International Journal of Engineering Sciences Paradigms and Researches (IJESPR)
(Vol. 31, Issue 01) and (Publishing Month: June 2016)
(An Indexed, Referred and Impact Factor Journal)
ISSN: 2319-6564
www.ijesonline.com

Implementation of Random Algorithm in Cellula Automata (CA) Matrices

Emad Aldeen Widaat Alla Musa[1] and Dr. Abdelrahman Elsharif Karrar[2]

[1]Lecurer, Faculty of Science and Humanities Studies-Computer Science Department,
Prince Sattam Bin Abdul-Azziz University, Saudi Arabia
e.musa@psau.edu.sa

[2]Associate Professor, College of Computer Science and Engineering,
Taibah University, Saudi Arabia
akarrar@taibahu.edu.sa

Publishing Date: June 08, 2016

Abstract

The implementation of random algorithm strategy through multiple industries, products with different levels and variation of some industrial products for purely marketing purposes and diversity of its quality of scientifically Academy according to standard specifications, as well as product differentiation between different countries needs hard focusing on discovery standard range for the specifications. Random algorithm requires knowledge of standard specifications in easier way and less expensive cost. The means and technicians to detect standard specifications needs to be accessible and affordable to explain the matching specifications for use in all countries of the world through the use of specific optimal strategy. The optimal use of random algorithm through defining specific steps and basic procedures in analysis of standard specifications by appropriate priorities still needs more studies to improve random algorithm quality to produce better test vector that cover a high range of faults that would compared to some of well known TPG methods such as anti random. The study of different approach in random algorithm will lead to achieve the purposes for which it was designed to detect the standard quality for any product.

Keywords: *Randomness, Random algorithm, Cellula Automata*.

1. Introduction

Research problem is focusing on defining random algorithm methods to fulfill the needs for researchers to analyze research gap in random algorithm methods and how to manipulate random algorithm in different scientific areas to achieve better result. the feature and specification of better random algorithm in terms of technology and size and preferential features can be analyzed in number of points:

1. Producing, organizing and analyzing binary random algorithm in random matrix method
2. Defining random algorithm processing to produce Cellula Automata (CA) matrix based on random algorithm
3. Testing the effectiveness of random algorithm matrix
4. Ensure test effectiveness for random algorithm matrix

2. Research Objective

The objective of this project is:

1. Evaluating random algorithm strategy by implementing the random number in Cellula Automata (CA)
2. Evaluate random algorithm according to standard matrix models in solving scientific problems.

3. Simplify and facilitate the efficiency of random algorithm.

3. Research Contribution

The significance of research comes through defining strengths and weakness syndrome for random algorithm which needs further study and research. The implementation of random algorithm strategy is an interest to a number of global academic institutions and professional electronics factories as Intel Corporation which currently started studying the effectiveness of algorithms in IC testing with high economic feasibility for achieving the best ratio in possible number of errors that can occur in computer circuits as stated by Gary Bradski director of automated education group within the laboratory systems technology, as well as the attention of technological institutes and local and global training centers in different scientific topics.

4. Research Hypotheses

1. Optimization the use of random algorithm leads to efficiency in performance of random matrix (Cellula Automata)
2. There is a relation between the size of consistency in algorithm and consistency of random matrices.
3. The appropriate testing of random algorithm leads to predict the positive and negative results of random matrices.

