Research Article

Antirandom Testing: A Distance-Based Approach

Shen Hui Wu,¹ Sridhar Jandhyala,¹ Yashwant K. Malaiya,² and Anura P. Jayasumana¹

¹ Electrical and Computer Engineering Department, Colorado State University, Fort Collins, CO 80523, USA

² Computer Science Department, Colorado State University, Fort Collins, CO 80523, USA

Correspondence should be addressed to Anura P. Jayasumana, anura.jayasumana@colostate.edu

Received 19 March 2007; Accepted 16 January 2008

Recommended by Jacob Abraham

Random testing requires each test to be selected randomly regardless of the tests previously applied. This paper introduces the concept of antirandom testing where each test applied is chosen such that its total distance from all previous tests is maximum. This spans the test vector space to the maximum extent possible for a given number of vectors. An algorithm for generating antirandom tests is presented. Compared with traditional pseudorandom testing, antirandom testing is found to be very effective when a high-fault coverage needs to be achieved with a limited number of test vectors. The superiority of the new approach is even more significant for testing bridging faults.

Copyright © 2008 Shen Hui Wu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Random testing is a form of black-box testing which does not require knowledge of the circuit under test. It avoids the problem of deterministic test generation using structural information about the circuit under test. Available evidence suggests that pseudorandom testing may be a reasonable choice for obtaining a moderate degree of confidence, however it becomes inefficient when only hard-to-test faults remain [1]. Generally in BIST environment, pseudorandom test vectors are used for testing digital circuits. In pseudorandom testing, each test vector is chosen with equal probability out of a pool that initially contains M different vectors without replacement [2]. A common approach is to generate them using an autonomous linear feedback shift register (ALFSR) based on a primitive polynomial [3]. An alternative approach is to use a cellular automata-based generator [4, 5]. The use of Galois LFSRs (GLFSRs) as test pattern generators for BIST schemes employing multiple scan chains is investigated in [6, 7]. The effectiveness of such techniques can be enhanced by using some information about the circuit under test and tailoring the generator hardware to achieve higher coverage faster [8]. Weighted random pattern techniques adjust the probability generating 1's and 0's at each input, rather than using equal probabilities. Such an approach has been shown to increase efficiency of testing, especially for modules

such as PLAs [9]. In some cases test generation for self-testing may be implemented in software [10].

Pseudorandom testing does not exploit some information that is available in black-box testing environment. This information consists of the previous tests applied. If an experienced test engineer is generating tests by hand, he would select each new test such that it covers some part of the functionality not yet covered by tests already generated. The objective of this paper is to formally define a black-box approach that uses this information, and to propose schemes that may allow such test generation to be done automatically. We term this approach *antirandom* testing, since selection of each test explicitly depends on the tests already obtained. This technique can potentially be used in place of the traditional pseudorandom testing.

In Section 2 we define and characterize the antirandom test patterns. A procedure for generating antirandom tests is described. Then we compare pseudorandom and antirandom strategies by evaluating their effectiveness for ISCAS benchmark circuits. Both stuck-at and bridging faults are considered.

2. BINARY ANTIRANDOM SEQUENCES

Antirandom testing [11] is a black-box strategy like pseudorandom testing, meaning that it assumes no information

FIGURE 1: Construction of 3-bit MHDATS.

about the internal implementation of the circuit. However, it explicitly uses information about the past vectors applied for generating a new vector. Here we start with formal definitions of the terms used and then examine construction of antirandom sequences.

Definition 1. Antirandom test sequence (ATS) is a test sequence in which a test t_i is chosen such that it satisfies some criterion with respect to all tests $t_0, t_1, \dots, t_{i-1}$ applied before. In this paper we use two specific criteria based on the *distance* concept introduced below.

Definition 2. Distance is a measure of how different two vectors t_i and t_j are. Here we use two measures of distance defined below.

Definition 3. Hamming distance (HD) [12] is the number of bits in which two binary vectors differ. It is not defined for vectors containing continuous values.

