A Built-In Self Test Scheme for VLSI

T. Raju Damarla
National Research Council
US Army Research Labs
AMSRL-PS-EA
Fort Monmouth, NJ 07703

Wei Su
US Army Research Labs
AMSRL-PS-EA
Fort Monmouth, NJ 07703

Moon J. Chung
Dept. of Computer Science
Michigan State University
East Lansing, MI 48824

Charles E. Stroud
Dept. of Electrical Engineering
University of Kentucky
Lexington, KY 40506

Gerald T. Michael
US Army Research Labs
AMSRL-PS-EA
Fort Monmouth, NJ 07703

Abstract: In this paper we present a novel approach for Built-In Self Test (BIST) for VLSI. Many conventional BIST schemes use signatures generated by a linear feedback shift register (LFSR) or a multiple input signature register (MISR) for determining whether the device under test is faulty or fault free. In the approach presented in this paper, fault detection is made based on the number of different states the LFSR visits. This number is called the cycle length. It is also shown that such an approach results in the probability of aliasing of $2^{-\left(2^{m-1}+m\right)}$, where *m* denotes the number of registers in the LFSR, compared to 2^{-m} achieved by conventional signature analyzers. We also present the complexity of the additional hardware required to implement the scheme.

1. Introduction

Currently, there are many built-in self test (BIST) schemes [1-5] in existence. Among them most widely used BIST approach is the signature analysis. In signature analysis, the test responses of a system are compacted into a signature using a linear feedback shift register (LFSR) or a multiple input signature register (MISR) as shown in Fig. 1. Then the signature of the device under test (DUT) is compared with the expected (reference) signature. If they both match, the DUT is declared fault free, else it is declared faulty. Since several thousands of test responses are compacted into a few bits of signature by an LFSR/MISR, there is an information loss. As a result some faulty devices may have the same correct signature. The probability of a faulty device having the same signature is called the probability of aliasing. The probability of aliasing is shown to be [6] approximately 2^{-m}, where *m* denotes the number of bits in the signature (number of registers in LFSR/MISR). This probability may be higher if the faults in the DUT are correlated. As a result there is no guarantee that the device declared to be fault free is really fault free.

In this paper, we present a new approach for built-in self test that results in lower probability of aliasing. In this scheme the outputs from the device under test will be applied to an LFSR, and the states generated by the LFSR will be monitored. The number of different states generated by an LFSR before repeating is called a "cycle length" and all those states are said to form a "cycle". The cycle is called a maximal length cycle if it equals 2^m. For a fault free device the cycle length is determined and this cycle length is used to determine whether a DUT is faulty or not. Suppose, a fault free device generates a maximal length cycle, then a faulty device may not generate a maximal length cycle, there by detecting the faults. The probability that a faulty device having the same cycle length is very low, and its calculations will be presented in Section 3. In general, determination of cycle length requires the knowledge of the states visited by an LFSR. This implies that the past history should be maintained in order to determine the cycle length. However, if the cycle length is 2^m, determination of cycle length would be easy. A simple circuit presented in Fig. 2 can be used to determine whether the cycle length is 2^m or not. Hence, if the fault free

