)$$

Table 6 Standard test $T_S(N, k)$ coverage $FC(T_S(N, k))$ estimation

$N = 2^m$	2^5	2^{10}	2^{15}	2^{20}	2^{25}	2^{30}	2^{35}
$k = 2$	98.44	99.95	99.99	99.99	99.99	99.99	99.99
$k = 3$	82.20	95.77	98.99	99.76	99.92	99.98	99.99
$k = 4$	63.87	76.98	88.19	93.94	96.89	98,40	99,18

Table 7 Random test $T_R(N,k)$ coverage $FC(T_R(N,k))$ estimations

$N=2^m$	2^5	2^{10}	2^{15}	2^{20}	2^{25}	2^{30}	2^{35}
$k=2$	96.83	99.82	99.98	99.99	99.99	99.99	99.99
$k=3$	79.85	94.70	98.60	99.63	99.90	99.97	99.99
$k=4$	63.87	76.98	88.19	93.94	96.89	98,40	99.18

To compare the coverage of iterative antirandom test with the random one the next differences can be analyzed:

$$\Delta = FC\left(T_I^q(N,k)\right) - FC\left(T_R(N,k)\right) = \left(\left(1-P\left(T_A^q(N,k)\right)\right)^{l/q} - \left(1-\frac{1}{2^k}\right)^l\right). \quad (24)$$

The value of $(1-P(T_A^q(N,k)))^r - ((1-1/2^k)^q)^r$, where $r=l/q$, $(1-1/2^k)^q < 1$ and $(1-P(T_A^q(N,k))) < 1$ with the growing value r is going to zero. It allows to make the conclusion that iterative antirandom tests are more coverage compare to the random one for reasonable (small) number of iterations. At the same time, increasing the size q of one iteration, leads to decreasing the coverage of one iteration of antirandom test compare to the random test with the same number of patterns. Then, as the optimal numbers of the patterns within one iteration we can choose $q = 2$, 3 and 4.

6 Experimental and Comparative Results

To confirm the presented ideas we have compared the coverage of several antirandom like tests strategy (iterative antirandom, pure antirandom, STPG) and random one in terms of number of generated binary combinations for all arbitrary k out of N bits. By the iterative antirandom test in this experiment we mean iterative antirandom tests based on optimal vectors. In each iteration of such tests we apply four patterns where the first of them is a random one and the rest are generated according to Eq. 11. The experiments have been done for $k = 2$ and $k = 3$. Each experimental run consists of the iterative application test patterns generated by different test strategies.

Due to the fact that the authors of STPG algorithm have not indicated how to determine the adding factors [33], a random value was used in our experiments. Moreover, as have been shown before, in the case of pure AR tests, we can't use Hamming distance as the fitness function. It does not allow to get the right answer for choosing the optimal test vectors for antirandom

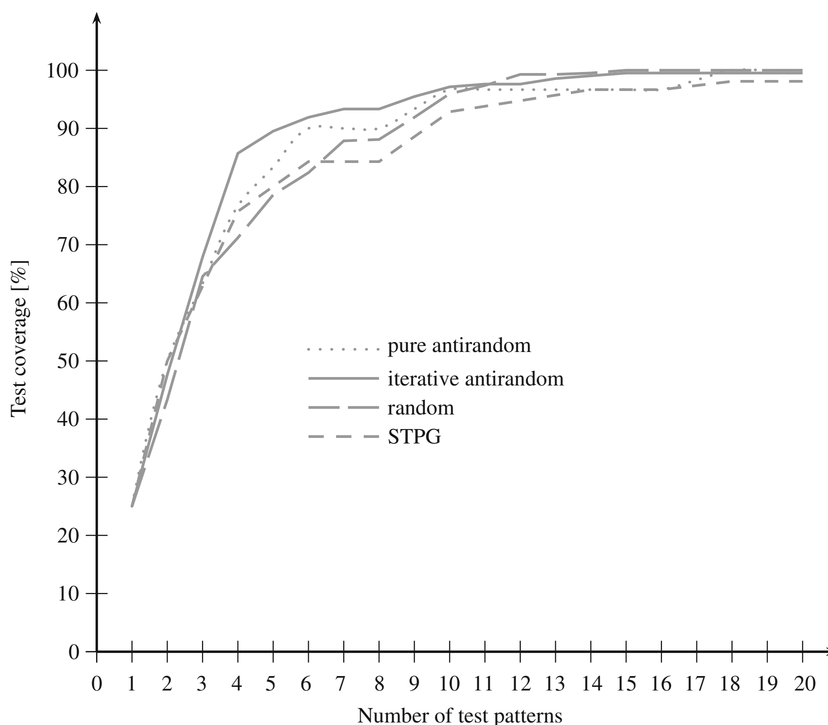

Fig. 3 The weighted number of binary combinations for all arbitrary $k = 2$ out of $N = 15$ bits generated with different test methods

test. Therefore, for this purpose a Cartesian distance was used as the fitness function.