Definition 4. Cartesian distance (CD) between two vectors, $A = \{a_N, a_{N-1}, \dots, a_1, a_0\}$ and $B = \{b_N, b_{N-1}, \dots, b_1, b_0\}$, is given by

$$CD(A, B) = \sqrt{(a_N - b_N)^2 + (a_{N-1} - b_{N-1})^2 + \dots + (a_0 - b_0)^2}. \quad (1)$$

Since the two test vectors are binary, then (1) can be written as

$$\begin{aligned} CD(A, B) &= \sqrt{|a_N - b_N| + |a_{N-1} - b_{N-1}| + \dots + |a_0 - b_0|} \\ &= \sqrt{HD(A, B)}. \end{aligned} \quad (2)$$

Definition 5. Total Cartesian distance (TCD) for any vector is the sum of its Cartesian distances with respect to all previous vectors. We refer to the total distance when Hamming and Cartesian distances are used by $THD(t_i)$ and $TCD(t_i)$, respectively.

Definition 6. Maximal distance antirandom test sequence (MDATS) is a test sequence such that each test t_i is chosen to make the total distance between t_i and each of $t_0, t_1, \dots, t_{i-1}$

maximum, that is,

$$TD(t_i) = \sum_{j=0}^{i-1} D(t_i, t_j) \quad (3)$$

is maximum for all possible choices of t_i .

Definition 7. Maximal Hamming distance antirandom test sequence (MHDATS) a MDATS that uses Hamming distance as the distance measure.

Definition 8. Maximal Cartesian distance antirandom test sequence (MCDATS) a MDATS that uses Hamming distance as the distance measure.

For functional testing, we have no structural information available about the actual circuit implementation. Using maximal distance criterion, every time we attempt to find a test vector as different as possible from all previously applied vectors. The antirandom testing scheme thus attempts to keep testing as efficient as possible. In this approach, we are using the hypothesis that if two input vectors 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 vectors is large, then the set of faults detected by one is likely to contain only a few of the faults detected by the other. Extensive evaluation of Torino benchmarks for fault coverage in [13] has confirmed this hypothesis to be valid for all the benchmarks tested.

If testing is less than exhaustive, then maximum distance antirandom testing (MDAT) is likely to be more efficient than either random or pseudorandom testing. Even when exhaustive testing is feasible, MDAT is likely to detect the presence of faults earlier.

Procedure 1. Construction of a MHDATS (MCDATS)

Step 1. For each of N input variables, assign an arbitrarily chosen value to obtain the first test vector. This does not result in any loss of generality.

Step 2. To obtain each new vector, 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 vectors.

TABLE 1: 3-bit MHDTS.

Test	xyz	THD	TCD
t_0	0 0 0	—	—
t_1	1 1 1	3	1.7320
t_2	0 1 0	3	2.4142
t_3	1 0 1	6	4.146
t_4	1 0 0	6	4.8284
t_5	0 1 1	9	6.5604
t_6	1 1 0	9	7.2426
t_7	0 0 1	12	8.9746

Step 3. Repeat Step 2 until all 2^N combinations have been used, or until the desired number of vectors have been generated.

This procedure uses exhaustive search. As we will see later, the computational complexity can be greatly reduced. Note that the procedure ensures that a vector will not be repeated.

To illustrate the process of generating MDATS, we consider in detail the generation of a complete sequence for three binary variables.

Example 9. Consider a system with three inputs $\{x, y, z\}$. We will illustrate the generation of MHDATS using a cube with each node representing one input combination.

Let us start with the input $\{0, 0, 0\}$. This does not result in any loss of generality as the procedure can easily be used for sequences starting with any vector. As we will see later, the polarity of any variable can be inverted. The next vector t_1 of the MHDTS is obviously $\{1, 1, 1\}$ with $\text{THD}(t_1) = 3$. At this point, the situation is shown in Figure 1(a), where the input combinations already chosen are marked.

As can be visually seen, a symmetrical situation exists now. Any vector chosen would have $\text{HD} = 1$ from one of the past chosen vectors and $\text{HD} = 2$ from the others. If we allow the variables to be reordered, then without any loss of generality we have the following choices:

$$\begin{aligned}
 t_0 &= \{0, 0, 0\}; \\
 t_1 &= \{1, 1, 1\}; \quad \text{THD} = 3; \\
 t_2 &= \{0, 1, 0\}; \quad \text{THD} = 3 \quad \text{or} \quad t_2 = \{1, 0, 1\}; \quad \text{THD} = 3.
 \end{aligned}
 \tag{4}$$

Let us consider the first choice. After $t_2 = \{1, 0, 1\}$, the clear choice for t_3 is $\{1, 0, 1\}$ at the opposite corner of the cube. The situation now is shown in Figure 1(b).