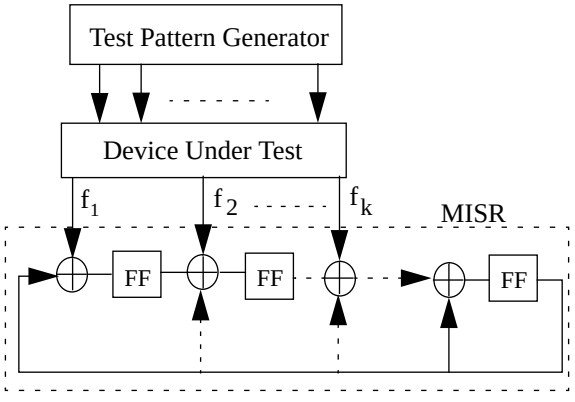

Fig. 1: General Set up for Signature Analysis

device does not generate all 2^m states, an auxiliary function

will be used to ensure that the LFSR would generate all 2^m

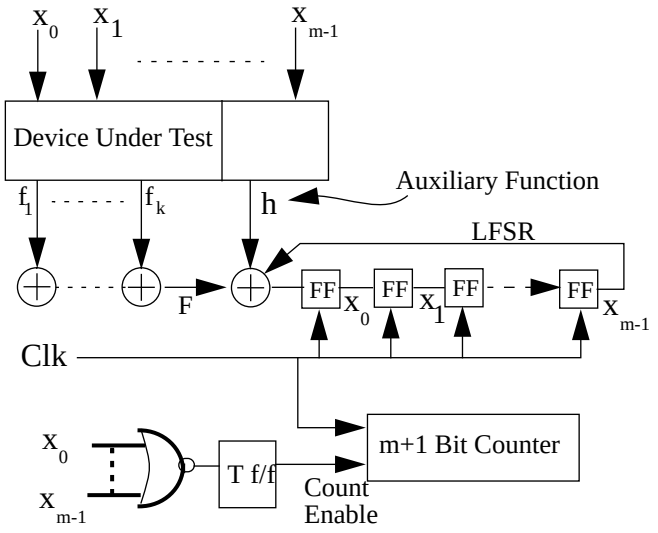

Fig. 2: General Scheme for New BIST

states. The outline of the rest of the paper is as follows. Section 2 presents a scheme for generating of cycles of required length along with the new BIST scheme. Section 3 presents the calculations for the probability of aliasing with the new scheme and it will be shown to be $2^{-\left(2^{m-1}+m\right)}$. For example, if m=8, the probability of aliasing by the new scheme would be 2^{-136}, while the probability of aliasing by signature analysis is 2^{-8}. Hence, the proposed technique results in a better BIST scheme. The additional hardware required by the new approach will be presented.

2. Description of New BIST Approach

Several researchers [8,9] have studied signature analysis techniques in order to reduce the probability of aliasing. It is this uncertainty that makes us to investigate for better testing schemes. In this section, we will present a new alternative technique to signature analysis. The general approach for this technique is shown in Fig. 2. The idea for this approach is that when the device is fault free, the LFSR should generate all possible states, giving the cycle length of 2^m. The designing process of such a system consists of designing an auxiliary function, if necessary, when applied with the function $F=f_1 \oplus ... \oplus f_k$, the LFSR would generate all 2^m states. The approach consists of first constructing an LFSR with a primitive polynomial of degree *m* that generates all non-zero states. Next an auxiliary function is selected such that when it is applied with the function *F*, the LFSR in Fig. 2 would generate the same states generated by the LFSR with the primitive polynomial as its characteristic polynomial. Note that the characteristic polynomial for the LFSR in Fig. 2 is x^m+1. Steps involved in designing a BIST scheme given in Fig. 2 are given below:

Step 1: First determine the maximum number of variables any function f_i, for all *i*, is dependent on for a given digital system. Let this number be '*m*'. Then construct an LFSR with a primitive polynomial of degree *m* as shown in Fig. 3b. Initialize the LFSR with a '100...0' pattern. Apply the states of the LFSR as test vectors to the digital system and collect the output '*F*', and the input to the LFSR 'Z' and x_{m-1}, for every state of the LFSR. Reset the LFSR with an all '00...0' pattern and collect '*F*'.

Step 2: Assume that the auxiliary function '*h*' is a function of *m* variables, and the state of the LFSR is the input to it as shown in Fig. 2. Then generate the truth table for the *h(X)* using the information collected in Step 1 and using the equation:

$$h(X) = F(X) \oplus Z(X) \oplus x_{m-1} \qquad (1)$$

for every state 'X' of the LFSR.

Step 3: Minimize the auxiliary function *h(X)* and implement it as a part of the test system as shown in Fig. 2.

In the test mode, initialize the LFSR in Fig. 2 with '00...0' vector, and run the system for 2^m clocks. Clearly from the above construction we find that the input to the LFSR in Fig. 2, $h(X) \oplus F(X) \oplus x_{m-1} = Z$ is same as the feedback input Z to the LFSR with a primitive polynomial (Fig. 3b) used in Step 1. Since the LFSR in Fig. 3b is constructed with EXOR gates outside shift register chain, its states are determined entirely by the input bit stream 'Z' entering it. Since the same bit stream 'Z' is applied to the LFSR in Fig. 2, its states would be identical to the states generated by the LFSR with a primitive characteristic polynomial constructed as shown in Fig. 3b.

