Comparison of LFSR and CA for BIST

Sachin Dhingra, *Student member* IEEE
Dept. of Electrical and Computer Engineering
200 Broun Hall, Auburn University, AL 36849
Email Address: dhingsa@auburn.edu

ABSTRACT: *Built-In Self-Test (BIST), as the name suggests is a technique in which the circuit is capable of testing itself. This paper presents two techniques: Linear Feedback Shift Register (LFSR) and Cellular Automata (CA), used for test pattern generation and test response analysis in a typical BIST circuit. Both LFSR and CA are analyzed based on their construction and characteristics. A comparison of LFSR and CA is presented to demonstrate their shortfalls and suitability to certain applications.*

I. INTRODUCTION

Built-In Self-Test is a Design for Testability (DFT) technique which allows the circuit to test itself without any external equipment [2]. BIST implementation requires primarily two components: a pseudo-random test patter generator (for test vector generation) and a data compactor (for output response analysis) [1]. These components are mostly implemented using either Linear Feedback Shift Registers (LFSRs) or Cellular Automatons (CA).

LFSR is constructed using flip-flops connected as a shift register with feedback paths that are linearly related using XOR gates. An LFSR can be used for generation of pseudo-random patterns, polynomial division, response compaction etc. The CA are very similar to the LFSRs except that the registers in CA have a logical relationship with their neighbors only. This leads more randomness in the pattern generated. LFSR is more popular for implementation of both TPG and ORA due its compact and simple structure. However, CA are gaining popularity in many cases because of their characteristics and ease of modification.

This paper presents a comparison between LFSR and CA implementations in BIST. The next Section of the paper describes the basics about the implementation of BIST. LFSR and its characteristics are discussed in Section III. Section IV elaborates upon the Cellular Automata. Section V presents a comparison and analysis of both the implementations described in Sections III & IV. This is followed by a brief summary of the paper and the conclusion.

II. IMPLEMENTATION OF BIST

In order to implement BIST, extra circuitry is added to an existing circuit to enable it to test itself. The circuitry for BIST consists of four modules: Test Pattern Generator (TPG), Input Isolation Circuitry, Test Controller and Output Response Analyzer (ORA). The TPG generates pseudo-random test patterns to test the Circuit under Test (CUT). Input isolation circuit isolates the inputs of the circuit during test mode and applies the TPG output to the input of the CUT. The test response of the CUT is analyzed by the ORA. A test controller is provided to sequence and run all the BIST operations. The following figure shows the implementation of BIST.

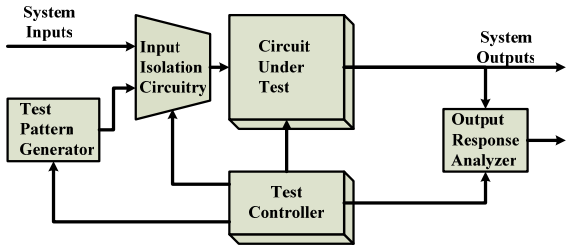

Figure 1. BIST Implementation

Test pattern generation and Output response analysis are the two most critical operations of BIST. Test patterns used in most implementations are pseudo-random in nature i.e. the random numbers are generated algorithmically and are repeatable [4]. This is a desired characteristic, as truly random test patterns will lead to different fault coverage in every execution [2]. LFSRs are most commonly used to build TPGs [6] but recently there has been interest in CA for test pattern generation. CA generate test vectors which are more random in nature. Highly random vectors help in detection of faults such as the stuck-open faults,

delay faults etc. which cannot be easily detected by vectors generated by LFSR [5].

The ORA cannot test the circuit response for every test vector due to the volume of the data required to be compared and stored, so the ORA compacts the response of CUT to a pass/fail indication, which is known as the signature of the circuit [2]. All the techniques used for ORA implementation perform test data compaction which leads to a loss of information. In some cases a fault detected in the output response may get masked due to compaction of the test response data, this situation is also known as "Signature Aliasing" [7]. Signature Analysis is the most commonly used method for implementation of ORA. Signature Analysis essentially uses an LFSR to divide the output response of a CUT by its characteristic polynomial (described in the next section) [7]. The remainder of the polynomial division at the end of the BIST sequence is known as the signature of the CUT. Difference in the signature of the CUT and the signature of a good/fault-free copy of the CUT implies a fault in the CUT. The following sections describe LFSR and CA, the two most commonly used circuits for TPG and ORA implementation in BIST.