Again a symmetrical situation exists. Any one of the remaining vectors have the same relationship with the set of vectors already chosen. Let us pick $\{1, 0, 0\}$ as t_4 . The next vector t_5 then has to be $\{0, 1, 1\}$ at the opposite corner of the cube. We can again choose any one of two remaining vectors as shown in Figure 1(c). Let us choose $t_6 = \{1, 1, 0\}$ which leaves $t_7 = \{0, 0, 1\}$. The complete MHDTS obtained here is given as sequence 1 in Table 1.

In this example, it is easy to see that with our chosen vectors for t_0 , t_1 and the two choices for t_2 , we could have con-

structed 16 distinct MHDATSs using all of the later choices available. We can verify that all of these are also MCDATSs.

A MCDATS needs to satisfy a more strict requirement than a MHDATS, for example, $t_0 = \{0, 0, 0, 0\}$, $t_1 = \{1, 1, 1, 1\}$, $t_2 = \{0, 0, 0, 1\}$, is a MHDATS but not a MCDATS. If we replace t_2 by $\{0, 1, 0, 1\}$ with equal number of 1's and 0's, it would be a MCDATS.

A large number of experiments with construction of MHDATSs and MCDATSs have been done. Based on these, the following results can be stated [11].

Definition 10. If a sequence B is obtained by reordering the variables of sequence A, then B is a *variable-order-variant* (VOV) of A.

Definition 11. If a sequence B is obtained by complementing some of the variables of sequence A, then B is a *polarity-variant* (PV) of A.

Theorem 1. *If a sequence B is a variable-order variant of a MHDATS (MCDATS) A, then B is also a MHDATS (MCDATS).*

Proof. Note that Hamming or Cartesian distance is independent of how the variables are ordered. Thus, a variable-order variant of a sequence will exhibit the same distance properties. Hence, if a sequence B is variable-order variant of a MHDATS (MCDATS) A, then B is also a MHDATS (MCDATS). $\square$

Theorem 2. *If a sequence B is a polarity variant of a MHDATS (MCDATS) A, then B is also MHDATS (MCDATS).*

Proof. For a pair of vectors, the distance remains the same if the same set of variables in both vectors is complemented. Thus, a polarity variant of a sequence will exhibit the same distance properties. Hence, if a sequence B is polarity variant of a MHDATS (MCDATS) A, then B is also a MHDATS (MCDATS). $\square$

Theorem 3 (see [11]). *A MHDATS (MCDATS) will always contain complementary pairs of vectors, that is, t_{2k} will always be followed by t_{2k+1} which is complementary for all bits in t_{2k} where $k = 1, 2, \dots$*

Proof. The first two vectors t_0 and t_1 will always be a complementary pair. Below we will show that if the vectors $t_0, t_1, \dots, t_{2k-1}$ are complementary pairs, then the vectors t_{2k} and t_{2k+1} will also be a complementary pair. Thus t_2 and t_3 , hence t_4 and t_5 , and so on, will all be complementary pairs.

Let us assume a MHDATS contains an even number of vectors, $t_0, t_1, \dots, t_{2k-2}, t_{2k-1}$, such that a vector with an odd subscript is a complement of the preceding vector. Let us now assume that a vector t_{2k} is found such that its total Hamming (Cartesian) distance from all of $t_0, \dots, t_{2k-1}$ is maximal.

If we change the polarity of all the variables, we get the sequence $\bar{t}_0, \bar{t}_1, \dots, \bar{t}_{2k-2}, \bar{t}_{2k-1}, \bar{t}_{2k}$, where $\bar{t}_i$ indicates

complement of t_i . By Theorem 2, this is also a MHDATS (MCDATS). Now since $\bar{t}_1 = t_0$, $\bar{t}_0 = t_1$, and so on, the partial sequence $\bar{t}_0, \bar{t}_1, \dots, \bar{t}_{2k-2}, \bar{t}_{2k-1}$ contains the same vectors as $t_0, t_1, \dots, t_{2k-2}, t_{2k-1}$ but in a different order. Thus, $\bar{t}_{2k}$ has the same total Hamming (Cartesian) distance as t_{2k} with respect to $t_0, t_1, \dots, t_{2k-2}, t_{2k-1}$.