Example 1: Design a BIST network for a system consisting of three functions given in Table 1. We first select the primitive polynomial $p(x)=x^4+x+1$ of degree 4 for the LFSR. Then the states generated by the LFSR with this primitive polynomial starting from the '0001' state are given in Table 1. The outputs corresponding to each state for various functions are also listed in Table 1. We monitor the function $F = f_1 \oplus f_2 \oplus f_3$,

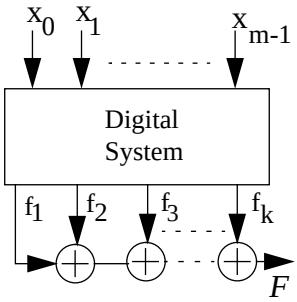

Fig 3a: System

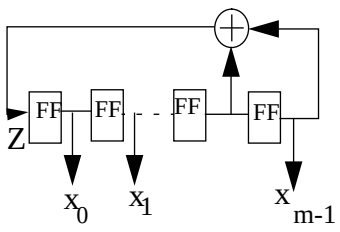

Fig 3b: LFSR for Test Patterns Generation

Fig. 3: Set up for collection of data

input '*Z*' to LFSR and compute value of the auxiliary function

Table 1: Various Values to be observed

State of LFSR $x_3x_2x_1x_0$	f_1	f_2	f_3	F	Z	h	$g = F \oplus h$
0000	1	0	0	1	1	0	1
0001	1	0	1	0	0	0	0
0010	1	1	1	1	0	1	0
0100	0	1	1	0	1	1	1
1001	1	0	0	1	1	1	0
0011	0	0	0	0	0	0	0
0110	0	0	1	1	1	0	1
1101	1	0	1	0	0	1	1
1010	0	0	1	1	1	1	0
0101	1	0	0	1	1	0	1
1011	1	1	0	0	1	0	0
0111	0	1	0	1	1	0	1
1111	1	1	0	0	0	1	1
1110	0	0	0	0	0	1	1
1100	1	0	1	0	0	1	1
1000	1	1	1	1	0	0	1

'*h*' for each state using Equation 1.

Once, the BIST is designed as described in Steps 1 - 3, one needs to determine the cycle length for a device under test in test mode to determine whether it is faulty. This can be done by a simple circuit given in Fig. 2. For the scheme shown in Fig. 2, initialize the LFSR with '00...0' and clock it 2^m times. If the number of clocks used are 2^m and the contents of LFSR is '00...0', then declare the DUT to be fault free else faulty.

What follows now is an explanation why a faulty circuit once enters a cycle it does not leave the cycle. Let $g(X)=F(X)\oplus h(X)$. Since there are no storage elements in the paths of outputs of f_i and h, the function $g(X)$ has an unique value for each state X resulting in an unique state of the circular shift register (LFSR).

Remark: Since the closed loop nature of the Fig. 2, the LFSR generates a set of states that form a cycle. Once the circuit enters a cycle, it does not come out of the cycle, since

$$X_i = \left(x_{i,0}, x_{i,1}, \ldots, x_{i,m-1} \right)$$
$$X_{i+1} = \left(g\left(X_i \right) \oplus x_{i,m-1}, x_{i,0}, x_{i,1}, \ldots, x_{i,m-2} \right)$$

where X_i and X_{i+1} denotes the ith and $(i+1)$th states of the LFSR. Since $g(X_i)$ is fixed, the next state is unique. Hence if the circuit enters cycle, it will remain in the cycle. If a faulty device results in a maximal cycle length, then the input function to the LFSR should be one of the $\aleph$ functions (see Section 3), otherwise the circuit enters a smaller cycle and remains in it. Hence the fault would be detected.

The circuit shown in Fig. 2 works as follows: When the LFSR is reset to '00...0', the T flip flop will be set to '1' allowing the counter to count. When the '00...0' state is reached the T flip flop will be reset to '0' stopping the counter. The contents of the counter would indicate how many clock pulses are applied.

