Sahar Z. Alawey
Ministry of Higher Education and Scientific Research, Baghdad, Iraq.
sahar_eng2006@yahoo.com
office@rpd-mohesr.edu.iq

Received on: 19/03/2017
Accepted on: 24/05/2018
Published on line: 25/09/2018

A Mixture between Rule 90 and Rule 150 Cellular Automata as a Test Pattern Generator

Abstract- *Built-in Self-test (BIST) is one of integrated circuit (IC) testing techniques that can be used as a pseudo-random generator for the Circuit Under Test (CUT). This paper introduces the design and simulation of a 4-bit test pattern generator (TPG) using a one dimension Linear Hybrid Cellular Automata (LHCA) with a mixture of rule 90 and 150. LHCA is an enhancement from Linear Feedback Shift Register (LFSR) which can have more random test vectors and improving the cycle length. Design and simulation have been performed using Quartus II and Model Sim software.*

Keywords- *Built-in Self-Test, Cellular Automata, Linear Hybrid Cellular Automata, Pseudo-random Generator, Test Pattern Generator.*

How to cite this article: S.Z. Alawey, "A Mixture between Rule 90 and Rule 150 Cellular Automata as a Test Pattern Generator," *Engineering and Technology Journal*, Vol. 36, Part A, No. 9, pp. 951-956, 2018.

1. Introduction

Built-In Self-Test (BIST) recognized as the preferable and widespread settlement for the case of the huge rising in test data volumes. Moreover, BIST is required for a complex testing in System-on-Chip (SoC) design. Pseudo random pattern testing considered as the extreme economical BIST schemes [1,2]. Figure 1 shows the implementation of BIST. There are two main procedures to design a BIST, Test Pattern Generator (TPG) and Signature Analyser (SA). BIST introduces a test pattern generation and it response to the output on-chip itself. A pattern generator is practically an independent Finite-State Machine (FSM). Furthermore, it is exactly represented as a shift register with feedback connections. The generator is considered linear if the design used only exclusive OR (XOR) gates in order to implement its feedback circuit. Nonetheless, it is known as a non-linear generator [3]. Linear Feedback Shift Registers (LFSR) and Cellular Automata (CA) are the most frequently used as pseudo-random pattern generators. Both of them are Linear Finite State Machines (LFSM) [4]. Semiconductors and electronics industry focus more on CA as a test pattern design than other methods because CA can provide more random test vectors [1]. Linear Hybrid Cellular Automata (LHCA) is one of the CA design structure that has different rules in every single cell. LHCA can be considered as the simplest one-dimensional CA design structure compared to Uniform Cellular Automata (UCA). It also can generate more randomized test patterns than LFSR and UCA does [5]. The ability to produce maximum length cycles is because of no-shift-induced correlation between its test vector bit values. In this paper, the construction of a 4 bit test pattern generator using LHCA can be implemented through Field Programmable Gate Array (FPGA) by using simple logic blocks with an appropriate rendering for BIST. To generate the random numbers, a vast diversity of original methods has been used by the designer. Test Pattern Generators can be one of the following: counter, ROM, XOR tree, LFSR and CA [1,6].
Generally, the sequence of numbers can be considered random in the following cases:

- If there is no generic pattern can be comprehended.
- No prognostication can be made on this sequence.
- No characterization can be found.

Yet, on related applications, random numbers are produced by the consecutive repetition of a confirmed transformation.

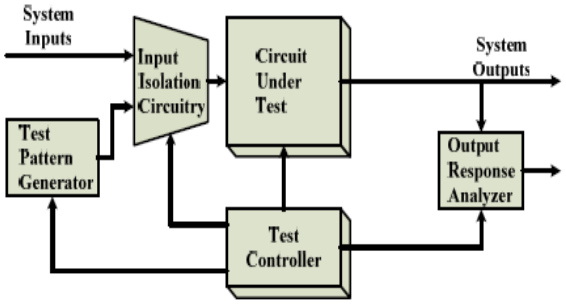

Figure 1: BIST implementation

2. General Concept of Cellular Automata

https://doi.org/10.30684/etj.36.9A.2
2412-0758/University of Technology-Iraq, Baghdad, Iraq