5. Research Methodology

The research follow scientific research methods form in terms of analyzing and addressing integrated circuits to study the subject and also deductive approach to develop hypotheses on scientific
The methodology is to build an effective method to test the efficiency of random matrices, the flowchart below define the methodology:

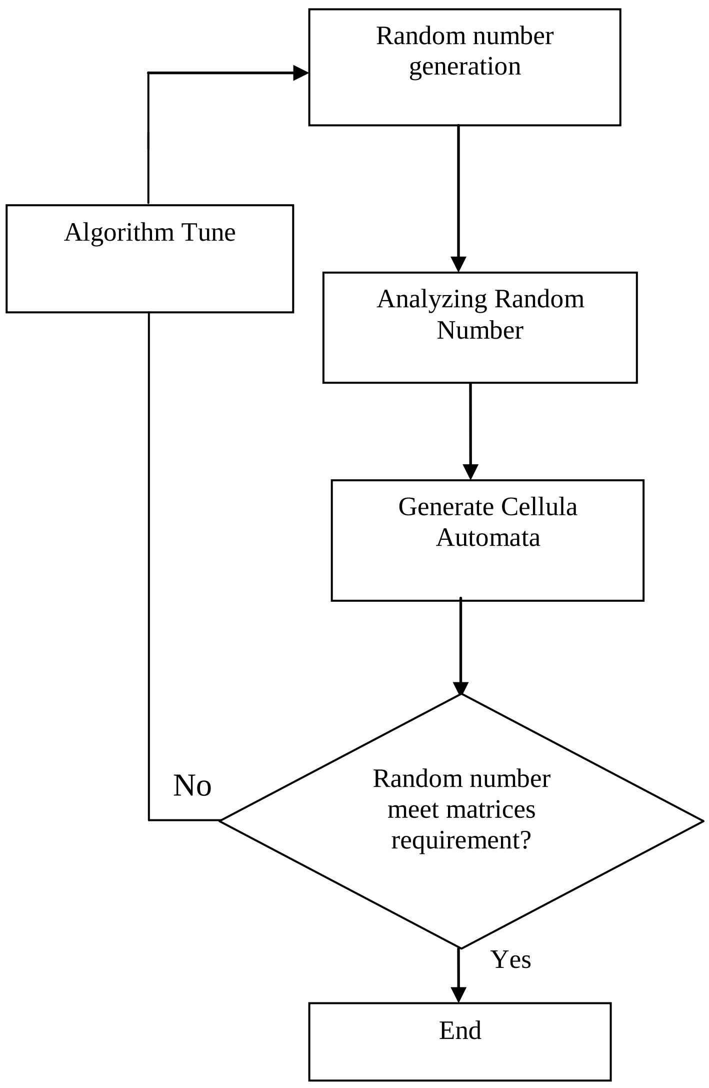


Figure 1: Algorithm Hypothesis Flowchart

6. Randomness Definition

Randomness is the situation where clear and specific patterns of absent behavior make the next step cannot be predict. This assumes that there is a random external environment affecting the system (a random input).

There are several ways in which a process or system can be seen as random:

1. Randomness coming from the environment (Brownian motion , hardware random number generators)
2. Randomness coming from the starting conditions. This aspect is studied by chaos theory.
3. Randomness generated by the system itself. This is also called pseudo-randomness, and is the kind used in pseudo-random number generators. There are many algorithms (based on arithmetic or cellular automata) to generate pseudo-random numbers. The behaviour of such a system can be predicted, if the random seed and the algorithm are known. These methods are quicker than getting "true" randomness from the environment.

6.1 Importance of Randomness

The early part of the 20th century saw a rapid growth in the formal analysis of randomness, as various approaches to the mathematical foundations of probability were introduced. In the mid- to late-20th century, ideas of algorithmic information theory introduced new dimensions to the field via the concept of algorithmic randomness.
Although randomness had often been viewed as an obstacle and a nuisance for many centuries, in the 20th century computer scientists began to realize that the deliberate introduction of randomness into computations can be an effective tool for designing better algorithms, in some cases such randomized algorithms outperform the best deterministic methods.

7. Random Number Definition

Random number is the number that was chosen by a random process and it’s used to generate simulated probability distributions values. There are many ways to generate random numbers as linear congruence, and the use of random number tables, or using specific functions such as Rand function in Matlab programming languages.

