Optimized Test Application Time and Test Data Volume Using LFSR for Combinational Circuits

Muddapu Harika[1], E. Lakshmi Prasad[2]
[1]M.tech in VLSI, JNTUA College Of Engineering, Ananthapuramu
[2]Lecturer, Department of ECE, JNTUA College Of Engineering,Ananthapuramu

Abstract- The main objective of this project is to reduce the application test data time and test data volume using customized LFSR for combinational ATPG and sequential ATPG circuits. There are many existing standard LFSRs are used for generating the test patterns but the proposed method able to reduce the application test data time and test data volume by selecting the appropriate seed value by using different characteristic polynomials. The main purpose of customized LFSR is to generate a limited number of bits to achieve test data compression. An LFSR will produce a pseudo-random sequence with a maximum length of (2^{n-1}) states (where n is the number of stages). The number of states can be reduced by selecting the appropriate characteristic polynomial for LFSR. The test application data and data volume can be reduced by selecting the appropriate seed value. Here, in this paper customized LFSR is designed with respect to 4-bit, 8-bit and 16-bit for all combinational circuits.

***Index Term*:**LFSR, Test compression, characteristic polynomial, data volume.

1. INTRODUCTION:

The non-zero first state of the LFSR flip-flops is called the seed value.The LSFR technique is to produce the total of random test patterns based on the selection of the seed value. This approach is executed in three steps: in the first step, selecting the test vectors. In the second step, an exhaustive set of pseudo-random patterns is generated with a non-zero value as the initial seed. In the third step, a heuristic approach is made by comparing witha minimumnumber of ATPG patterns and the large set of random patterns. An N-bit LFSR produces the (Pseudo) random patterns initially from the seed value and continues to generate all possible 2^{N-1} combinations. The number of faultsis distinguished by applying the random. By using an LFSR few patterns are enough to get good fault coverage. Later, to avoid the huge number of redundant patterns between the usage patterns, the LFSR can be loaded with an initial value to produce the group of patterns which are more useful to detect faults[1]. A test vector is considered as the seed (initial test vector) of that group based on the following factors.

1. Seed vector can produce all the successive patterns in the group of consecutive clock pulses.

2. The seed vector could be achieved with less number of clock cycles based on the above conditions. Test generation process strives to generate a test set which can detect all the targeted faults. Faults are usually of the same fault model, but a mix of faults fromvarious fault models can also be used[2]. The patterns in the test set are eventually proposed to detect defects occurring in the circuit. Test generation consists of two main steps: fault activation and fault propagation.

Fault activation: Fault activation sets the signal on a given line to a given value, depending on the fault model[3]. For stuck-at faults, an opposite value is applied and determine the fault.

Fault propagation: Fault effect propagation means to set proper values to the particular path inputs along with a selected path so that the path is sensitized and the fault effect can be observed at an observation point[4].

By test compaction or generation of compact tests, we mean the generation of minimal-sized test sets to achieve the desired fault coverage. Two methods exist to generate minimal test sizes. The first approach is to use extra hardware such as test points. Another method uses software techniques to produce minimal test set sizes[5]. In this work, we use software techniques. Several works have considered methods to generate small test sets.

2. RELATED WORK:

Hellebrand et al. A new pattern are generated based on LFSR which is capable of providinga numeral primitive polynomial. These polynomials are generated by using the feedback taps of the LFSR. The generator producing sequences will eliminate linear dependence practically. By using this property
to widen the encoding efficiency for test cubes to test faults will be difficult. So by reducing the linear dependences, many other applications are used.

C.L.Chen et al. Linear feedback Shift registers have been suggested to result in test patterns for the self-test of logic networks. The probability of linear dependences among k-bit positions of a subset of k- bits in a maximum length shift register sequence is an outstanding problem. To overcome this problem derived a problem for calculation of probability.

X.Lin and J.Rajski et al. The concept of chaining the sequential elements of the circuit together is to form shift register LFSR. Thus, by the proper selection of feedback taps with minimum possible taps save a lot of test time, Since the LFSR cells are used further for test data compression.

Ahmed et al. The critical role for polynomial seeds exists in LFSR based testing, whereas in pseudo-random test pattern generator and in signature analyzer any possible primitive polynomial combinations which are in same order can be used, any change in these polynomial seeds will not affect the level of the probability of aliasing errors.