Now after having constructed the MHDATS (MCDATS) $t_0, t_1, \dots, t_{2k-2}, t_{2k-1}, t_{2k}$, the addition of $\bar{t}_{2k+1} = t_{2k}$ will keep it a MHDATS (MCDATS) because

- (1) $\bar{t}_{2k}$ provides the maximal total HD (CD) with respect to $\{t_0, t_1, \dots, t_{2k-2}, t_{2k-1}\}$;
- (2) $\bar{t}_{2k}$ also provides the maximal HD (CD) with respect to t_{2k} . $\square$

The following procedures can be used to construct MHDATS (MCDATS) for a larger number of variables starting with MHDATS (MCDATS) for a smaller number of variables.

Procedure 2. Expansion of MHDATS (MCDATS)

Step 1. Start with a complete MHDATS of N variables, $X_{N-1}, X_{N-2}, \dots, X_1, X_0$.

Step 2. For each vector t_i , $i = 0, 1, \dots, (2^N - 1)$, add an additional bit corresponding to an added variable X_N , such that t_i has the maximum total HD (CD) with respect to all the previous vectors.

Procedure 3. Expansion and Unfolding of a MHDATS (MCDATS)

Step 1. Start with a complete $(N - 1)$ variable MHDATS (MCDATS) with 2^{N-1} vectors.

Step 2. Expand by adding a variable using Procedure 2. We now have the first $(2^N/2)$ vectors needed.

Step 3. Complement one of the columns and append the resulting vectors to the first set of vectors obtained in Step 2. Here, it would be convenient to complement the variable added in Step 2.

The above procedures have been implemented in a program called ATG [14]. It generates MCDATs which are also MHDATs. The application of antirandom testing for software has been reported in [15]. Three related approaches termed *fast antirandom* [16], *random-like* [17] and *maximum distance* [18] testing have recently been proposed that attempt to incorporate some of the features of antirandom sequences.

A pseudorandom test sequence attempts to achieve randomness. However, Pradhan and Chatterjee [19] have shown that LFSR-based sequences fail many of the tests used to measure randomness. An antirandom test sequence attempts to span the input space by applying inputs that are as different as possible [16, 20]. It does it by being deliberately nonrandom. Let us compare the first 30 test patterns for 14-input sequences generated using antirandom property and conventional pseudorandom tests, as given in Table 2. Pseudo0 and Pseudo2 represent the pseudorandom sequences starting with the seeds 00000000000000 and 01010101010101, respectively. The successive vectors in the time sequence are listed sequentially. A more visual representation is given in

TABLE 2: Comparison of randomness: antirandom versus Pseudorandom.

Antirandom	Pseudo0	Pseudo2
00000000000000	00000000000000	01010101010101
11111111111111	10000000000000	00101010101010
01010101010101	11000000000000	10010101010101
10101010101010	11100000000000	11001010101010
00110011011000	11110000000000	01100101010101
11001100100111	11111000000000	00110010101010
01100110001101	11111100000000	10011001010101
10011001110010	11111110000000	11001100101010
00011110010011	11111111000000	01100110010101
11100001101100	11111111100000	00110011001010
01001011000110	11111111110000	10011001100101
10110100111001	11111111111000	11001100110010
00101101011110	01111111111100	11100110011001
11010010100001	00111111111110	111100110011000
01111000001011	00011111111111	11111001100110
10000111110100	10001111111111	01111100110011
00001111101001	01000111111111	10111110011001
11110000010110	10100011111111	11011111001100
01011010111100	01010001111111	11101111100110
10100101000011	10101000111111	01110111110011
00111100100100	01010100011111	10111011111001
11000011011011	10101010001111	11011101111100
01101001110001	01010101000111	11101110111110
10010110001110	00101010100011	11110111011111
00010001101111	10010101010001	01111011101111
11101110010000	01001010101000	10111101110111
01000100111010	10100101010100	11011110111011
10111011000101	01010010101010	11101111011101
00100010110111	10101001010101	01110111101110
11011101001000	11010100101010	10111011111001

Figure 2. A black pixel indicates value 0, while the white pixel indicate value 1. The time progresses from top to bottom for each bitmap.