7.1 Importance of Random Number

Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena and for selecting random samples from larger data sets. When discussing single numbers, a random number is one that is drawn from a set of possible values, when discussing a sequence of random numbers, each number drawn must be statistically independent of the others.
With the advent of computers, programmers recognized the need for a means of introducing randomness into a computer program.

7.2 Random Number Characteristics

Random number can be generated by more than (Law, algorithm, method) and whenever the law is more complicated the number will be closest to a random.
There are two main approaches to generating random numbers using a computer: Pseudo-Random Number Generators (PRNGs) and True Random Number Generators (TRNGs) and each has its advantage and disadvantage.

7.2.1 Pseudo-Random Number Generators (PRNGs)

As the word ‘pseudo’ suggests, pseudo-random numbers are not random in the way you might expect, PRNGs are basically the algorithms that use mathematical equations to produce strings of digits that appear randomly. Modern algorithms which used to generate pseudo-random numbers it gives good numbers look exactly as if they were truly random.

7.2.2 True Random Number Generators (TRNGs)

In comparison with PRNGs, TRNGs extract randomness from physical phenomena and introduce it into a computer.
The characteristics of TRNGs are quite different from PRNGs. First, TRNGs are generally rather inefficient compared to PRNGs,

taking considerably longer time to produce numbers.

They are also nondeterministic, meaning that a given sequence of numbers cannot be reproduced, although the same sequence may of course occur several times by chance. TRNGs have no period.

Random numbers can be generated by different programming language as Matlab and C++, but Matlab code is more easier and shorter than C++ and Matlab code can be executed without compiler because once the Matlab code have been designed, there will not be much left for the programmer to improve the performance of Matlab code so it is better to use Matlab language to improve algorithms, Random numbers in MATLAB comes mainly from the disciples (Rand , randi and randn).

Many other disciples call these three disciples basically, but these disciples are considered to be the basic to produce random number and they are based on one random number generator which can be controlled by using RNG function.

7.3 Random number Table

Random number tables is constructed from (0,1,2,…9) numbers which built in random order in rows and columns, the table below shows random numbers

Table 1: Random Numbers Model Table

38	71	81	39	18	24	33	94	56	48	80	95	52	63	01
27	29	03	62	76	85	37	00	44	11	07	61	17	26	87
34	24	23	64	18	79	80	33	98	94	56	23	17	05	96
32	44	31	87	37	41	18	38	01	71	19	42	52	78	80
41	88	20	11	60	81	02	15	09	49	96	38	27	07	74
95	65	36	89	80	51	03	64	87	19	06	09	53	69	37
77	66	74	33	70	97	79	01	19	44	06	64	39	70	63
54	55	22	17	35	56	66	38	15	50	77	94	08	46	57
33	95	06	68	60	97	09	45	44	60	60	07	49	98	78
83	48	36	10	11	70	07	00	66	50	51	93	19	56	45
34	35	86	77	88	40	03	63	36	35	73	39	44	06	51
58	35	66	95	48	56	17	04	44	99	79	87	85	01	73
98	48	03	63	53	58	03	87	97	57	16	38	46	55	96
83	12	51	88	33	98	68	72	79	69	88	41	71	55	85
56	66	06	69	44	70	43	49	35	46	98	61	17	63	14
68	07	59	51	48	87	74	79	19	76	46	68	50	55	01
20	11	75	63	05	16	96	95	66	00	18	86	66	67	54
26	56	75	77	75	69	93	54	47	39	67	49	56	96	94
26	45	74	77	74	55	92	43	37	80	76	31	03	48	40
73	39	44	06	59	48	48	99	72	90	88	96	49	09	57
34	36	64	17	21	39	09	97	33	34	40	99	36	12	12
20	32	06	40	37	02	11	83	79	28	38	49	44	84	94
04	52	85	62	24	76	53	83	52	05	14	14	49	19	94
33	93	35	91	24	92	47	57	23	06	33	56	07	94	98
16	09	10	97	86	31	45	96	33	83	77	28	14	40	53