Cellular Automata (CA) was firstly introduced by the scientists Von Neuman and Stanisly Wlam who found the cellular spaces of CA. Then Stephen Wolfram introduced his book "A New Kind of Science" [5]. Since that time the researchers were motivated to go deeply in studying the structure of CA. Cellular Automata is a special type of dynamical systems which can describe the development of compound systems using modest rules, dispensing the use of partial differential equations [7]. CA develops in discrete procedure, where the next value (state) of the cell is found by its previous values and that is why the set of cells called the neighbour cells. Depending on many factors, like the dimensions of CA, the range of the neighbourhood can change. Based on its neighbour cells at a specific time, the state of each cell is changed simultaneously. The algorithm that used to find the next cell state is called CA local rule. Generally, if the CA uses the same type of cell then it is called a uniform and it uses the same local rule, otherwise it is called hybrid. Cellular Automata develops by a group of cells which organized in a grid. This cell normally represent by D-Flip Flop. Each of these flip-flips connects with its neighbours and shifts to next state according to its rule. Cellular Automata called Uniform Cellular Automata (UCA) if one type of CA rule used in its structure, otherwise it is called Hybrid Cellular Automata (HCA) [8,9]. Figure 2 shows the construction of Rule 90. It shows exactly how cell C assembles with its neighbours; C-1 and C+1. Besides, this rule also can be recognized by its state transition in Eq. 1.

$$Xc(t+1) = Xc-1(t) \oplus Xc+1(t) \qquad (1)$$

Besides that, Figure 3 shows the structure of Rule 150. This rule also can be represented in its state transition equation as shown in Eq.2 below:

$$Xc(t+1) = Xc-1(t) \oplus Xc(t) \oplus Xc+1(t) \qquad (2)$$

The next state of cell for both rule 90 and 150 is shown in Table 1.

Example of 5-bits LHCA is shown in Figure 4 below. CA can increase the number of length cycle with the aid of Null Boundary condition. This condition will make the extreme cell connected to logic-0 state.

Moreover, CA has gained great attention because every cell is connected to its neighbours only. Where as in LFSR the extension in size needs main alterations to feedback loop connections, CA are readily scalable. It is sufficient to simply connect more cells to the end of existing CA.

Besides, CA also can be characterized by using transition matrix (C-matrix). Equation (3) shows CA general transition matrix where Ci represented as CA Rules. If Ci = 0; stage **i** is Rule-90 else, stage **i** is Rules-150 [3]. The matrix analysis will give the measure of the randomness in the patterns generated. This matrix can generate pseudo-random code words.

$$T_{CA} = \begin{bmatrix} c1 & 1 & 0 & \dots & 0 & 0 \\ 1 & c2 & 1 & \dots & 0 & 0 \\ 0 & 1 & c3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & cm-1 & 1 \\ 0 & 0 & 0 & \dots & 1 & cm \end{bmatrix} \qquad (3)$$

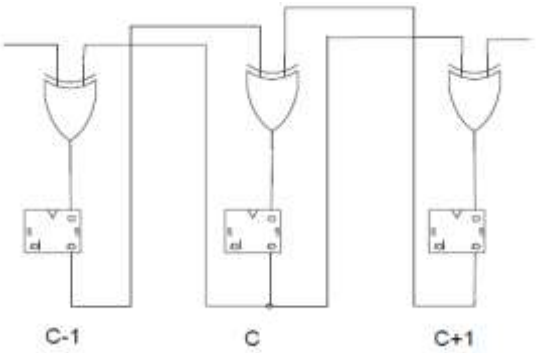

Figure 2: Rule-90 Cellular Automata construction

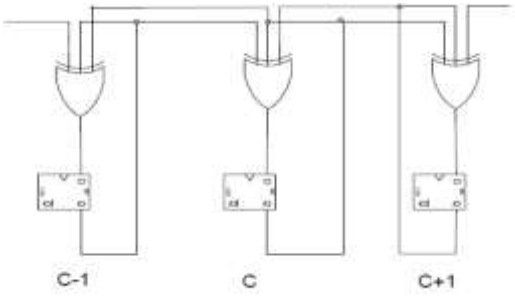

Figure 3: Rule-150 Cellular Automata construction

Table 1: Rules Updating Next State of Cells

Rule	7 111	6 110	5 101	4 100	3 011	2 010	1 001	0 000
90	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**
150	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