3. Aliasing Probability based on Cycle Lengths

We assume that we are using the approach shown in Fig. 2 to determine whether a DUT is faulty or fault free based on the cycle length. We are also assuming that the error masking due to the linear combination of double bit errors at the outputs $f_1,\ldots,f_k$ is statistically the same as that which would occur in a MISR due to diagonal errors. There is a danger of declaring a faulty circuit as fault free if the faults are such that the resulting input to the LFSR takes it through all states but in a different order starting from '00...0' and ending in '00...0' preserving the cycle length to 2^m. We will now compute this probability of aliasing P_a.

Since the DUT is declared fault free only if the number of clocks used is 2^m and the contents of the LFSR is '00...0', in order to compute the probability of aliasing we need to compute how many functions $g(X)$ give cycle length of 2^m, in the setting shown in Fig. 4. Consider the network shown in Fig. 4. The input to this network is a four variable function and the number of flip flops in the circular shift register is four. If the original function applied to this circuit is same as the function given in Table 1, then the cycle generated by this function is 2^4=16. Suppose, due to a fault the input function changed to another function $g'(X)$ giving a cycle length of 16, then we have aliasing. Suppose, there are $\aleph$ number of functions that generate maximal cycle length, then the aliasing probability in general is given by

$$P_a = \frac{\aleph - 1}{2^{2^m}} \qquad (2)$$

where $\aleph$ is the total number of different functions that generate the cycle length of 2^m and 2^{2^m} is the total number of possible functions with m variables, where m denotes the number of flip flops in the LFSR shown in Fig. 4. For the case m=4, $P_a = \frac{\aleph - 1}{65536}$. The above probability is derived with the assumption that all functions are equally likely. Now, we will compute $\aleph$. In order to compute $\aleph$, we will first find the properties of the functions that result in cycle length 2^m.

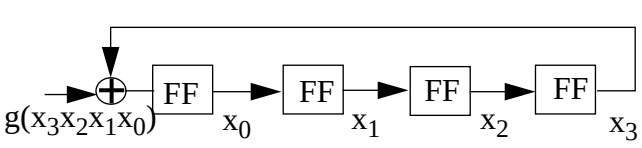

Fig. 4. Circular Shift Register with a single input

At this point we would like to mention that the input g to the LFSR in Fig. 4 appears in a particular order deter-

mined by the states of the LFSR when compared to the same function represented in a truth table. Table 2 illustrates this point for the function g given in Table 1. The function re-written in the numerical order of the states as it should appear in a truth table is given in columns 3 and 4. So the function f in column 4 is exactly same as g in column 2 but written in different order. In order to find their properties, we will be dealing with the sequences 'g' and their corresponding functions 'f'. Some of the lemmas given below relate to the properties of transformations that take sequences 'g' into functions 'f'.

Definition: Let $g(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$, $a_n \neq 0$. Then the reciprocal polynomial g^* of g is defined by [5]:

$$g^*(x) = x^n g(\frac{1}{x}) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1}x + a_n.$$

Property 1: Let $g(x)=q(x)p(x)$ then from the above definition $g^*(x)=q^*(x)p^*(x)$

Property 2: An input polynomial $g(x)$ that generates all 2^m states in an LFSR with characteristic polynomial $p(x)=x^m+1$ shown in Fig. 4 is divisible by $p(x)$ and hence $g(x)=q(x)p(x)$, and the states generated by the LFSR are identical to the m-bits taken at a time in the normal bit pattern representation of $q(x)$.

Example 2: Let m = 4, and let the $g(x)$ applied to the LFSR with $p(x) = x^m+1$ shown in Fig. 4 is g=1001001101011111 (represented in its normal form) and the q=0000100110101111. This sequence of bits for q is same as the sequence of bits that appear at the output of the last register of LFSR (x_3 register, in Fig. 4). The first four bits of q gives the state '0000', the second four bits give '0001' state.