8. Random Number Algorithm Definition

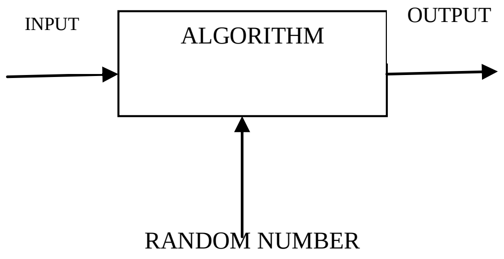
Random number algorithm is an algorithm that generate a sequence of numbers who's statistical properties are such that the sequence is indistinguishable from a true random sequence , of course, it is technically impossible to generate a random sequence ,since by definition , a random process is not deterministic, therefore, some people prefer use the term pseudorandom to describe a sequence of numbers that are generated by a computer and that behave similarly to a random variants from a specified distribution and some others define the random number algorithm as it's the algorithm which uses a random number at least once during the computation to make a decision as when trying to factor a large number by choosing random numbers as possible divisors.

8.1 Importance of Random Algorithm

Very often, the definition of random is enough to turn a simple deterministic algorithm with worst cases of behavior to a random algorithm performs well on each possible input. The mechanism of deterministic algorithm is that the solution produced by the algorithm is correct, and the number of computational steps is same for different runs of the algorithm with the same input.

INPUT
ALGORITHM
OUTPUT

By implementing a computational problem in deterministic algorithm, it may be difficult to formulate an algorithm with good running time, therefore, the solution is to use Random algorithm.



In addition to the input, the algorithm uses a source of random numbers during execution, it takes random choices depending on those random numbers so the behavior (output) can vary if the algorithm is run multiple times on the same input.[2]

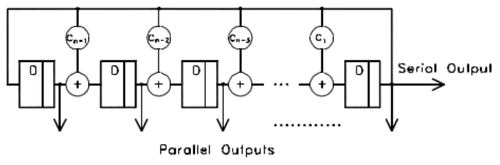
8.2 Advantage of Randomized Algorithm

1 The algorithm is usually simple and easy to implement.
2 The algorithm is fast with very high probability, and/or It produces optimum output with very high probability.

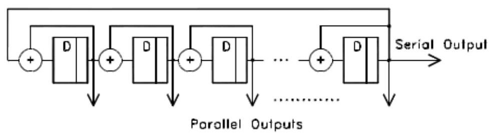
In many cases the run time for random algorithm is better than deterministic algorithm.

9. Cellular Automata (CA)

Cellular automata are sequential structures similar to LFSRs. Their periods are often shorter, but code words generated by CA are sometimes more suitable for test patterns with preferred numbers of one's or zeros at the outputs.



(a) LFSR structure



(b) CA structure

Figure2: LFSR and Cellula Automata Architecture

10. Literature review

10.1 Antirandom Testing Algorithm

Random testing is one form of the black-box testing strategy, which means no any circuit structure information needed. This technique depends on Hamming Distance between factors, the vector with the higher distance is going to be chosen as new test vector.

Definition 1. Distance is a measure of how different two vectors ti and tj are. Here we use two measures of distance defined below.
Definition 2. Hamming distance (HD) is the number of bits in which two binary vectors or patterns differ. It is not defined for vectors containing continuous values.

$$\mathbf{HD} = \sum_{j=0}^{i-1} \mathbf{HD}[\mathbf{t_i}, \mathbf{t_j}] \quad (1)$$

Definition 3.Total Hamming Distance (THD):Total hamming distance is defined as Sum of hamming distance between test vectors in the sequence. Let hamming distance d[ti, tj] be the total number of changes between ith the and jth test vector , the Total Hamming Distance(THD) for the whole test vector set is calculated by the following relation.

$$\mathbf{THD} = \sum_{i=1}^{n-1} \mathbf{d}[\mathbf{t_i}, \mathbf{t_{i+1}}] \qquad (2)$$

Where n represents total number of test vectors in the whole set[6].