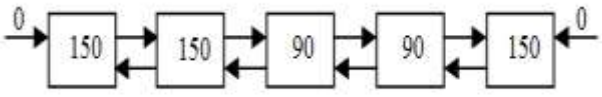

Figure 4: Example of 5-bits Hybrid 90/150 Cellular Automata

3. Test Pattern Generator

The design of CA as a test pattern generator (TPG) can be implemented using the following steps:

Step1: Maximal Cycle Length

If CA running in an independent mode, it will have a maximal cycle length, in which there is no zero state. The number of the cycles is (2^n-1), where n represents the number of bits [10]. Table 2 shows the hybrid constructions necessary to achieve a CA with maximal cycle length; n is up to 10 bits. The list of hybrid constructions is not unique; necessary to achieve a CA with maximal cycle length.

Table 2: Constructions to 10-bits Hybrid Cellular Automata

CA	Test Patterns Sequences	Construction Rule 90/150	Cycle Length
3	001	90/90/150	7
4	0101	90/150/90/150	15
5	11001	150/150/90/90/150	31
6	010101	90/150/90/150/90/150	63
7	1101010	150/150/90/150/90/150/90	127
8	11010101	150/150/90/150/90/150/90/150	225
9	110010101	150/150/90/90/150/90/150/90/150	511
10	1010101010	90/150/90/150/90/150/90/150/90/150	1023

Step2: Pseudo-random

If the test vectors generated by the CA are pseudo-random, the test set will contain appropriate vectors to activate enough faults in CUT [11]. A number of experiments have been performed to assess fault coverage. The result of the experiments is that stuck at fault coverage offered by the linear CA is satisfying, similar to the LFSR. Therefore, the test vectors generated by the CA are pseudo-random.

4. Methodology

To design an LHCA, it is desirable to determine 4-bits of Hybrid CA construction block, followed by forming a matrix transaction [12]. To extract the LHCA into hardware design, procedure starts with writing a Register Transfer Level (RTL) Code and sketching ASM Chart. Then, derive Verilog code according to RTL code and Algorithmic State Machine (ASM) chart. A complete hardware design should include a test bench for design validation. Last but not least, design compilation and simulation have been done in FPGA using Quartus II and ModelSim software:

I. Hybrid Cellular Automata Construction

A 4-bit of 90/150 LHCA will be considered in this project. By referring Table 2, a construction of 0101with 15 cycle length has been chosen. Bit 0 in this construction goes to CA Rule-90 while 1 bit is Rule-150. Thus, a four-stage hybrid CA construction can be shown in Figure 5.

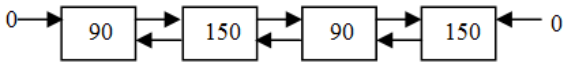

Figure 5: Four Stage of 90/150 LHCA Construction Block

Table 3 shows the test pattern sequences that generated using a 4 bit LHCA. The table shows that LHCA was able to reach the maximal cycle length.

Table 3: 4-Bits 90/150 LHCA Pattern Sequence

Cycle Length	Test Patterns Sequences			
1	0	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	0	1	1
5	0	0	1	0
6	0	1	0	1
7	1	1	0	1
8	1	0	0	1
9	0	1	1	1
10	1	0	0	0
11	0	1	0	0
12	1	1	1	0
13	1	1	1	1
14	1	1	0	0
15	1	0	1	0
	0	0	0	1