III. LFSR

"Linear Feedback Shift Register or LFSR is a shift register whose input is the result of XOR of some of its inputs" [8]. There are two ways to implement LFSRs: Internal feedback and External feedback. These techniques differ in the way feedback is applied. All the flip-flops that feed an XOR gate are known as '*taps*'. These taps decide the patterns generated by the LFSR and hence define the *characteristic polynomial* of an LFSR. In case of an external feedback LFSR the XOR gates are in the feedback path and the input to the shift register is the XOR of all the taps. For an internal feedback LFSR, the feedback from the last FF is the input to the first FF of the shift register and all the taps are XORed with the feedback to modify the input to the next FF in the shift register as illustrated in figure 2 [2]. An internal feedback LFSR can also operate at higher speeds compared to an External feedback LFSR as there is maximum one XOR gate in any path between FFs, which is not the case for External feedback LFSR.

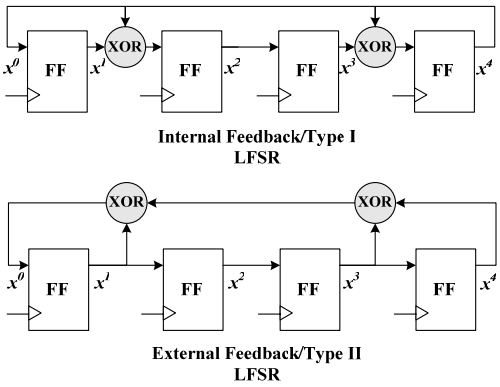

Figure 2. Two types of Linear Feedback Shift Registers

The characteristic polynomial P(x) for both types of LFSRs shown in figure 1 is:

$$P(x) = "x^0 + x^1 + x^3 + x^4" \; ; n=4$$

'n' (n=4 for LFSR in figure 1) is the degree of the polynomial which is defined by the number of bits/nodes of the LFSR. Notice that the terms *'x^0'* and *'x^n'* are always present and the remaining terms indicate the location of the taps in the circuit. The degree of the polynomial n is equal to the number of bits in an n-bit LFSR pattern. An all zeroes state is invalid for an LFSR as the state would never change if all the bits are '0'. Therefore, the maximum number of unique patterns an n-bit LFSR can generate $= 2^n - 1$, where n is the number of bits. Special LFSRs can be constructed which can generate the all zeroes state also, but they have a larger area overhead associated with them, as described in [9]. The characteristic polynomials of an n-bit LFSR which results in the generation of maximum possible unique patterns ($= 2^n - 1$) are known as primitive polynomials. The primitive polynomials are valid for both types of LFSRs. The reciprocal of a primitive polynomial is also primitive [6], i.e. Reciprocal polynomial P*(x) of primitive polynomial P(x) is also primitive. P*(x) is defined by the following equation:

$$P^*(x) = x^n P(1/x)$$

These primitive polynomials are used to construct LFSRs which generate exhaustive pseudo-random test patterns. LFSRs occupy smaller area com-

pared to a counter, as they use only a few XOR gates and flip-flops to generate exhaustive patterns. This makes them very attractive for BIST applications. Different FF outputs can be tapped to generate a test pattern more than 1-bit wide (also known as parallel patterns). For practical test times the test patterns generated by LFSRs are not more than 22-25 bits wide, so bigger circuits are partitioned into small sub-circuits of less than 25 primary inputs [2].

LFSR is also used in signature analysis. The following diagram shows typical signature analyzer constructed using LFSRs.