Oscar Acevedo et al. In built-in test pattern generation for a test cube that is usually compressed by the initial state of LFSR. By using this seed vector, we can solve a linear set of equations using a fixed characteristic polynomial. By using the Berlekamp Massey algorithm, we can solve the equations, but it cannot work with don't care. Hence Berlekamp Massey with X(BMX) is an iterative procedure that deals with don't care.

J. Savir et al. The random sequences test length are short when compared to linear feedback shift register clock cycle length and it generates same patterns starting with a seed of LFSR, the produced random sequence constitute a short length. It is so advantageous to draw patterns uniformly from LFSR. A way for achieving this is by changing seeds every so often. Thus, new LFSR is controlled by two separate clocks one for normal LFSR operation and other to change the seeds.

3. METHODOLOGY:

A single pattern test is denoted by $a_i = <x_i, y_i, 1>$, where T_i denotes target faults set and A is an initial target in the multipattern test for a number of clock cycles. The multipattern test is denoted by $b_i = < c_i z_i, e_i>$. It is used to detect as many faults as possible from T_i.

To make sure whether a_i is viable in directing the creation of multipattern test, the process carries out fault simulation of T_i under a_i.Det_i denotes the set of detected faults. If Det_i is a null vector, the process does not continue to determine multipattern test based on $a_i$then a_i is not viable. If Det_i is not a null vector, the processcontinues as follows:

Not every predetermined value of $a_i = <x_i, y_i, 1>$ are used for fault detection. Only important values lead the creation of b_i without losing the detection of any fault from T_i and are useful for target fault detection. Thus, they can be used in directing the generation of b_i.

To calculate $b_i = < c_i, z_i, e_i>$, the process compute x_i and assigns $y_i = z_i$ and e_i=A. Let f_i be the scan in state that c_i produces.

The procedure stimulates T_i under $< c_i, z_i, e_i>$ and stores detected faults in D_{best} . Also, it calculates the hamming distance between p_i andx_i and stores it in H_{best}. The aim of changing y_i is to increase D_{best} and reduce H_{best}. The changing of b_i is done in 3 steps.

Step 1:To complement bits of c_i,

Step 2: To complement bits of z_i

Step 3: To complement bits ofe_i with different values from set {1,2,3………A_{max}}.

c_i stimulates T_i under b_i and stores the numeral detected faults in variable $det_i$and also calculates the hamming distance between p_i andx_i and stores it in H_i. Thus to allow the complementation of s_i the process requires either increase in det_i or reduction in H_i that means $det_i > D_{best}$ order$_i = D_{best}$ and $H_i = H_{best}$.

If this condition is satisfied, the process updates D_{best} and H_{best}. Otherwise, the process continues with previous values of c_i. Similar steps will be followed again to $z_i$and e_i, irrespective of H_{best}.

For e_i, the process considers different functional clock cycles represented by $e_i = A_{max}, A_{max}-1, \ldots 1$.

If $e_{new}$is not equal to e_i ,the process assigns $e_{new} = e_i$ and again same process repeat similar to step 1.

I_{MOD} represents the number of iterations of 3 steps is constant. After I_{MOD} iterations, the process returns the test b_i and number of faults it detected is D_{best}.

Here, we reduce the application test data time and test data volume using customized LFSR for combinational ATPG and sequential ATPG circuits.There are already existing standard LFSR but here we will reduce the application test data time and test data volume by selecting the appropriate seed value by using different characteristic polynomials. The seed value we have taken should reduce the number of clock cycles for an LFSR. The initial value which reduces as many clock cycles as possible is taken as a seed value for that LFSR. We have designed LFSR for 4 BIT,8 BIT and 16 BIT. The bit positions selected for use in the feedback function are called taps. The list of taps is called a tap sequence. Based on the tap sequence, the seed value changes for the LFSR. N bit LFSR will have different tap sequence that changes the value of the seed. So, the seed value mainly depends on the tap sequence of the LFSR. The characteristic polynomial is alsowritten from the tap sequence of the LFSR. The characteristic polynomial written from the LFSR that reduces as many faults as possible is used to reduce the test application time and test data volume.