II. T ransition Matrix

A set of stage *i* transition configured from LHCA construction block; 0101. When ci= 0; stage *i* is Rule-90 whereas, stage *i* is Rules-150 when ci=1. Thus, by using general CA transition format from Figure 5, transition matrix for 4-bits of 90/150 LHCA can be formed as shown in Eq.4 below:

$$T_{CA} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (4)$$

III. RT L code and ASM chart

RTL code and ASM chart are used to simplify a complex digital circuit design on rearranged components and algorithm that needs to be counted. Besides, those methods are extremely used to describe any cycle-driven RTL circuit [13]. Each state may have several functions which can run in concurrent fashion. RTL code and ASM chart also give simple and compact hardware information that can be enough to translate into Verilog code. Below are RTL code and Figure 6 that shows the ASM chart for the overall (4-bit) LHCA system.

```
S0:   (start)*/goto S0;
      (start)/
       Xc[1:4]←4'b0001;
       done ←0;
       goto S1;

S1:   Xc[1] = 0⊕Xc[2];
       Xc[2] = Xc[1] ⊕ Xc[2] ⊕ Xc[3];
       Xc[3] = Xc[2] ⊕ Xc[4];
       Xc[4] = Xc[3] ⊕ Xc[4] ⊕ 0;

S2:   (Xc==4'b0001)/goto S1;
      (Xc==4'b0001)*/goto S3;
S3:   done←1;
      goto S0;
```

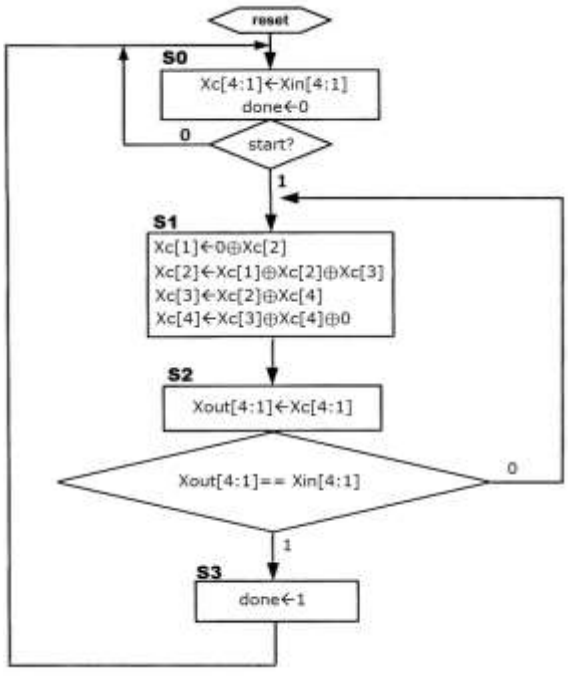

Figure 6: ASM chart

IV. Hardware Block Diagram of Hybrid CA

Input for this system consist 4-bits of Xin with a start bit and output for this system are 4-bits of Xout and done signal. Hence, overall system input and output can be represented on a simple input output block diagram (I/O Block Diagram) in Figure 7.

The details of I/O block diagram can illustrate in functional block diagram as in Figure 8.