The obtained results for $k = 2$ are shown in the Fig. 3. The x-axis represents the number of the test patterns, and the y-axis – the weighted number of binary combinations for all arbitrary $k = 2$ out of $N = 15$ bits. For Fig. 3, we observe that all coverages curves rise sharply and exhibits a smooth behavior. We observe too that for small number of iterations pure antirandom, pure random and STPG antirandom tests significantly lag in performance compared with iterative antirandom tests what agree with analytical investigation (see Eq. 24). In this case iterative antirandom test pattern generation scheme can obtain a high coverage with significantly fewer test patterns compared to other tested schemes.

The results for $k = 3$ are shown in the Fig. 4. The x-axis represents the number of the test patterns, and the y-axis – the weighted number of binary combinations for all arbitrary $k = 3$ out of $N = 15$ bits. For Fig. 4, we observe that iterative antirandom tests obtain similar coverage to pure antirandom tests. The results show that both of them generally provide higher coverage than pure-random and STPG tests schemes.

In the next experiment we have compared the coverage of the iterative antirandom tests with other iterative methods known from the literature. In this experiment three different iterative antirandom tests and random one have been investigated and compared. All investigated tests can be described as follows:

- T_R Iterative test based on random patterns ($q = 1$). Each successive pattern is randomly selected regardless of the patterns previously applied.
- T_N Iterative test based on pairs of patterns ($q = 2$). In each iteration two patterns (T_0, T_1) are applied. T_0 is a random pattern, T_1 is negation of T_0.
- T_S Iterative test based on standard patterns ($q = 4$). In each iteration four patterns (T_0, T_1, T_2, T_3) are applied. T_0 is a random pattern, T_1 is negation of T_0, T_2 is a pattern where $HD(T_0, T_2) = N/2$, T_3 is negation of T_2.
- T_O Iterative test based on optimal patterns ($q = 4$). In each iteration four patterns (T_0, T_1, T_2, T_3) are applied. T_0 is a random pattern, T_1, T_2, T_3 are generated according to Eq. 11.

The investigation was based on simulation where N-bits patterns were generated. Each successive pattern was generated according to the scheme of the investigated test. As the results we obtained the number of N-bits patterns, which we have to generate, to exhaustively cover all possible $\binom{N}{k}$ k-subspaces. Average numbers of these patterns (coverage of the tests) for $N = 15$ (in case of T_S $N = 16$) and various k are given in Table 8. We observe that in each case the iterative

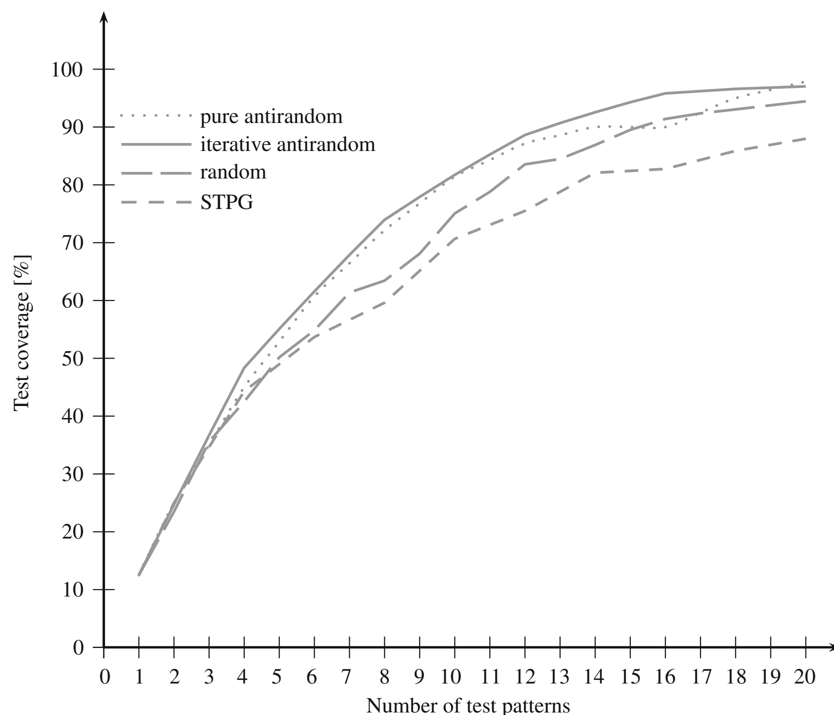

Fig. 4 The weighted number of binary combinations for all arbitrary $k = 3$ out of $N = 15$ bits generated with different test methods