Definition 4. Maximal distance antirandom test sequence (MDATS) is a test sequence such that each test ti is chosen to make the total distance between ti and each of t0, t1, . . . , ti−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 ti[4]

Definition 5. Maximal Hamming distance antirandom test sequence (MHDATS) a MDATS that uses Hamming distance as the distance measure.

The construction of a MHDATS , for a system with three input using this procedure algorithm work in three steps which:

Step 1: Assign an arbitrarily chosen value to obtain the first test vector.

Step 2 : To obtain each new vector, evaluate the THD 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.

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

To construct MHDATS In case of system with three Input (X,Y,Z) If we start with 000 as T0, then the next one is 111 (THD=3) max.

assume that {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}. The fourth member of the test set depends upon the difference function used. If we use Hamming distance:

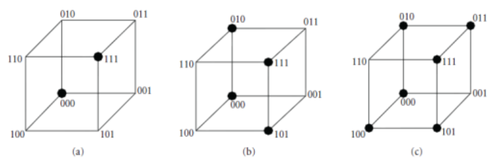


Figure 3: 3-Bit MHDATS

Table 2: THD Calculations

	000	111	001	THD
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

10.2 Scalable Test Pattern Generation (STPG) Algorithm

Another approach to generate antirandom tests, 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.

TPG algorithm works as following:

1. Initialize the input value and assign an adding factor
2. To obtain a maximum distance between two test patterns, T0 and T1, complement the first pattern, to become the second test pattern, T1, to obtain third test pattern, T2, use the adding factor and add it to the first test pattern, T0, to generate third sequence, T2.
3. Repeat step 2 and step 3 to generate other pattern

Table 3: STPG Test Pattern Calculation

Test vector
T0 = 000
T1 = 111 Complement of T0
T2 = T0+010 = 000 + 010 = 010 (010 is the adding factor)
T3 =101 Complement of T2
T4 = T2+010 = 010 + 010 + 100
T5 = 011 Complement of T4
T6 = T4+010 = 100 + 010 = 110
T7 = 001 Complement T6

10.3 Adaptive Random Testing Algorithm:

Is one of the testing methods that use a related concept of "distance" to generate test cases, the main algorithm is the Fixed Size Candidate Set ART (Adaptive Random Testing) algorithm (FSCS-ART).

The algorithm can be divided into two steps:

– First, a set of k candidates ci 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' ci

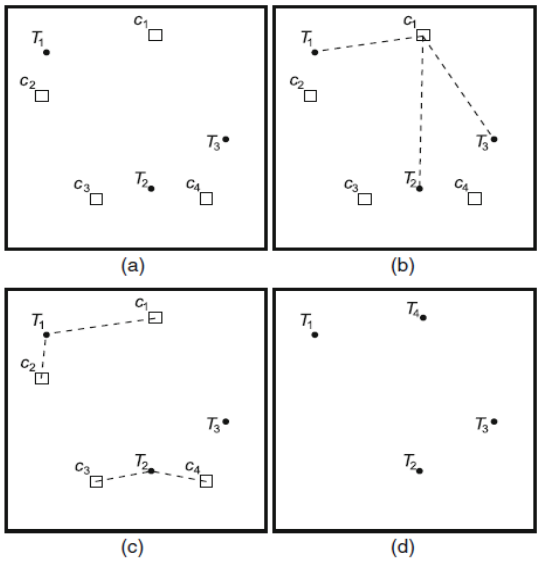


Figure 4: Tests Cases T and Candidates CI Selection

As we see c1 is candidate with the largest distance and it has been treated as new test case. Adaptive Random testing allows us to skip the counting of the whole test sequence , this method needs still a lot of computations of distances for each potential test pattern.