A. External LFSR:

External LFSR is also called as Fibonacci LFSR or Many to One LFSR or standard LFSR. We can use either xor or XNOR gates for feedback in LFSR. A state with all ones should not be given as the seed value for XNOR based LFSR. It results in lockup state. Similarly, a state with all zeroes should not be given as the seed value for XOR based LFSR if given this also results in lockup state. The taps are XOR'd sequentially with the output bit and then feedback into the leftmost bit.

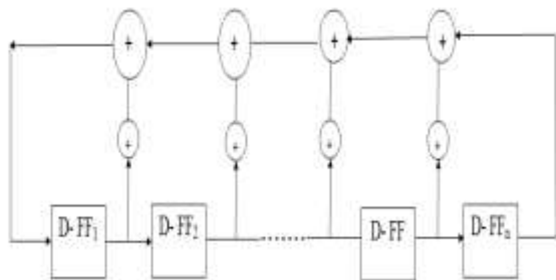


Fig:N bit External LFSR

B. Internal LFSR:

Internal LFSR is also called as Galois LFSR or One to Many LFSR or Modular LFSR. Here also we can use either xor or XNOR gates for feedback in LFSR. A state with all ones should not be given as the seed value for XNOR based LFSR. It results in lockup state. Similarly, a state with all zeroes should not be given as the seed value for XOR based LFSR if given this also results in lockup state.In the internal LFSR, when the system is clocked, bits that are not taps are shifted one position to the left unchanged. The taps, on the other hand, are XOR'd with the output bit before they are stored in the next position.

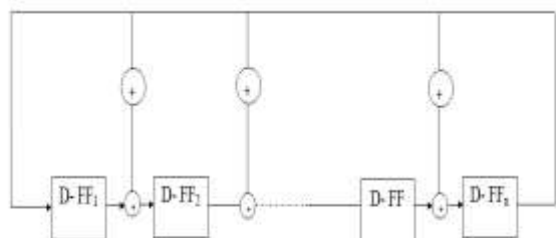


Fig:N-bit Internal LFSR

4. RESULTS:

The main purpose of LFSR with a limited number of bits is to achieve test data compression. Here we have selected seed values for LFSR that will reduce the test data time and data volume for 4 bit, 8 bit, 16 bit LFSR. The seed value will reduce themaximum number of clock cycles as well as it will detect as many faults as possible. Then we can determine the characteristic polynomial from that seed value. This process uses the seed value for LFSR to reduce test data volume and test application time.

A. Theoretical values for External LFSR:

4 Bit LFSR 4 to 16 decoder:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total clock cycles
0011	0111 1110	2nd 4th	SA1	X^4+X^2+1	5
1001	0111 1110 1101	2nd 3rd 4th	SA1	$X^4+X^3+X^2+1$	6
0110	1000 0001 0010	2nd 3rd 4th	SA0	$X^4+X^3+X^2+1$	6
0110	1000 0001 0010	2nd 3rd 4th	SA0	$X^4+X^3+X^2+1$	6
1010	1000 0001 0010	2nd 3rd 4th	SA0	$X^4+X^3+X^2+1$	6
1001	0010 0001 1000 0100	3rd 4th 5th 6th	SA0	X^4+X^3+1	6
1001	1110 0111	3rd 5th	SA1	X^4+X^3+1	5
1001	0010 0100 1000 0001	2nd 3rd 4th 5th	SA0	X^4+X+1	15
0110	0111 1110 1101 1011	2nd 3rd 4th 8th	SA1	X^4+X+1	15
1010	0100 1000 0001	2nd 3rd 4th	SA0	X^4+X^2+X+1	6
0101	1011 0111 1110	2nd 3rd 4th	SA1	X^4+X^2+X+1	6
1100	1000 0001	2nd 3rd	SA0	X^4+X^3+X+1	5
0110	1101 1011	2nd 3rd	SA1	X^4+X^3+X+1	3

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
1100	1001 0111 1111 1100	2 4 5 7	SA0	X^4+X^2+1	5
0100	1001 1110	2 5	SA0	$X^4+X^3+X^2+1$	6
0111	1111 1110 1100 1001	2 3 4 9	SA0	X^4+X^3+1	15
1011	1110 1001	3 5	SA0	X^4+X^2+X+1	6