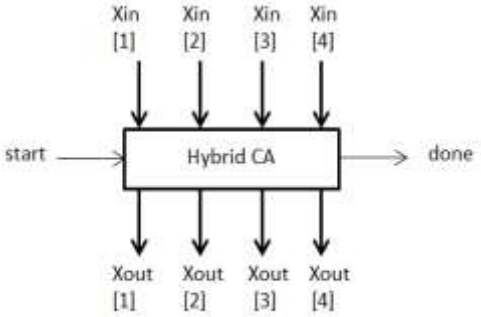

Figure 7: I/O Block Diagram

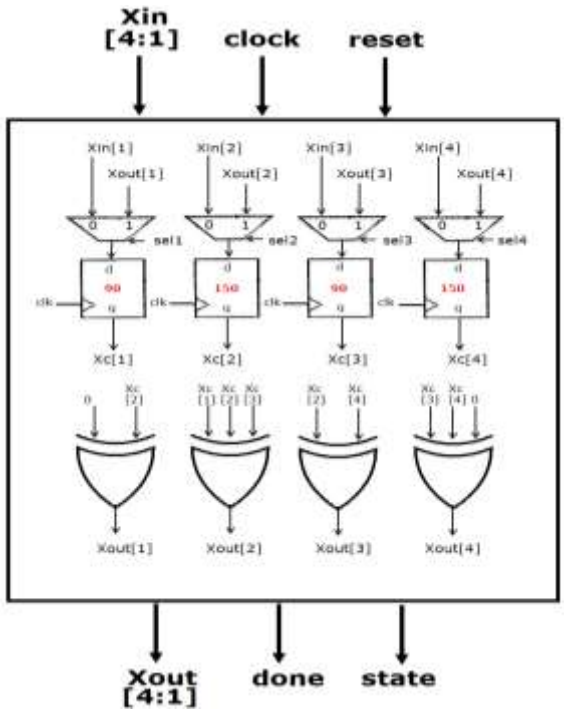

Figure 8: Functional Block Diagram

2:1 multiplexers are used as selector between primary input (seed value) and the output from the previous state (XOR output). 4-bits output from XOR gates will be the primary output for this circuit. Finite State Machine with Datapath (FSMD) structure has been determined and device family used for this work is Cyclone II EP2C50F672C6

V. Verilog code and Testbench

There are two common architectures in RTL Design which are; control unit (CU)- device unit (DU) and FSM-D architectures. In order to simplify the design, FSM-D is chosen instead of CU-DU architecture. Furthermore, system resource can be reduced by using FSM-D architecture. Less resource will produce the minimum logic area utilization. Moreover, a set

of testbench coding needed to ensure the entire design works as desired.

5. Results and Discussion

The major challenge for modelling 4-bits of 90/150 LHCA is in designing its hardware constructions. Figure 9 is a test pattern generator schematic diagram for 4-bits of 90/150 LHCA. The main components in hardware design contain four blocks of D-flip flop and four blocks of XOR gates. Therefore the novelty of this work appears through its implementation using FPGA platform. As mentioned in design methodology before, multiplexer selects between seed value and the XOR output. First of all, the flip flop is loaded with 0001 which is a seed value for this system. Then, these 4-bit values will go to XOR gates. Then, the output value from XOR gate will be loaded back into flip flop replacing seed value. Those steps will be repeated until output from XOR gate equals to seed value.

By comparing the states that obtained from simulation with those in Table 3, notice that both of test pattern values and sequence are the same. Therefore the design is correct and the LHCA is functioning well as test pattern generator. It reaches 15 maximal cycle length. Output waveform for test pattern generator in binary and decimal form obtained in Figure 10 and 11.

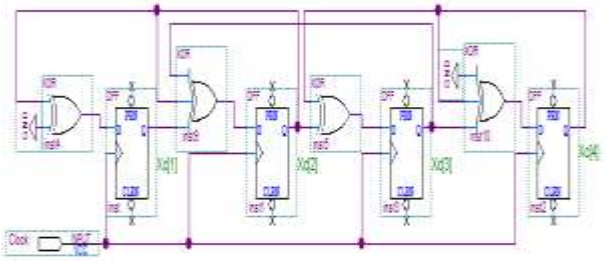

Figure 9: 4-bits 90/150 LHCA Hardware Design Schematic

Figure 10: 4-bits 90/150 LHCA output waveform in binary form

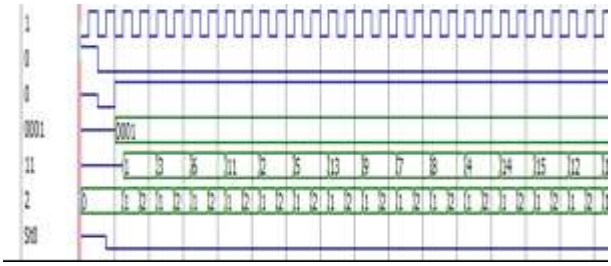

Figure 11: 4-bits 90/150 LHCA Output Waveform in Decimal Form

For this implementation, Cyclone II EP2C50F672C6 device family have been used. Table 4 shows resource and timing analyser summary for overall system.