10.4 Iterative Antirandom Testing Algorithm

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
Maximal Minimal Hamming Distance (MMHD) between two test patterns Ti and Tj is used instead of Hamming Distance or Cartesian Distance. This allows to construct optimal antirandom test with restricted number of patterns.
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:

$$T_0 = t_{i,N-1}, t_{i,N-2}, \ldots, t_{i,N/2}, t_{i,N/2-1}, \ldots, t_{i,2}, t_{i,1}, t_{i,0}$$
$$T_1 = \overline{t_{i,N-1}}, \overline{t_{i,N-2}}, \ldots, \overline{t_{i,N/2}}, \overline{t_{i,N/2-1}}, \ldots, \overline{t_{i,2}}, \overline{t_{i,1}}, \overline{t_{i,0}}$$
$$T_2 = \overline{t_{i,N-1}}, \overline{t_{i,N-2}}, \ldots, \overline{t_{i,N/2}}, t_{i,N/2-1}, \ldots, t_{i,2}, t_{i,1}, t_{i,0}.$$

----------- (3)

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:

$$T_0 = t_{i,N-1}, \ldots, t_{i,2N/3}, t_{i,2N/3-1}, \ldots, t_{i,N/3}, t_{i,N/3-1}, \ldots, t_{i,1}, t_{i,0}$$
$$T_1 = \overline{t_{i,N-1}}, \ldots, \overline{t_{i,2N/3}}, \overline{t_{i,2N/3-1}}, \ldots, \overline{t_{i,N/3}}, t_{i,N/3-1}, \ldots, t_{i,1}, t_{i,0}$$
$$T_2 = t_{i,N-1}, \ldots, t_{i,2N/3}, \overline{t_{i,2N/3-1}}, \ldots, \overline{t_{i,N/3}}, \overline{t_{i,N/3-1}}, \ldots, \overline{t_{i,1}}, \overline{t_{i,0}}.$$

----

(4)

Iterative test based on standard patterns (q = 4).In each iteration four patterns (T0, T1, T2, T3) are applied. T0 is a random pattern, T1 is negation of T0, T2 is a pattern where HD(T0, T2) = N/2, T3 is negation of T2.

The algorithm for q = 4 is shown below

$$T_0 = t_{i,N-1}, \ldots, t_{i,2N/3}, t_{i,2N/3-1}, \ldots, t_{i,N/3}, t_{i,N/3-1}, \ldots, t_{i,1}, t_{i,0}$$
$$T_1 = \overline{t_{i,N-1}}, \ldots, \overline{t_{i,2N/3}}, \overline{t_{i,2N/3-1}}, \ldots, \overline{t_{i,N/3}}, t_{i,N/3-1}, \ldots, t_{i,1}, t_{i,0}$$
$$T_2 = t_{i,N-1}, \ldots, t_{i,2N/3}, \overline{t_{i,2N/3-1}}, \ldots, \overline{t_{i,N/3}}, \overline{t_{i,N/3-1}}, \ldots, \overline{t_{i,1}}, \overline{t_{i,0}}$$
$$T_3 = \overline{t_{i,N-1}}, \ldots, \overline{t_{i,2N/3}}, t_{i,2N/3-1}, \ldots, t_{i,N/3}, \overline{t_{i,N/3-1}}, \ldots, \overline{t_{i,1}}, \overline{t_{i,0}}$$

------(5)

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 (3), (4) and (5). The simplest solution is the iterative test with two patterns per iteration. The first pattern T0 is the random one and the second pattern T1 is the negation of the T0.

Table 4: Standard Set of Pattern 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

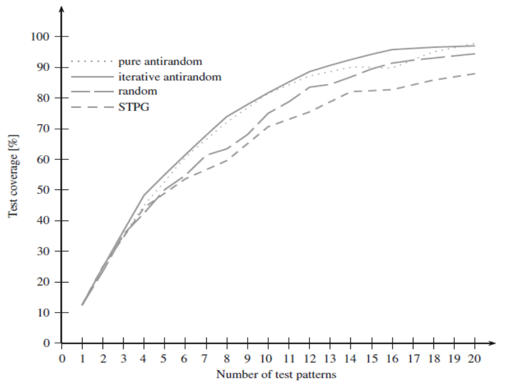


Figure 5: Comparison between Different Algorithms

This figure show Comparison between different algorithm, as we can see iterative antirandom algorithm gave high fault coverage compare to other algorithms.