3. EFFECTIVENESS OF ANTIRANDOM AND PSEUDORANDOM TESTING

To compare the effectiveness of antirandom sequences with pseudorandom tests generated using linear feedback shift registers (LFSRs), a series of experiments has been performed. We have measured the effectiveness of a test set using two coverage measures. The *stuck-at 0/1* coverage is the ratio of the number of stuck-at faults detected to the total number of faults. Some faults may be undetectable. We have also evaluated the *bridging-fault* coverage in the IDDQ test environment. A bridging fault is assumed to be detected if the two nodes have opposite logic values, because in such a case a very high quiescent supply current would indicate the presence of a fault.

The basic characteristics of the ISACAS85 combinational circuit used in the study are given in Table 3. It lists the detail

FIGURE 2: State-time diagrams of antirandom, pseudorandom pattern generator.

information such as the number of inputs and outputs, number of gates, the circuit function, and the total number of stuck-at faults.

3.1. Testing combinational circuits for stuck-at fault

A pseudorandom test generator will generate different sequences depending on the seed used, and thus different seeds will yield different results. To see the variation in coverage due to the choice of the seed value, three different seed values were used for the pseudorandom test generator.

We have simulated several ISCAS85 combinational benchmark circuits using Nemesis, a software tool developed at University of California, Santa Cruz [21].

Figures 3 and 4 show the stuck-at fault coverage obtained by the test sequence generators for the circuits C880 and C3540. In these plots, the x-axis represents the number of the test patterns, and the y-axis fault coverage. Tables 4, 5, 6, and 7 show the fault coverage for different number of test patterns for c880, c3540, c1355, and c499, respectively, with two digits after the decimal point. Note that the Pseudo0, Pseudo1, and Pseudo2 in the figures and plots represent the different initial seeds for pseudorandom generator which are all zeros (000000...), all ones (111111...), and alternating bits (101010...).

For c880 (Figure 3), we observe that antirandom tests perform well, obtaining 90.87% coverage for 100 vectors. The fault coverage curve rises sharply and exhibits a smooth behavior. In case of pseudorandom tests, there is a significant difference depending on the initial seed, the coverage obtained with 100 vectors ranges from 44.90% to 73.76%. For all three seeds, pseudorandom tests significantly lag in performance compared with antirandom tests. We also observe that the plots for pseudorandom tests show somewhat irregular growth. The plot for c3540 (Figure 4) shows that antirandom testing is again the best, the difference between it and pseudorandom testing is most pronounced at about 70% coverage. Similar observations can be made for c1355 and c499 (Tables 6 and 7).

FIGURE 3: C880 stuck-at fault coverage simulation.

The results show that

- (1) the antirandom sequences generally provide higher coverage than pseudorandom testing;
- (2) the new test pattern generation scheme can obtain a high-fault coverage with significantly fewer test patterns compared to pseudorandom testing;
- (3) for pseudorandom testing, the effectiveness can depend on the initial seed chosen.

These results show that the proposed scheme exercises the circuits under test better because it spans the test vector space better than pseudorandom testing.

3.2. Testing combinational circuit for bridging faults

A bridging fault is a short between two nodes in a circuit. We have applied the new test pattern generation scheme to test bridging faults in the four ISCAS benchmark circuits. The bridging faults are identified by Carafe [22] by considering the layout information. Nemesis assumes that a bridging fault is being tested in the IDDQ test environment. If the two bridged nodes have opposite logic values, then the bridging fault will cause a high value of IDDQ thus detecting the fault.

Figures 5 and 6 give the fault coverages obtained for the circuits c880 and c3540 for different schemes. Tables 8, 9, 10, and 11 list the fault-coverage values for specific number of test patterns for the four circuits.

Figure 5 for c880 shows that the difference between the proposed approach and the traditional pseudorandom testing is quite remarkable. Furthermore with antirandom sequences, the coverage of bridging faults rises much faster than that for stuck-at faults. The gap between the curves for the new approaches and pseudorandom testing is wider

TABLE 3: Characteristic of simulated combinational circuits.

Circuit	Number of inputs	Number of outputs	Number of gates	Total lines	Fanout stem	Circuit function	Total stuck-at faults	Total bridging faults
c880	60	26	383	880	125	ALU and control	942	3254
c3540	50	22	1669	3540	579	ALU and control	3428	16459
c1355	41	32	546	1355	259	ECAT	1574	4422
c499	41	32	202	499	59	ECAT	758	2781

TABLE 4: Stuck-at fault coverage for c880.