1 TO 8 DEMUX:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
1000	0001 0010 0101 0110	2 3 4 6	SA1	$X^4+X^3+X^2+1$	6
1110	1111	1	SA0	$X^4+X^3+X^2+1$	6
1000	0001 0011 0111	2 3 4	SA1	X^4+X^3+1	5
1110	1111	1	SA0	X^4+X^3+1	5
1010	0100 0001 0011 0110	2 4 5 6	SA0	X^4+X^2+X+1	6
1000	0001 0010 0101 0100	1 2 3 5	SA0	X^4+X^2+1	5
1010	0101 0110 0010 0100 0001 0011 0111	1 3 6 7 9 10 11	SA0	X^4+X+1	15
1000	0001 0010 0100 0011 0110 0101 0111	2 3 4 6 7 10 12	SA0	X^4+X^3+1	15

2 TO 1 MUX:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
101	010	2	SA0	X^3+X^2+X+1	2
001	011	2	SA1	X^3+X^2+X+1	5
001	010	2	SA0	X^3+X^2+1	7
101	011	2	SA1	X^3+X^2+1	7

8 TO 3 DECODER:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic Polynomial	Total Clock cycles
01011111	10111111 01111111 11111110 11111101 11111011 11110111 11101111	2 3 5 6 7 8 9	SA1	$X^8+X^7+X^4+X^3+X^2+X+1$	51
01011111	10111111 11111101 11111011 11110111 11101111 11011111 01111111 111111110	2 4 5 6 10 11 53 55	SA1	$X^8+X^5+X^3+X^2+1$	255
00111111	01111111 11111110 11111101	2 4 6	SA1	$X^8+X^6+X^3+X+1$	21
01111101	11111011 11110111 11101111 10111111	2 3 4 5	SA1	$X^8+X^7+X^5+X^4+X^3+X+1$	15
01111011	11110111 11101111 01111111 11111110 11111101 11011111 10111111	2 3 12 14 15 113 114	SA1	$X^8+X^6+X^4+X+1$	217
10001000	00010000 00100000 01000000 00000010	13 14 15 17	SA0	$X^8+X^7+X^4+X^3+X^2+X+1$	51
100100000	00100000 01000000 10000000 00000001 00000010 00000100	2 3 4 5 6 36	SA0	$X^8+X^5+X^3+X^2+1$	255

	00001000 00010000	111 112			
10100 000	01000000 10000000 00000001	2 3 4	S A 0	$X^8+X^5+X^3+X+1$	21
10010 000	00100000 01000000 00000010 00000001	2 3 5 6	S A 0	$X^8+X^7+X^5+X^4+X^3+X+1$	15
10001 000	00010000 00100000 00000010 00000100 00001000 01000000 10000000 00000001	2 3 24 25 26 40 41 42	S A 0	$X^8+X^6+X^4+X+1$	217

7 1 TO 8 DEMUX:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
10000	00001 00010 00100 01000 00011 00110 01100 00101 01010 01011 01111	2 3 4 5 7 8 9 12 13 15 17	SA1	X^5+X^4+1	21
10101	01010 00110 01100 00010 00100 01000 00001 00011 00111 01111	2 4 5 8 9 10 12 13 14 15	SA1	X^5+X+1	21
11111	00100 01001 00110 01100 00001 00010 00101 01011 01101 01010 01000 00011 00111 01110 01111	6 7 9 10 13 14 15 16 18 21 23 25 27 31	SA1	$X^5+X^4+X^3+X^2+1$	32

B. Theoretical analysis for INTERNAL LFSR:

4 BIT LFSR for DECODER:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
0011	1011 1101	2 4	SA1	X^4+X^2+1	6
1111	1110 0111 1011 1101	2 3 6 15	SA1	X^4+X^3+1	15
1111	1110 0111	2 3	SA1	$X^4+X^3+X^2+X+1$	5
0001	1011 1110 0111	2 3 4	SA1	$X^4+X^3+X^2+1$	7
0101	1000 0100 0010 0001	2 3 4 5	SA0	X^4+X^2+1	6
0011	1000 0100 0010 0001	2 3 4 5	SA0	X^4+X^3+1	15
0011	1000 0100 0010 0001	2 3 4 5	SA0	$X^4+X^3+X^2+X+1$	5
0111	1000 0100 0010 0001	2 3 4 5	SA0	$X^4+X^3+X^2+1$	7