11. Experiment Result

There are several rules for CA depending on which gates used in CA (XOR or XNOR), it's been known for some time that certain composite automata constructed from rule 90 and rule 150 are reversible.
The construction fellow two main rules of CA which are:

CA_rule 90 equations:

$$q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$$

CA_rule 150 equations:

$$q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t) \oplus q_{i}(t)$$

Suppose that, a random sequence of 15 bits has been generated as:

X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14

The CA sequence is shorter than random sequence so by dividing the random sequences in a group of three bits the consistency can be proved as:

$$[X0\ X1\ X2]\ [X3\ X4\ X5]\ [X6\ X7\ X8]$$

$$[X9\ X10\ X11]\ [X12\ X13\ X14]$$

And by XORing every 3 bits of the random sequences to generate CA rule 150 then a new sequence (CA rule 150) is constructed:

Y0 = $X_0 \oplus X_1 \oplus X_2$

Y1 = X3$\oplus$ X4$\oplus$X5

Y2= X6$\oplus$ X7$\oplus$ X8

Y3= X9$\oplus$ X10$\oplus$ X11

Y4= X12$\oplus$ X13$\oplus$ X14

Table 5: Random Number Matrix

Antirandom														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	1	1	0	0	1	1	0	0	1	0	0	1	1	1
0	0	1	1	0	0	1	1	0	0	0	1	1	0	1
0	1	0	0	1	1	0	0	1	1	1	0	0	1	0
0	0	0	0	1	1	1	1	0	0	1	0	0	1	1
0	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Then by implementing CA_rule 90 equations on the same random number matrix, CA rule 90 is generated as stated in table 6 below where it show CA matrices for rule 90 and 150.

Table 6: Cellular Automata Rule (90&150) Matrices

CA rule 90	CA rule 150
0 0 0 0 0	0 0 0 0 0
1 0 0 0 0	0 1 1 1 1
1 0 0 0 0	1 1 0 1 0
0 0 0 0 0	1 0 1 0 0
0 1 1 1 0	0 0 0 0 0
1 1 1 1 0	0 1 1 1 1
1 1 1 1 0	1 1 0 1 0
0 1 1 1 0	1 0 1 0 1
0 1 1 0 1	0 0 0 1 0
1 1 1 0 1	0 1 1 0 1

To study the efficiency performance of CA matrices there's a need to calculate the Total Hamming Distance (THD) for each of the CA matrix.

Table 7: THD for CA Rule 90 and 150

CA rule 90	THD CA rule 90	CA rule 150	THD CA rule 150
0 0 0 0 0	0	0 0 0 0 0	0
1 0 0 0 0	1	0 1 1 1 1	4
1 0 0 0 0	2	1 1 0 1 0	6
0 0 0 0 0	2	1 0 1 0 0	9
0 1 1 1 0	14	0 0 0 0 0	9
1 1 1 1 0	15	0 1 1 1 1	15
1 1 1 1 0	15	1 1 0 1 0	15
0 1 1 1 0	16	1 0 1 0 1	21
0 1 1 0 1	24	0 0 0 1 0	19
1 1 1 0 1	25	0 1 1 0 1	25