Test no.	1	5	10	20	50	75	100	200
Antirandom	8.91	42.99	67.46	76.86	85.56	88.64	90.87	95.01
Pseudo0	8.91	9.87	12.63	22.29	37.15	40.13	44.90	81.95
Pseudo1	10.93	13.59	15.39	16.24	42.12	58.49	66.45	82.27
Pseudo2	29.94	40.55	44.59	46.39	65.81	71.66	73.46	88.21

FIGURE 4: C3540 stuck-at fault coverage simulation.

FIGURE 5: C880 bridging fault coverage.

for bridging faults. The same behavior is observed for other three circuits [23]. For example, we can consider the number of vectors needed to obtain a stuck-at coverage of about 90% using the new approach. For c880 for the same number of vectors (about 100), pseudorandom vectors obtain coverage values of 44.90 for Pseudo0 to 73.46 for Pseudo2, that is, about 19–50% less. For bridging faults, for about the same level of coverage, this difference between antirandom and pseudorandom coverage for c880 widens to 31–63%. The similar gaps for c3540 widens from 3–4% to 25–52%, for c1355 from about 0–3 to 4–14%, and for c499 from 0–5% to 4–15%. This makes the proposed schemes suitable for

generating tests for bridging faults in the IDDQ test environment. Using a very few vectors, very high coverage for bridging faults can be achieved.

An explanation for this improvement can be found by observing the fact that different inputs are toggled relatively independently with the proposed schemes, making it more likely that any two nodes will have opposite logic values.

4. CONCLUSIONS

Antirandom testing is a new test generation approach. Here we have demonstrated that it can achieve high-fault coverage

TABLE 5: Stuck-at fault coverage for c3540.

Test no.	1	5	10	20	50	75	100	200
Antirandom	14.21	34.98	47.78	63.85	75.32	80.06	83.50	87.51
Pseudo0	14.21	17.21	20.21	26.22	51.78	60.18	66.24	83.43
Pseudo1	8.66	28.56	30.66	32.70	53.25	60.79	66.71	83.51
Pseudo2	11.84	33.11	36.98	41.36	56.74	67.00	70.85	84.48

TABLE 6: Stuck-at fault coverage for c1355.

Test no.	1	5	10	20	50	75	100	200
Antirandom	8.13	34.18	53.62	72.23	80.62	87.03	88.43	92.56
Pseudo0	8.13	32.91	37.04	44.66	74.14	81.89	85.13	91.48
Pseudo1	14.74	36.59	45.43	59.08	75.47	83.79	88.62	92.12
Pseudo2	9.14	46.95	48.86	60.48	78.90	83.54	86.59	91.42

TABLE 7: Stuck-at fault coverage for c499.

Test no.	1	5	10	20	50	75	100	200
Antirandom	12.66	49.34	69.92	84.03	88.12	92.35	93.40	97.22
Pseudo0	12.66	47.89	53.30	60.81	84.96	89.71	92.61	96.30
Pseudo1	22.16	48.42	57.26	68.33	84.82	89.84	93.00	96.17
Pseudo2	12.66	64.51	67.28	76.91	86.93	88.18	91.55	96.59

TABLE 8: Bridging fault coverage for c880

Test no.	1	5	10	20	50	75	100	200
Antirandom	10.26	62.55	85.69	93.2	97.37	98.24	98.95	98.95
Pseudo0	10.26	17.46	26.84	39.4	65.97	69.27	72.16	95.29
Pseudo1	15.60	25.05	29.69	33.91	58.78	76.16	85.19	95.10
Pseudo2	25.84	53.4	60.14	64.12	85.5	90.72	91.37	97.71

TABLE 9: Bridging fault coverage for c3540.

Test no.	1	5	10	20	50	75	100	200
Antirandom	16.57	60.46	78.18	88.55	95.17	96.22	97.47	98.67
Pseudo0	16.57	28.05	29.09	42.41	71.38	85.46	89.34	97.68
Pseudo1	18.52	24.96	31.93	41.69	81.48	90.74	94.88	97.94
Pseudo2	22.34	44.55	57.25	65.55	86.37	90.57	93.38	96.75

TABLE 10: Bridging fault coverage for c1355.