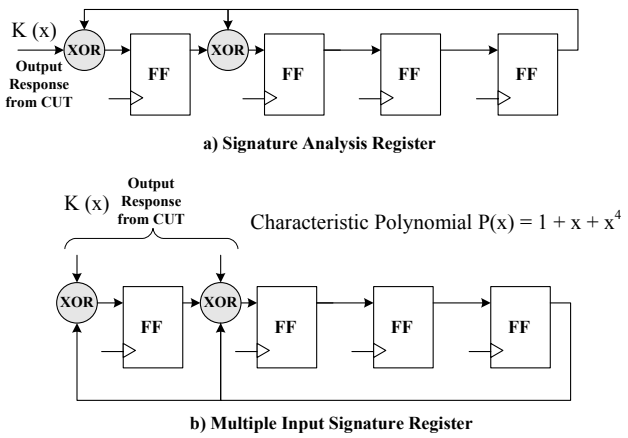

Figure 3. Signature analyzer using LFSR

As shown in part a) of figure 2 the input to the LFSR is XORed with the output response of the CUT (this structure is known as a Signature Analysis Register - SAR). The input polynomial K(x) is divided by the characteristic polynomial P(x). At the end of the BIST sequence the LFSR contains the remainder of K(x)/P(x), denoted by R(x). R(x) is called the signature of the CUT. The signature of the CUT is compared with the signature of a known good circuit. A mismatch indicates a faulty circuit [2]. Part b) of figure 3 shows multiple outputs of CUT being used to generate the signature, this structure is also known as a Multiple Input Signature Register (MISR).
Different flavors of LFSRs can be explored in [10].

IV. CA

Cellular Automaton (plural: cellular automata) evolves in steps and the value of a node depends on the value of its neighbors [11]. A Cellular Automata (CA) consists of a collection of cells/nodes formed by flip-flops which are logically related to their nearest neighbors using XOR gates [2] [4]. When the value of a node is determined only by two neighboring cells the CA is known as one-dimensional linear CA (for the rest of the text one-dimensional linear CA is referred as a CA). The logical relations which relate a node to its neighbors are known as *rules* and they define the characteristics of a CA. There are many rules which can be used to construct a CA register, the most popular being rules 90 and 150 illustrated in figure 4.

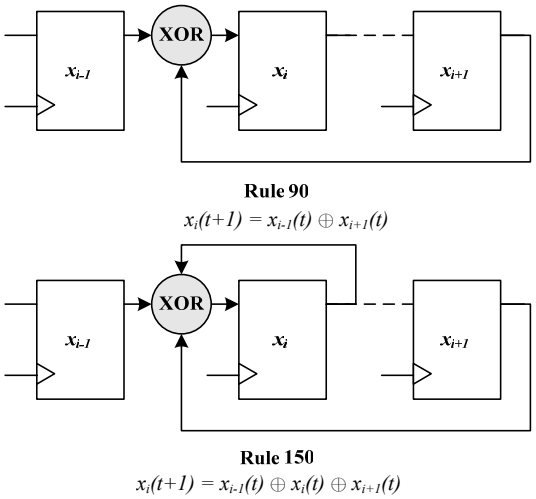

Figure 4. Cellular Automata Implementations

The next state $x(t+1)$ of node x_i is determined by the current state $x(t)$ of neighboring nodes x_{i-1} and x_{i+1} for rule 90 and nodes x_i , x_{i-1} and x_{i+1} for rule 150. All the nodes of a CA register do not have to be implemented with the same rule, different nodes can employ different rules. The first and the last nodes of a CA register have only one neighbor unlike all other nodes which have two, hence normal *rules* cannot be applied here. One solution is to assume that the missing neighbor is fixed at logic '0' (null boundary condition). The other solution assumes the last and first nodes to be neighbors and are connected using normal *rules* (cyclic condition) [2]. Connection between the end nodes (first and last nodes) introduces a feedback loop in the cyclic boundary condition; this makes null boundary condition a better choice. Figure 5 shows the construction of a 4-bit CA reg-

ister using rules 90 & 150 and null boundary condition.