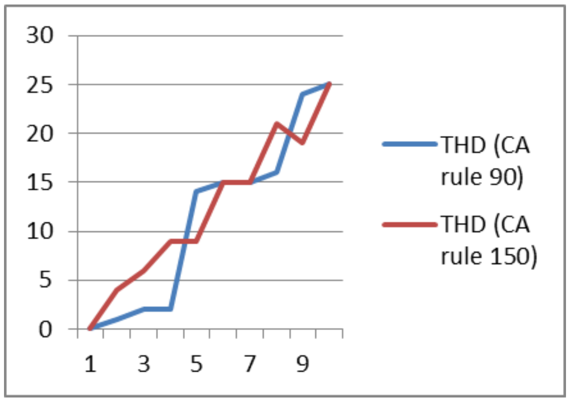


Figure 6: THD for Cellula Automata (rule 90 and 150)

From table 7 and graph 3 the THD which generated from CA (rule 150) has much better distance value than THD from CA (rule90) mostly which can be implemented in different scientific application such as in circuit testing where the higher distance between vectors play an import rule in fault coverage.

12. Conclusion

The idea of turning random numbers, which generated by algorithms, to a matrices leads to facilitate the work of the researcher and reduce the time spent in deliveries. The matrix quality depends on the type of the experiment to be implemented in.
A random matrix needs further study and revision to generate more efficient and flexible matrices in order to enrich the scientific research in all scientific fields.

References

[1] Dr Mads Haahr, Introduction to Randomness and Random Numbers, School of Computer Science and Statistics, Trinity College Dublin 2, Ireland.

[2] Subhas C. Nandy “Introduction to Randomized Algorithms, Advanced Computing and Microelectronics Unit , Indian Statistical Institute, Kolkata 700108, India

[3] s. mourad,t j. l. a. hughes and e. j. mccluskey, Effectiveness of Single Fault Tests to Detect Multiple Faults in Parity Trees ,Center for Reliable Computing, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305, U.S.A.

[4] Tanveer Ahmed, Liakot Ali , Evaluation of Fibonacci Test Pattern Generator for Cost Effective IC Testing, Journal of Computer Engineering and Informatics(JCEI), Vol. 2 Iss. 2, pp 28-36, Apr. 2014

[5] Elcheleburger E.B., and Lindoboloome, ‘Random-Pattern coverage enhancement and diagnostics for LSSD self-test”, volume 27 Issue 3, pp 265-272 May 1983

[6] Shen HuiWu, Sridhar Jandhyala, Yashwant K. Malaiya, and Anura P. Jayasumana,” Antirandom Testing A Distance-Based Approach”. VLSI design, Yverdon, Switzerland, 2008.

[7] Malaiya, Y.K. ,” Antirandom testing: getting the most out of black-box testing”,Software Reliability Engineering, Sixth International Symposium on. ISSRE ’95, pp 86–95. IEEE Computer Society 1995.

[8] Praveen Kumar Aggarwal, Vandana Yadav, Dr. Arti Noor, "DFT (Design for Testability) Pattern Generation Task for Circuit Under Test",International Journal of Engineering Research and Applications (IJERA) - Vol. 1, Issue 2, pp.190-193, volume 1- issue 2, July-Aug 2011

[9] Ireneusz Mrozek , Vyacheslav N. Yarmolik," Iterative Antirandom Testing",

[10] Journal of electronic Testing: theory and applications, Volume 28 Issue 3, June 2012

[11] DavidH. K. Hoe, JonathanM. Comer, JuanC. Cerda,Chris D.Martinez, andMukul V. Shirvaikar Cellular Automata-Based Parallel Random Number Generators Using FPGAs

[12] Ireneusz Mrozek , Yarmolik ,Antirandom Test Vectors for BIST in Hardware/Software Systems, Journal Fundamental Informaticae archive , Volume 119 Issue 2, Pages 163-185 ,April 2012

[13] Hortensius, P.D. , Cellular automata circuits for built-in self-test, IBM Journal of Research and Development (Volume: 34, Issue: 2.3), Date of Current Version :06 April 2010

[14] Stavros Athanassopoulo , Christos Kaklamanis, Gerasimos Kalfoutzos, Evi Papaioannou, Cellular Automata: Simulations Using Matlab, The Sixth International Conference on Digital Society 2012,