Test no.	1	5	10	20	50	75	100	200
Antirandom	10.83	59.66	82.55	92.59	96.84	97.31	97.76	98.10
Pseudo0	10.83	45.23	59.01	78.86	93.23	95.97	96.92	98.05
Pseudo1	18.29	31.27	57.27	79.93	95.24	96.97	97.73	98.48
Pseudo2	24.88	51.3	84.86	88.71	96.05	96.03	97.46	98.42

TABLE 11: Bridging fault coverage for c499.

Test no.	1	5	10	20	50	75	100	200
Antirandom	4.52	71.09	84.62	91.37	97.15	98.55	99.52	99.76
Pseudo0	4.52	37.98	63.04	77.48	96.62	98.21	98.91	99.52
Pseudo1	18.77	45.62	68.86	85.52	98.45	99.21	99.72	99.81
Pseudo2	31.34	54.50	63.40	87.07	97.88	98.35	99.42	99.68

FIGURE 6: C3540 bridging fault coverage.

much faster than the conventional pseudorandom testing. It has also been successfully applied for software testing [15] and testing of VHDL descriptions [24]. Its effectiveness is specially remarkable for bridging faults. The scheme is well suited for IDDQ testing because it provides very good coverage with only a few vectors when black-box testing is used. One possible way to exploit the capabilities of antirandom testing is to use it until a suitable high coverage is obtained, and then to switch to deterministic testing.

This paper considers the testing of combinational blocks. Testing of sequential circuits presents challenges that need to be considered separately. In DFT approaches, the problem of testing sequential circuits is converted to the problem of testing combinational blocks during the test mode. In such cases, antirandom testing for combinational blocks is still applicable. However, testing becomes complex if no such DFT approaches are used. Studies have shown [25] that holding a vector for a few clock periods can improve coverage. This suggests that using antirandom testing for sequential circuits will require the development of special approaches. Such approaches are currently being researched. One can also envisage circuits for which clustering of patterns, as opposed to antirandom, may be beneficial. For example, a vector that sensitizes a long path in the circuit can be slightly modified to detect many other faults. A second example is a single AND gate with a large number of inputs. Most of good tests are clustered around all-1 vector. However, exploiting such scenarios usually require the use of structural information about the circuit.

The concept of complementary pairs was one of our contributions as a part of the antirandom concept. While application of a pattern and its complement [17] with pseudorandom testing can provide some benefit, antirandom is more

effective due to the fact that it increases the overall Hamming distance between new vectors in addition to using the complement of each vector.

In this paper, we have considered only *black-box* testing which assumes that we do not have any structural information available. It is possible to generate antirandom sequences that can exploit some structural information thus further increasing coverage [8, 26]. Further research is needed to develop techniques that will allow us to do that.

ACKNOWLEDGMENTS

This work was supported in part by a BMDO funded project monitored by ONR. The authors thank Ruyi Fan, Aditya Kaul, and Vivek Ukidve for their assistance.

REFERENCES

- [1] Y. K. Malaiya and S. Yang, "The coverage problem for random testing," in *Proceedings of IEEE International Test Conference (ITC '84)*, pp. 237–245, Philadelphia, Pa, USA, October 1984.
- [2] K. D. Wagner, C. K. Chin, and E. J. McCluskey, "Pseudorandom Testing," *Transactions on Computers*, vol. 36, no. 3, pp. 332–343, 1987.
- [3] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI Pseudorandom Techniques*, John Wiley & Sons, New York, NY, USA, 1987.
- [4] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications*, vol. 1, IEEE Computer Society Press, Los Alamitos, Calif, USA, 1997, Please provide the rest of the authors if possible.
- [5] P. Dasgupta, S. Chattopadhyay, P. P. Chaudhuri, and I. Sengupta, "Cellular automata-based recursive pseudoexhaustive test pattern generator," *IEEE Transactions on Computers*, vol. 50, no. 2, pp. 177–185, 2001.
- [6] S. Chidambaram, D. Kagaris, and D. K. Pradhan, "Comparative study of CA with phase shifters and GLFSRs," in *Proceedings of IEEE International Test Conference (ITC '95)*, pp. 926–935, Austin, Tex, USA, November 2005.
- [7] D. K. Pradhan, D. Kagaris, and R. Gambhir, "A hamming distance based test pattern generator with improved fault coverage," in *Proceedings of the 11th IEEE International On-Line Testing Symposium (IOLTS '05)*, pp. 221–226, Saint Raphael, French Riviera, France, July 2005.
- [8] C.-A. Chen and S. K. Gupta, "A methodology to design efficient BIST test pattern generators," in *Proceedings of IEEE International Test Conference (ITC '95)*, pp. 814–823, Washington, DC, USA, October 1995.
- [9] J. A. Waicukauski and E. Lindbloom, "Fault detection effectiveness of weighted random patterns," in *Proceedings of IEEE International Test Conference (ITC '88)*, pp. 245–255, Washington, DC, USA, September 1988.
- [10] L. Chen and S. Dey, "Software-based self-testing methodology for processor cores," *IEEE Transactions on Computer*, vol. 20, no. 3, pp. 369–380, 2001.
- [11] Y. K. Malaiya, "Antirandom testing: getting the most out of black-box testing," in *Proceedings of the 6th IEEE International Symposium on Software Reliability Engineering (ISSRE '95)*, pp. 86–95, Toulouse, France, October 1995.
- [12] W. R. Hamming, "Error detecting and error correction codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