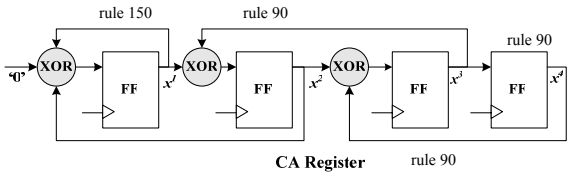

Figure 5. Cellular Automata implementation

A CA register is also known as a Linear Hybrid Cellular Automata (LHCA) or a Linear Cellular Automata Register (LCAR). Similar to LFSRs there are some combinations of rules which produce exhaustive pseudo-random patterns. LHCAs are capable of generating patterns which are more random in nature as compared to an LFSR [3]. But LHCAs have larger nodes and require many more XOR gates as compared to LFSRs. Large number of XOR gates results in higher area overhead of LHCAs. In cases where LHCA is used as a TPG, it is desirable to use a CA for Signature analysis instead of a LFSR [11].

V. COMPARISON AND ANALYSIS

There are some undesirable properties of the pseudo-random patterns generated by LFSRs, these properties led to the investigation of alternative ways to generate test patterns for BIST implementation. CA was proposed to solve some of those issues. This section presents the analysis of some of those issues and a comparison of LFSR and CA with respect to those issues.

One of the biggest drawbacks of LFSR is the inability to efficiently generate test patterns which can detect stuck-open faults in CMOS [3] and sequential faults such as delay faults in combinational circuits [1]. Such faults are detected by transitions on the inputs of a CUT and hence need a pair of consecutive input patterns in order to be detected. This kind of testing is known as two-pattern testing, described in [1] [5] [12]. The key characteristic that determines the fault coverage in two-pattern testing is the measure of randomness of the test patterns (more random patterns lead to more transitions). Parallel patterns generated by LFSRs (using outputs from different nodes of an LFSR) have a strong correlation between each other due to the shifting of data [12]. Pattern generation in CAs does not involve shifting of data and it has been shown that CAs show a better fault coverage for two-pattern tests compared to LFSRs [12] [1]. LFSR and CA are characterized by their transition matrices, the analysis of these matrices along with simulations give the measure of the randomness in the patterns generated [5], these measures show the higher randomness of patterns produced by CAs. The stuck-at fault coverage for both LFSR and CA are comparable. The mathematical explanations, theorems and proofs for all the comparisons and claims are not discussed in this paper, but are described in detail in the references.

LFSRs have a feedback from their end nodes; this means a redesign of the LFSR is needed if the pattern length has to be changed. This is not the case with CAs. CAs are logically connected to their only to their neighbors and there is no feedback for a CA employing the null boundary condition. Therefore, the pattern length generated by CAs can be easily changed by cascading the nodes. The regular structure of the nodes for CA makes them ideal for CAD tools by providing the much needed flexibility in design [13]. However, it is difficult to construct a maximum length sequence CA as compared to an LFSR which can be constructed using the primitive polynomials which are very well documented [15]. An LFSR can be implemented using only a few XOR gates whereas a CA requires at least one XOR gate for each node. This fact brings up an obvious drawback of CA: Higher area overhead involved in implementation of CA compared to an LFSR. So, the designer has to pay a penalty on the area overhead by choosing CA over LFSR.

Signature Analysis based on CA show better results than LFSR based signature analyzers in terms of Signature Aliasing [14]. There is a greater probability of a missing an error by aliasing in LFSR compared to CA due to shifting of data in a LFSR. In case of CA each node value is a function of the neighboring nodes resulting in a lower probability of missing an error [11]. Moreover, it is preferable to construct a MISR or a SAR using CA, if CA is used to generate the test patterns. The performance of CA based TPGs in terms of speed of operation is higher than an External Feedback LFSR based TPG. The presence of XOR gates in the feedback path of an External

Feedback LFSR and lack of a feedback path in a null boundary condition CA results in higher operating speed for CAs. The following table lists the comparisons between LFSR and CA.