Table 2: Function g(X) to circular shift register

State of LFSR $x_3x_2x_1x_0$	g	Truth Table $x_3x_2x_1x_0$	f	f^*
0000	1	0000	1	1
0001	0	0001	0	1
0010	0	0010	0	1
0100	1	0011	0	1
1001	0	0100	1	0
0011	0	0101	1	0
0110	1	0110	1	0
1101	1	0111	1	1
1010	0	1000	1	1
0101	1	1001	0	1
1011	0	1010	0	1
0111	1	1011	0	1
1111	1	1100	1	0
1110	1	1101	1	0
1100	1	1110	1	0
1000	1	1111	1	1

and so on. Similarly, if g^*=1111101011001001 then q^*=0000111101011001 and clearly the states of the LFSR generated by g^* are exactly identical to the ones generated by g but in reverse order. Hence if g generates all 2^m states, then g^* would generate all 2^m states.

Lemma 1: Let g be a vector, and $g^*=Eg$ be the reciprocal of g, where

$$E = \begin{bmatrix} 0 & \dots & 0 & 1 \\ 0 & \dots & 1 & 0 \\ & \dots & & \\ 1 & \dots & 0 & 0 \end{bmatrix},$$

if there exists linear transforms T and T^* such that $f=Tg$ and $f^*=T^*g^*$, then the vector $f^*=Ef$ is the reciprocal of f.

Proof: Let $T^*=ETE^{-1}$, then

$$f^*=T^*g^*=ETE^{-1}Eg=ETg=Ef.$$

Lemma 2: Let g be an input polynomial when applied to an LFSR whose characteristic polynomial is x^m+1 generates all 2^m states and the corresponding Boolean function $f=(f(0),f(1),...,f(2^m-1))$ (g written as a function of the output state of LFSR). Then the reciprocal of f denoted by $f^*=(f(2^m-1),...,f(1),f(0))$ and the corresponding polynomial g^* also results in a cycle length of 2^m**.**

Proof: Clearly there is a linear transform T that converts g to f since we can generate g from f. From Properties 1 & 2, it is clear that g^*, the reciprocal of g, also generates all 2^m state. Now, from Lemma 1, we have $T^*g^* = f^*$. But the relationship between f and f^* is given by

$$f(x_{m-1}x_{m-2}...x_1x_0) = f^*(\bar{x}_{m-1}\bar{x}_{m-2}...\bar{x}_1\bar{x}_0).$$

Hence the proof.

From now onwards we use the function f and g interchangeably as they both are identical except for reordering. Table 2 gives the function $f(X)$ and its $f^*(X)$.

Lemma 3: Let $f(x_{m-1}x_{m-2}...x_1x_0)$ be a function that generates a cycle of maximal length 2^m, then the function should satisfy the property $f(0x_{m-2}...x_0)=f(1x_{m-2}...x_0)$ where x_{m-1} is the most significant variable in the truth table. In other words, the functional values in the first half of the truth table should be identical to the second half.

Proof: Let $X(t) = (x_{m-1}(t),x_{m-2}(t), ... ,x_0(t))$ be the current state of the circular shift register (LFSR) in the Fig. 4 setting, then the next state is given by

$$X(t+1) = (x_{m-2}(t), ... ,x_0(t),f(X(t)\oplus x_{m-1}(t)) \qquad (3)$$

If $f(x_{m-1}(t),x_{m-2}(t), ... ,x_0(t)) \neq f(\bar{x}_{m-1}(t),x_{m-2}(t), ... ,x_0(t))$, then $f(X(t))\oplus x_{m-1}(t)=\bar{f}(X(t))\oplus\bar{x}_{m-1}(t)$, where $\bar{f}$ denotes the comple-

ment of *f*, hence by Equation 3 the state generated after the present state $(x_{m-1}(t),...,x_0(t))$ and $(\bar{x}_{m-1}(t),...,x_0(t))$ will be identical. Since no two states should go to the same next state for maximal length cycle

$$f(x_{m-1}(t),x_{m-2}(t), \ldots ,x_0(t)) = f(\bar{x}_{m-1}(t),x_{m-2}(t), \ldots ,x_0(t)).$$

Table 2, illustrates the concepts described in Lemma 3, that is the first half of the table for *f* is identical to the later half.