- [13] C. Gupte, "Antirandom testing of full-scan sequential circuits," M.S. thesis, Electrical and Computer Engineering Department, Colorado State University, Fort Collins, Colo, USA, 2004.
- [14] H. Yin, "Test data generation and evaluation for antirandom testing with checkpointing," M.S. thesis, Computer Science Department, Colorado State University, Fort Collins, Colo, USA, 1997.
- [15] H. Yin, Z. Lebne-Dengel, and Y. K. Malaiya, "Automatic test generation using checkpoint encoding and antirandom testing," in *Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE '97)*, pp. 84–95, Albuquerque, NM, USA, November 1997.
- [16] T. Chen, A. Bai, A. Hajjar, A. A. K. Andrews, and C. Anderson, "Fast anti-random (FAR) test generation to improve the quality of behavioral model verification," *Journal of Electronic Testing*, vol. 18, no. 6, pp. 583–594, 2002.
- [17] S. Xu and J. Gao, "An efficient random-like testing," in *Proceedings of the 7th Asian Test Symposium (ATS '98)*, pp. 504–508, Singapore, December 1998.
- [18] S. Xu and J. Chen, "Maximum distance testing," in *Proceedings of the 11th Asian Test Symposium (ATS '02)*, pp. 15–20, Hyatt Regency Guam, Guam, USA, November 2002.
- [19] D. K. Pradhan and M. Chatterjee, "GLFSR-a new test pattern generation for BIST," in *Proceedings of IEEE International Test Conference (ITC '94)*, pp. 481–490, Washington, DC, USA, October 1994.
- [20] S. Wu, Y. K. Malaiya, and A. P. Jayasumana, "Antirandom vs. pseudorandom testing," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '98)*, pp. 221–223, Austin, Tex, USA, October 1998.
- [21] C. Hall, B. Chess, T. Larrabee, and H. Manley, *The Nemesis Manual*, Computer Engineering, University of California, Santa Cruz, Calif, USA, 1995.
- [22] A. Jee, D. Dahle, C. Bazeghi, and F. J. Ferguson, *Carafe User's Manual*, Computer Engineering, University of California, Santa Cruz, Calif, USA, 1996.
- [23] S. Wu, "Effectiveness of antirandom and reed-solomon code based testing," M.S. thesis, Electrical Engineering Department, Colorado State University, Fort Collins, Colo, USA, 1998.
- [24] A. von Mayrhauser, T. Chen, A. Hajjar, A. Bai, and C. Anderson, "Fast antirandom (FAR) test generation," in *Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium (HASE '98)*, pp. 262–269, Washington, DC, USA, November 1998.
- [25] L. Nachman, K. Saluja, S. J. Upadhyaya, and R. Reuse, "A novel approach to random pattern testing of sequential circuits," *IEEE Transactions on Computers*, vol. 47, no. 1, pp. 129–134, 1998.
- [26] N. A. Toubia and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan-based BIST," in *Proceedings of IEEE International Test Conference (ITC '96)*, pp. 167–175, Washington, DC, USA, October 1996.