Antirandom vs. Psuedorandom Testing [*]

ShenHui Wu
Yashwant K. Malaiya
A.P. Jayasumana
Colorado State University
malaiya@cs.colostate.edu

ABSTRACT

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. This strategy results in a higher fault coverage when the number of vectors that are applied is limited. Results on several ISCAS benchmarks show this strategy to be very effective when a high fault coverage needs to be achieved with a limited number of test vectors. The superiority of the antirandom testing approach is even more significant for testing bridging faults.

1 Introduction

Available evidence suggests that random testing may be a reasonable choice for obtaining a moderate degree of confidence, however it becomes inefficient when only hard to test faults remain [2]. Random 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 tester 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. Here we present a new approach termed *antirandom* testing, since selection of each test explicitly depends on the tests already obtained.

2 Binary Antirandom Sequences

Antirandom testing [3] is a black-box strategy like psuedorandom testing, meaning that it assumes no information about the internal implementation of the circuit. **Antirandom test sequence (ATS)** is a test sequence such that a test t_i is chosen such that it satisfies some criterion with respect to all tests t_0, t_1, ... t_{i-1} applied before. **Distance** is a measure of how different two vectors t_i and t_j are. Here we use two measures of distance. **Hamming Distance (HD)** is the number of bits in which two binary vectors differ. It is not defined for vectors containing continuous values. **Cartesian Distance (CD)** between two vectors, $A = \{a_N, a_{N-1}, ...a_1, a_0\}$ and $B = \{b_N, b_{N-1}, ...b_1, a_0\}$ is given by:

[*]This work was supported in part by a BMDO funded project monitored by ONR

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

If all the variables in the two vectors are binary, then equation 1 can be written as:

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

Total Cartesian Distance (TCD) for any vector is the sum of its Cartesian distances with respect to all previous vectors. **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 to,t_1... t_{i-1} maximum, i.e.

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

is maximum for all possible choices of t_i. We will use Hamming distance and Cartesian distance to construct MHDATSs and MCDATSs.

If testing is less than exhaustive, then MDAT (maximum distance antirandom testing) 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.

Example 1: Construction of a MHDATS (MCDATS): For a system, the inputs {x,y,z} can be either 0 or 1. 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 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 THD(t_1) = 3. At this point, the situation is shown in Fig. 1a, where the input combinations already chosen are marked.

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

t_0= {0,0,0}
t_1= {1,1,1}; THD = 3
t_2 = {0,1,0}; THD = 3 *or* t_2 = {1,0,1}; THD = 3

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

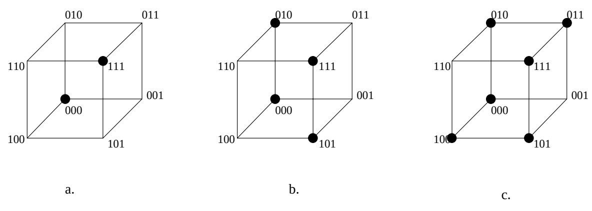

Figure 1: Construction of 3-bit MHDATS

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 Fig. 1c. Let us choose t6 = {1,1,0} which leaves t_7 = {0,0,1}. The complete MHDTS obtained here is given as sequence 1 in Table 1. □

Table 1: 3-bit MHDTS (Example 3)

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

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

Here we consider generation and use of MCDATSs. They satisfy a more strict criterion that automatically satisfies the MHDATS requirements. Once a complete MCDATS of (n-1) bits is available, an n-bit sequence can be obtained using the following two procedures.

Procedure 1. Expansion of MHDATS (MCDATS):
Step 1. Start with a complete MHDATS of N variables, $X_{N-1}, X_{N-2}, \ldots X_1, X_0$.
Step 2. For each vector t_i, i = 0, 1, ... (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 previous vectors. □

Antirandom	Pseudo 0
00000000000000	00000000000000
11111111111111	10000000000000
01010101010101	11000000000000
10101010101010	11100000000000
00110011011000	11110000000000
11001100100111	11111000000000
01100110001101	11111100000000
10011001110010	11111110000000
00011110010011	11111111000000
11100001101100	11111111100000

Table 2: Comparison of randomness: Antirandom vs. Pseudorandom

Procedure 2. Expansion and Unfolding of a MHDATS (MCDATS):
Step 0. Start with a complete (N-1) variable MHDATS (MCDATS) with 2^{N-1} vectors.
Step 1. Expand by adding a variable using Procedure 2. We now have the first ($2^N/2$) vectors needed.
Step 2. Complement one of the columns and append the resulting vectors to first set of vectors obtained in Step 1. Here, it would be convenient to complement the variable added in Step 1. □

The above procedures have been implemented in a program called ATG. It generates MCDATSs which are also MHDATSs. The application of antirandom testing for software has been reported in [6].

A scheme can be considered to be more *random* if the ones and the zeros are evenly distributed in space and time and if there is very little correlation between one pattern and the next. Let us compare the randomness of the first 10 test patterns for 14-input sequences generated using antirandom property, and conventional pseudo-random tests, as given in Table 2. Pseudo 0 represent the pseudorandom sequences starting with the seed 00000000000000. The successive vectors in the time sequence are listed sequentially. Table 2, shows that the antirandom sequence is more random than the the pseudorandom sequences. There are several formal tests for randomness. Pradhan and Chatterjee [4] have shown that LFSR based sequences fail most of these tests.