Table 8 Coverage of iterative tests for $N = 15$

k	T_R	T_N	T_S	T_O
3	21.78	16.60	16.40	13.46
4	53.78	43.5	40.7	33.90
5	129.6	108.0	100.5	86.60

antirandom tests are more efficient than random one. Moreover the best results were obtained in the case of antirandom test which is based on optimal vectors (T_O). For example for $k = 5$ and iterative random test the average number of applied test patterns which is needed to exhaustively cover all possible $\binom{N}{k}$ k-subspaces equals to 129.6. For the same $k = 5$ and iterative antirandom test based on optimal vectors the average number of applied test vectors equals to 86.60. So the coverage of the test based on optimal patterns are about 33% higher in comparison to iterative random test.

7 Conclusion

A new approach to improve the test coverage of random testing was presented in this paper. Instead of standard random testing iterative antirandom testing was proposed and investigated. The key idea of this kind of tests is repeating short antirandom test. In these tests the first pattern for all iteration is a random pattern, like in random tests, but the rest of iteration is constructed according to a specific algorithm. Three different anirandom tests were investigated. Obtained results show that all analyzed iterative antirandom tests have higher efficiency in comparison to standard random test. The best efficiency was obtained for antirandom test based on optimal vectors.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Barzilai Z, Coppersmith D, Rozenberg A (1983) Exhaustive generation of bit pattern with application to VLSI self-testing. IEEE Trans Comput c-31(2):190–194
2. Chen TY, Kuo FC, Merkel RG, Tse TH (2010) Adaptive random testing: the art of test case diversity. J Syst Softw 83:60–66
3. Chen TY, Leung H, Mak IK (2004) Adaptive random testing. In: Proceedings of the 9th Asian computing science conference, ASIAN '04, pp 320–329
4. Furuya K (1989) A probabilistic approach to locally exhaustive testing. Trans Inst Electron Inf Commun Eng E72(5): 656–660
5. Geng J, Zhang J (2010) A new method to solve the "boundary effect" of adaptive random testing. In: Proceedings of the international conference on educational and information technology, ICEIT'10, vol 1, pp 298–302. Chongqing
6. Grindal M, Offutt J, Andler SF (2004) Combination testing strategies - a survey. Tech. Rep. ISE-TR-04-05, GMU Technical Report
7. Gupta S, Rajski J, Tyszer J (1996) Arithmetic additive generators of pseudo-exhaustive test patterns. IEEE Trans Comput 45:939–949
8. Hamming RW (1950) Error detecting and error correcting codes. Bell Syst Tech J 29(2):147–160
9. Hartman A, Raskin L (2004) Problems and algorithms for covering arrays. Discrete Math 284(1–3):149–156
10. Karpovsky M, Yarmolik VN (1996) Transparent random access memory testing for pattern sensitive faults. J Electron Test: Theory and Applications 9(3):251–266
11. Karpovsky MG, Goor AJvd, Yarmolik VN (1995) Pseudo-exhaustive word-oriented DRAM testing. In: Proceedings of the 1995 European conference on design and test, EDTC '95, p 126. IEEE Computer Society, Washington, DC, USA
12. Kemnitz G (1995) Synthesis of locally exhaustive test pattern generators. In: Proceedings of the 13th IEEE VLSI test symposium, VTS '95, pp 440–447. IEEE Computer Society, Washington, DC, USA
13. Liu H, Xie X, Yang J, Lu Y, Chen TY (2010) Adaptive random testing by exclusion through test profile. In: Proceedings of the 10th international conference on quality software, QSIC '10, pp 92–101. IEEE Computer Society, Washington, DC, USA
14. Malaiya YK (1995) Antirandom testing: getting the most out of black-box testing. In: Proceedings of 6th IEEE international symposium on software reliability engineering, ISSRE '95, pp 86–95. IEEE Computer Society
15. Malaiya YK, von Mayrhauser A, Srimani PK (1993) An examination of fault exposure ratio. IEEE Trans Softw Eng 19:1087–1094
16. Malaiya YK, Yang S (1984) The coverage problem for random testing. In: Proceedings of the 1994 IEEE international test conference, ITC '84, pp 237–245
17. Mayrhauser A, von Bai A, Chen T, Anderson C, Hajjar A Fast antirandom (FAR) test generation. In: Proceedings of 3rd IEEE international symposium on high-assurance systems engineering, HASE '98, pp 262–269. IEEE Computer Society, Washington, DC, USA (1998)
18. McCaffrey JD An empirical study of the effectiveness of partial antirandom testing. In: Proceedings of the 18th international conference on software engineering and data engineering, SEDE, pp 260–265 (2009)
19. Mrozek I, Yarmolik V (2009) Problemy funkcjonalnego testowania pamieci RAM. Bialystok University of Technology, Bialystok, Poland. ISSN 0867-096X
20. Schnurmann H, Lindbloom E, Carpenter R (1975) The weighted random test-pattern generator. IEEE Trans Comput 24:695–700
21. Seth S, Agrawal V, Farhat H (1990) A statistical theory of digital circuit testability. IEEE Trans Comput 39:582–586
22. Sosnowski J (1998) Experimental evaluation of pseudorandom test effectiveness. In: Proceedings of 24th Euromicro Conference, pp 184–187. IEEE Computer Society, Vasteras, Sweden