Table 1. Comparison of LFSR and CA

Characteristic	LFSR	CA
Area Overhead	Least – very few XOR gates	More – XOR gates are needed for every node
Performance	Very Good in case of internal feedback LFSR. Poor for external feedback shift register	Good – no feedback paths and maximum one XOR gate between nodes
Randomness of Parallel Patterns	Low – shifting of data causes correlation between patterns	High – No shifting of data
Error detection:		
stuck-at faults	High	High
stuck-open faults	Low – due to less transitions	Higher than LFSR – more transitions
delay faults	Low – due to less transitions	Higher than LFSR – more transitions
CAD friendly	No – Requires redesign for change of pattern length	Yes – nodes can be cascaded and pattern length can be easily changed
Signature Aliasing	Low probability	Lower probability than LFSR – No shifting of data prevents masking of error bits

VI. Summary and Conclusions

LFSRs have been well researched and provide a compact circuit for applications using polynomial division and generation of patterns. Their popularity is owed to simple and compact design. For BIST applications they find use in the pseudo random test pattern generation and signature analysis. But research has shown some of the shortfalls of the LFSR can be overcome by the use of Cellular Automata. CA use larger nodes as compared to the LFSR but provide much more random patterns and can be easily cascaded for design flexibility. For testing of delay faults and stuck-open faults, test patterns which are more random in nature provide better fault coverage, so in many cases they are being looked upon as the alternatives to conventional LFSRs for test pattern generation and output response analysis in BIST.

References

[1] S. Zhang, R. Byrne, J.C. Muzio, D.M. Miller, "Why cellular automata are better than LFSRs as built-in self-test generators for sequential-type faults", IEEE International Symposium on Circuits and Systems, Vol. 1, pp 69-72, 1994

[2] C. Stroud, *A Designer's Guide to Built-In Self-Test*, Kluwer Academic Publishers, Boston MA, 2002

[3] P.D. Hortensius, R.D. McLeod, W. Pries, D.M. Miller, H.C. Card, "Cellular automata-based pseudorandom number generators for built-in self-test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, pp 842 - 859, 1989

[4] M.L. Bushnell, V.D. Agrawal, *Essentials of Electronics Testing for Digital, Memory & Mixed Signal VLSI Circuits,* Kluwer Academic Publishers, Boston MA, 2000

[5] K. Furuya, E.J. McCluskey, "Two-Pattern test capabilities of autonomous TPG circuits," *Proc. of International Test Conference*, pp 704 – 711, 1991.

[6] P.H. Bardell, W.H. McAnney, J. Savir, *Built-in test for VLSI: Pseudorandom Techniques,* John Wiley and Sons, New York, 1987

[7] D. Bhavsar and R. Heckelman, "Self Testing by Polynomial Division," *Proc. IEEE International Test Conference,* pp. 208 – 216, 1981.

[8] __, "Linear Feedback Shift Register," en.Wikipedia.org

[9] L. Wang and E. McCluskey, "Complete Feedback Shift Register Design for Built-In Self-Test," *Proc. IEEE International Conference on Computer-Aided Design,* pp. 56-59, 1986

[10] L.T. Wang, E.J. McCluskey, “Circuits for Pseudoexhaustive Test Pattern Generation,” *Proc. IEEE International Conference on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 7, pp. 1068 – 1080, 1988

[11] P.D. Hortensius, R.D. McLeod, H.C. Card, “Cellular automata-based signature analysis for built-in self-test,” *IEEE Transactions on Computers*, Vol. 39, pp. 1273 – 1283, 1990

[12] K. Furuya, S. Yamazaki, M. Sato, “Evaluations of various TPG circuits for use in two-pattern testing,” *Proceedings of the Third Asian Test Symposium*, pp. 242 – 247, 1994

[13] C.S. Jr. Gloster, F. Brglez, “Boundary scan with cellular-based built-in self-test,” *Proc. 'New Frontiers in Testing' International Test Conference,* pp. 138 - 145, 1988

[14] D.M. Miller, S. Zhang, W. Pries, R.D. McLeod, “Estimating aliasing in CA and LFSR based signature registers,” *Proc. of IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 157 – 160, 1990

[15] M. Serra, T. Slater, J. C. Muzio & D. M. Miller, “The Analysis of One Dimensional Linear Cellular Automata and Their Aliasing Properties,” *IEEE Trans. on CAD*, pp. 767-778, 1990