3 Effectiveness of Antirandom and Psuedo-random Testing

We have measured the effectiveness of a test set using stuck-at and bridging fault coverage measures. We have used four benchmark circuits C880, C3540, C1355 and C499 for evaluations. The results for C880 are presented here.

Figures 2 shows the stuck-at fault coverage obtained for the circuits C880. The x-axis represents the number of the test patterns, and the y-axis fault coverage. Tables 4 shows the fault coverage for different number of test patterns for c880. 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 Figure 2, we observe that antirandom tests obtaining 91.3% coverage for 105 vectors. The fault coverage curves rise sharply and exhibits a smooth behavior. In case of psuedo-random tests, there is a significant difference depending on the initial seed, the coverage obtained with 105 vectors ranges from 51.06% to 73.9%. For all three seeds, psuedo-random tests significantly lag in performance compared with antirandom tests. We also observe that the plots for psuedo-random tests show somewhat irregular growth.

The results show that the antirandom sequences generally provide higher coverage than psuedo-random testing. Both new test pattern generation schemes can obtain a high fault coverage with significantly fewer test patterns compared to pseudorandom testing. Often antirandom sequences obtain similar coverage values, antirandom testing is generally slightly better.

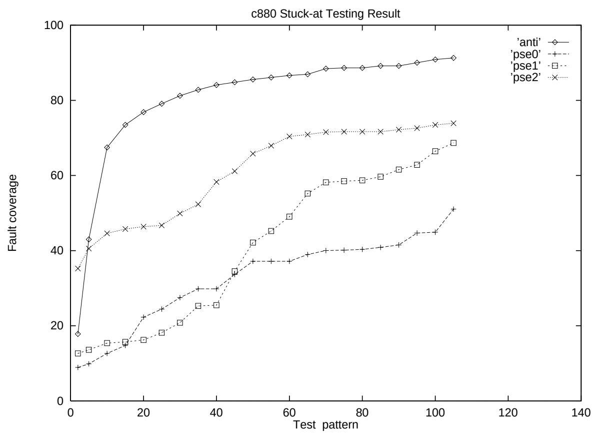

Figure 2: C880 Stuck-at Fault Coverage Simulation

The bridging faults are identified by Carafe by considering the layout information. Nemesis assumes that a bridging fault is being tested in the IDDQ test environment. Figures 3 for c880 show that the difference between the proposed approache and the traditional psuedo-random testing for bridging faults is quite remarkable. With antirandom sequence, the coverage of bridging faults rises much faster than that for stuck-at faults. The gap between the curves for the new approaches and psuedo-random testing is wider for bridging faults. The same behavior is observed for other three circuits. For about 88-90% coverage obtained by our approaches, the gap for c880 widens from 17-49% to 27-75%.

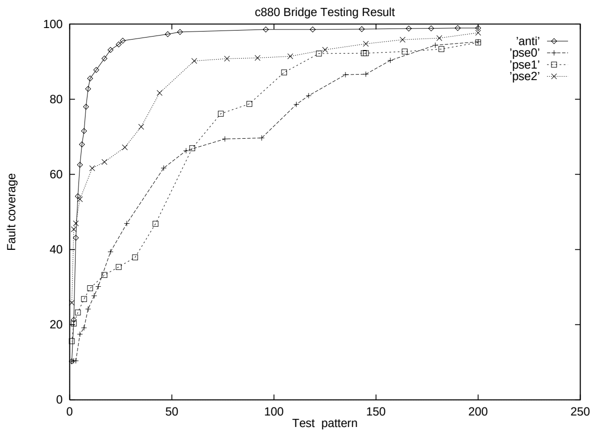

Figure 3: C880 Bridging Fault Coverage

4 Conclusions

Here we have demonstrated that antirandom testing can achieve high fault coverage much faster than the conventional psuedo-random testing. It has also been successfully applied for software testing and testing of VHDL descriptions [5]. Its effectiveness is specially remarkable for bridging faults. The scheme is well suited for IDDQ tsting because it provides very good coverage with only a few vectors.

References

[1] *The Nemesis Manual*, C. Hall, B. Chess, T. Larrabee and H. Manley, Computer Engineering, University of California, Santa Cruze, 1995.

[2] Y.K. Malaiya and S. Yang, "The Coverage Problem for Random Testing," *Proc. International Test Conference*, October 1984, pp. 237-245.

[3] Y. K. Malaiya, "Antirandom Testing: Getting the most out of black-box testing," *Proc. International Symposium On Software Reliability Engineering*, Oct. 1995, pp. 86-95.

[4] D. K. Pradhan and M. Chatterjee, "GLFSR-A New Test Pattern Generation for BIST", *Proc. International Test Conference*, 1994 pp. 481-490.

[5] A. vonMayrhauser, A. Bai, T. Chen, A. Hajjar and C. Anderson, "Fast Antirandom (FAR) Test Generation to Improve Code Coverage" *Proc. International Software Quality Week*, May 1998.

[6] H. Yin, Z. Lebne-Denge and Y. K. Malaiya, "Automatic Test Generation using Checkpoint Encoding and Antirandom Testing" *Int. Symp. on Software Reliability Engineering*, 1997, pp. 84-95.