23. Tang D, Woo L (1983) Exhaustive test pattern generation with constant weight vectors. IEEE Trans Comput 32:1145–1150
24. Tappenden A, Miller J (2009) A novel evolutionary approach for adaptive random testing. IEEE Trans Reliab 58(4):619–633
25. Wu SH, Jandhyala S, Malaiya YK, Jayasumana AP Antirandom testing: a distance-based approach. VLSI Des 2008:1–2 (2008)
26. Wu SH, Malaiya YK, Jayasumana AP (1998) Antirandom vs. pseudorandom testing. In: Proceedings of IEEE international conference on computer design: VLSI in computers and processors, ICCD '98, p 221
27. Wunderlich HJ (1988) Multiple distributions for biased random test patterns. In: Proceedings of the 1988 international conference on test: new frontiers in testing, ITC'88, pp 236–244. IEEE Computer Society, Washington, DC, USA
28. Wunderlich HJ, Kiefer G (1996) Bit-flipping BIST. In: Proceedings of the 1996 IEEE/ACM international conference on computer-aided design, ICCAD '96, pp 337–343. IEEE Computer Society, Washington, DC, USA
29. Xu S (2008) Orderly random testing for both hardware and software. In: Proceedings of the 2008 14th IEEE pacific rim international symposium on dependable computing, pp 160–167. IEEE Computer Society, Washington, DC, USA
30. Xu S, Chen J (2002) Maximum distance testing. In: Asian test symposium, pp 15–20
31. Yarmolik S (2010) Iterative near pseudoexhaustive random testing. Informatics 2(26):66–75
32. Yarmolik SV, Mrozek I (2007) Multi background memory testing. In: Proceedings of the 14th international conference mixed design of integrated circuits and systems, MIXDES '07, pp 511–516. IEEE Computer Society, Ciechocinek, Poland
33. Yiunn D, Bin A'ain A, Khor Ghee J (2010) Scalable test pattern generation (STPG). In: Proceedings of the industrial electronics applications (ISIEA), 2010 IEEE Symposium on, ISIEA '2010, pp 433–435
34. Zhou Z (2010) Using coverage information to guide test case selection in adaptive random testing. In: Proceedings of the IEEE 34th annual computer software and applications conference workshops, pp 208–213

Ireneusz Mrozek received his M.Sc. and Ph.D. degrees in computer science in 1994 and 2004 respectively. Since 1994 he has been employed at theFaculty of Computer Science of Bialystok University of Technology, Bialystok, Poland. His main research interests include the area of diagnostic testing of embedded memories. Especially he focuses on transparent tests for RAM as well as the applicatiion of these tests in BIST or BISR scheme.

Vyacheslav N. Yarmolik received the MSc degree from the Computer Science Department of the Minsk Radio-Engineering Institute, Republic of Belarus, USSR, in 1973. In 1979, he received the PhD degree and the Dr. degree from the same institute in 1980. After being an assistant professor from 1974 to 1980 and an associate professor from 1980 to 1990, he became a full professor and director of the Research Laboratory INforTEST at the Minsk Radio-Engineering Institute in 1990. The institute became the Belarussian State University of Informatics and Radio-Electronics in 1994. He also became chairman of the Computer Science Department. Since 1996 he concurrently holds the position of a full professor in the Faculty of Computer Science of Bialystok University of Technology, Bialystok, Poland.