Table 4: 4-Bits 90/150 LHCA output waveform in decimal form

Resource Summary		
logic Element Usage	19/50,528(<1%)	
Memory Block Usage	0/594,432	
I/O Usage	14/450(3%)	
Timing Analyzer Summary		
Type	Actual Time	
Worst-case tsu	4.288 ns	
Worst-case tco	6.657 ns	
Worst-case th	0.182 ns	
Clock Setup	420.17MHz	(period=2.380 ns)

6. Conclusion

Cellular Automata is the alternative BIST structure to traditional LFSRs for test pattern generation. CA can produce a lot of random pattern and easy cascaded for design flexibility. In this work, 4-bits of 90/150 LHCA serial based was implemented with 15 maximal cycle length with the aid of FPGA. By referring to the resource summary, it is found that logic element usage is less than 1%, memory block usage is 0/594,432 and I/O usage is about 3%.Timing analyzer summary shows the (worst-case tsu, tco, th) are (4.288, 6.657, 0.182) ns respectively. FSM-D design architecture is selected to extract a 4-bits of 90/150 LHCA system. Therefore; this work introduces a complete design which can be used by researchers.

References

[1] H. Jabbari, J.C. Muzio, and L. Sun, "A New Class of Cellular Automata, " In Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, " Methods and Tools (DSD '07). IEEE

Computer Society, Washington, USA, pp.331-338, 2007.

[2] C. Chen, S.K. Gupta. "BIST Test Pattern Generators For Stuck-Open And Delay Testing," European Design and Test Conference, The European Conference on Design Automation, ETC European Test Conference. EUROASIC, The European Event in ASIC Design, Proceedings, pp.289-296, 1994.

[3] A. Yarahmadi, N. Moarefi, and S. Setayeshi, "Implementing Cellular Automata with Dissimilar Rule on Serial Base," In Proceedings of the 2010 Fourth UKSim European Symposium on Computer Modelling and Simulation , IEEE Computer Society, Washington, USA, 2010.

[4] L. Gao, Y. Zhang, J. Zhao, "BIST using Cellular Automata as Test Pattern Generator and Response Compaction," Consumer Electronics, Communications and Networks (CECNet), 2nd International Conference on , Vol., No., pp.200-203, 21-23 April 2012.

[5] P. Anghelescu, E. Sofron, S. Ionita, L. Ionescu, "FPGA Implementations of Cellular Automata for Pseudo-Random Number Generation," International Semiconductor Conference, Vol. 2, pp.371, 374, 27-29 Sept. 2006.

[6] Y. Wang, D. Gu, J. Liu, X. Tian, and J. Li. "Research on Multi-dimensional Cellular Automation Pseudorandom Generator of LFSR Architecture," In Proceedings of the International Symposium on Information Engineering and Electronic Commerce (IEEC '09), IEEE Computer Society, Washington, USA, 2009.

[7] S. Cho, U. Choi, H. Kim, Y. Hwang, and J. Kim, "Analysis of 90/150 Two Predecessor Nongroup Cellular Automata, " In Proceedings of the 8th international conference on Cellular Automata for Reseach and Industry (ACRI '08), Hiroshi Umeo, Heidelberg, pp.128-135, 2008.

[8] H. Jabbari, J.C. Muzio and L. Sun, " Improved Built-In Self-Test of Sequential Circuits," Electrical and Computer Engineering, Canadian Conference, pp. 78, 81, 22-26, April 2007.

[9] I. Kokolakis, I. Andreadis, Ph. Tsalides, "Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators," Microprocessors and Microsystems, Vol. 20, Issue 10, pp. 643-658, 1997.

[10] P. Shaswati, P. Supriti Sinhamaha, M. Samaresh "Computational Intelligence in Data Mining", Springer, Singapore, 2017.

[11] S. Nandi, B. K. Kar, and P. P. Chaudhuri. "Theory and Applications of Cellular Automata in Cryptography" IEEE Trans. Computer. 43,12, 1994.

[12] P. Harshala, P. Shyam Sunder "Cellular automata for built in test pattern generation and test response analyzer" International Journal of Electrical and Electronics Research, Vol. 2, Issue 3, pp. 44-50, 2014.

[13] C. Juan Cerda, D. Chris Martinez, M. Jonathan Comer, H. David Hoe "An Efficient FPGA Random Number Generator using LFSRs and Cellular Automata," IEEE Computer Society, 2012.