Lemma 4: Let $f(x_{m-1} \ldots x_0)$ is a function that results in maximal length cycle, then the function should satisfy the property

$$f(00...0) = f(011...1) = 1$$

$$f(10...0) = f(111...1) = 1$$

where x_{m-1} is the most significant variable.

Proof: Suppose *f*(00...0)=0. In a maximal length cycle '00...0' is one of the states. If *f*(00...0)=0 and the present state is 00...0, then it will stay put in the same state for ever giving a cycle of cycle length 1. Hence *f*(00...0) = 1 for a maximal length cycle generating function. From Lemma 3, we have *f*(00...0) = *f*(10...0) = 1. Hence for any function *f* that generates maximal cycle length *f*(100..0) = 1. Since, the reciprocal function *f** also generates maximal length cycle, we have *f**(100...0) = 1. Then, from Lemma 2, we get *f**(10...0) = *f*(01...1) = 1. Since *f*(01...1) = 1, from Lemma 3 we conclude *f*(11...1) = 1.

Theorem 1: The number of functions with *m* number of variables that generate maximal length cycle when applied to a circular shift register (LFSR) of *m* flip flops whose outputs are in turn applied as inputs to the function is given by

$$\aleph = 2^{2^{m-1} - m} \quad (4)$$

Proof: Proof is given in the Appendix.

From Equation 2, we find that the probability of aliasing as

$$P_a = \frac{2^{2^{m-1} - m} - 1}{2^{2^m}} \approx \frac{1}{2^{2^{m-1} + m}} = 2^{-\left(2^{m-1} + m\right)} \quad (5)$$

which is much smaller compared to the probability of aliasing obtained by signature analysis, which is equal to 2^{-m}. Clearly, finding the cycles is a better way of testing the circuits.

4. Hardware Complexity of Auxiliary Function *h*(X)

In this section, we will present the hardware complexity of the auxiliary function *h(X)*. In general this hardware complexity could be high. However, the size of the LFSR for a given probability of aliasing will be small. For example, if the desired probability of aliasing for a system is 2^{-100}, then the size of the LFSR would be 8. In the worst case, the hardware complexity would be 2^7. For this calculation, we assumed that *h(X)* can have at most 2^{m-1} minterms, where *m* denotes the number of variables in function *h(X)*. If *h(X)* contains more than 2^{m-1}, then implement the complement of *h(X)* and use an inverter to get *h(X)*. The following Table 3 presents the worst case hardware complexities for various *m* and the resulting probability of aliasing. From Table 3, it is clear that the probability of aliasing is very small even for *m=8*. However, it should be noted that the LFSR size will normally be selected based on the number of test patterns that must be applied to the largest function in the DUT as opposed to the desired aliasing probability. As a result, for most practical logic functions, the probability of aliasing with this approach will be extremely low.

5. Conclusion

In this paper we have introduced a new BIST approach that provides better probability of aliasing than the conventional methods. In this new scheme, fault detection is performed based on the cycle length of the LFSR rather than the signature. It is shown that, if the size of the LFSR is *m* then the probability of aliasing is shown to be $2^{-\left(2^{m-1} + m\right)}$. The hardware complexity is also shown to be much smaller than the conventional approach. The approach, as described in this paper, assumes that the DUT is composed completely of combinational logic. However, it should be noted that this approach can be extended to test sequential logic by the incorporation of the necessary initialization and control circuitry to obtain reproducible results from sequential logic.

References

[1]. M. Abramovici, M.A. Breuer and A.D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, New York, 1990.

[2]. Agrawal, Vishwani D. and Sharad C. Seth. *Tutorial: Test Generation for VLSI Chips.* Washington: Computer Society Press. 1988.

Table 3: Hardware Complexity

Size of LFSR *m*	HW Complexity	Prob. of Aliasing	Prob. of Aliasing by Signature Analysis
4	2^3	2^{-12}	2^{-4}
5	2^4	2^{-20}	2^{-5}
6	2^5	2^{-38}	2^{-6}
7	2^6	2^{-71}	2^{-7}
8	2^7	2^{-136}	2^{-8}

[3]. Maunder, Colin M. and Rodham E. Tulloss. *The Test Access Port and Boundary Scan Architecture.* Washington: Computer Society Press. 1990.

[4]. Reghbati, Haffan K. *Tutorial: VLSI Testing and Validation Techniques.* Washington: Computer Society Press. 1985.

[5]. P.H. Bardell, W.H. McAnney and J. Savir. *Built-in Test for VLSI.* John Wiley & Sons, New York, 1987.

[6]. J.E. Smith, "Measures of the effectiveness of fault signature analysis," IEEE Transactions on Computers, Vol. C-29, No. 6, pp. 510-514, June, 1980.

[7]. R. G. Bennetts: *Design of Testable Logic Circuits,* Addison-Wesley Publishing Co. Reading, MA, 1984.

[8]. D.K. Pradhan and S.K. Gupta, "A new framework for designing and analyzing BIST techniques and zero aliasing compression," IEEE Trans. on Computers, Vol. 40, No. 6, June 1991, pp. 743-762.

[9]. T.W. Williams, W. Daehn, M. Gruetzner and C.W.Starke, "Aliasing errors in signature analysis registers," IEEE Design and Test, Vol. C-36, No. 4. pp.39-45, April 1987.

[10]. H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms", SIAM Review, Industrial and Applied Mathematics, pp. 195-221, 1982.

APPENDIX

Here, we will present the proof for Theorem 1. Consider the circuit given in Fig. 4. Let us denote the state of the LFSR by a vector $\underline{X}(t)=[x_{m-1}(t), ..., x_0(t)]^T$, where T denotes the transpose of a matrix. Then the relation between the next state and the present state is given by:

$$\underline{X}(t+1) = [A]\,\underline{X}(t) \oplus [B]\,f(\underline{X}(t))$$

where $[A] = \begin{bmatrix} 0 & I \\ 1 & 0 \end{bmatrix}$, $[B]=[0 \ldots 0\ 1]^T$, where I denotes a m-1 x m-1 identity matrix and $f(\underline{X}(t))$ is a scalar. The state of the machine after 2^m clocks is given by

$$\begin{aligned}
\underline{X}(t+1) &= [A]\,\underline{X}(t) \oplus [B]\,f(\underline{X}(t)) \\
\underline{X}(t+2) &= [A]^2\,\underline{X}(t) \oplus [AB|B] \begin{bmatrix} f(\underline{X}(t)) \\ f(\underline{X}(t+1)) \end{bmatrix} \\
\ldots &\quad \ldots \\
\underline{X}\left(t+2^m\right) &= [A]^{2^m}\,\underline{X}(t) \oplus [\Phi] \begin{bmatrix} f(\underline{X}(t)) \\ \cdots \\ f\left(\underline{X}\left(t+2^m-1\right)\right) \end{bmatrix}
\end{aligned}$$

where $\Phi = \begin{bmatrix} A^{2^m-1}B| \ \ldots | \ AB|B \end{bmatrix}$ is the controllability matrix with a dimension of m x 2^m. Since the pair (A,B) is controllable, its rank$(\Phi)=m$. Substituting that $\underline{X}(t+2^m)=\underline{X}(t)$ for a function with cycle length of 2^m and solving for the function vector, we get

$$\Phi \begin{bmatrix} f(\underline{X}(t)) \\ \cdots \\ f\left(\underline{X}\left(t+2^m-1\right)\right) \end{bmatrix} = \underline{X}(t) \oplus [A]^{2^m}\,\underline{X}(t) \qquad (6)$$

$f(\underline{X}(j))$ can be obtained by solving m linear equations given by (6). Hence, the above equation results in m linear equations and 2^m functional values $f(\underline{X}(j))$ to be determined. But from Lemma 3, only the first half (2^{m-1}) of the elements in truth table need be determined as the other half is identical. Hence, the degree of freedom in the above equation is 2^{m-1}-m and hence the number of functions that generate cycles of length 2^m is given by $\aleph = 2^{2^{m-1}-m}$ as 2^{m-1}-m of combinations of 0's and 1's are there. Hence the proof.

Computer simulations were made to determine the number of functions that generate maximal length cycle. It is found for m = 2,3,4, the numbers obtained by simulations and the numbers obtained by computing $\aleph$ given by the above formula are identical. A graph theoretical proof for $\aleph$ is presented in [10].