4 BIT LFSR FOR ENCODER:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
1011	1111 1100	2 4	SA0	X^4+X^2+1	6

0001	1001 1111 1110 1100	2 4 5 10	SA0	X^4+X^3+1	15
0011	1110 1100	2 4	SA0	$X^4+X^3+X^2+X+1$	5
0101	1001 1111 1100	2 3 6	SA0	$X^4+X^3+X^2+1$	7

1 TO 8 DEMUX:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
1010	0101 0100 0010 0001	2 4 5 6	SA0	X^4+X^2+1	6
1110	0111 0101 0110 0011 0100 0010 0001	2 4 7 8 10 11 12	SA0	X^4+X^3+1	15
1100	0110 0011 0111	2 3 5	SA0	$X^4+X^3+X^2+X+1$	5
1010	0011 0110 0101	4 5 7	SA0	$X^4+X^3+X^2+1$	7

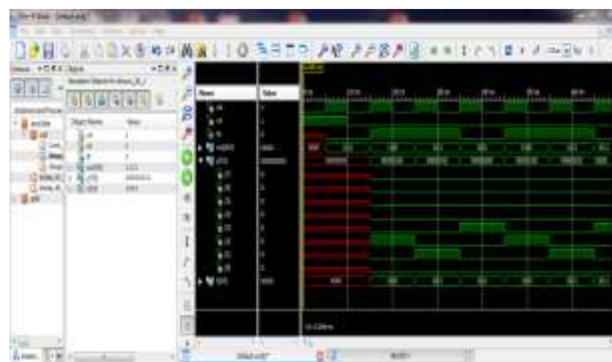
8 BIT LFSR FOR DECODER:

Seed	$t_{(faulty)}$	Clock cycle	Fault	Characteristic polynomial	Total Clock cycles
01000101	11110111 11111110 01111111 11101111 11111101 10111111	2 5 6 9 12 24	SA1	$X^8+X^7+X^5$ $+X^3+X+1$	25
11011101	11110111 11111110 01111111 11101111 11111101 10111111 11011111 11111011	12 35 48 56 89 114 157 198	SA1	$X^8+X^6+X^4$ $+X^3+X^2+1$	255
10101001	11111110 01111111	2 3	SA1	$X^8+X^6+X^4+X^2$ $+1$	10
01000111	11111110 01111111	2 3	SA1	$X^8+X^7+X^5+X^4$ $+$ X^3+X+1	15
10101011	10000000 01000000 00100000 00010000 00001000 00000100 00000010 00000001	2 3 4 5 6 7 8 9	SA1	$X^8+X^7+X^5$ $+X^3+X+1$	25
11111111	10000000 01000000 00100000 00010000 00001000 00000100 00000010 00000001	2 3 4 5 6 7 8 9	SA1	$X^8+X^6+X^4+X^3$ $+X^2+1$	255
01010101	10000000 01000000 00100000 00010000 00001000 00000100 00000010 00000001	2 3 4 5 6 7 8 9	SA1	$X^8+X^6+X^4+X^2$ $+1$	10
10111011	10000000 01000000 00100000 00010000 00001000 00000100 00000010 00000001	2 3 4 5 6 7 8 9	SA1	$X^8+X^7+X^5+X^4$ $+X^3+X+1$	15

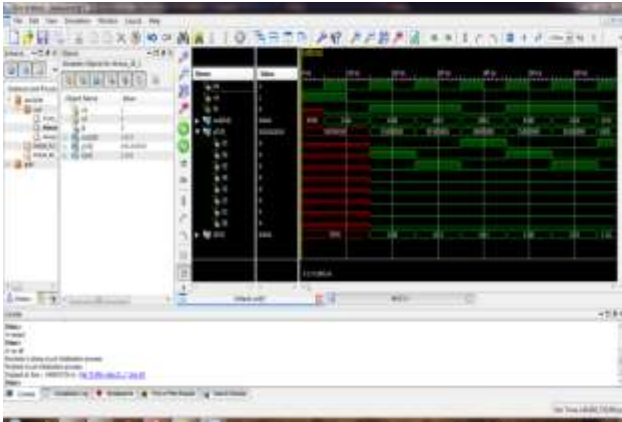
5. SIMULATION RESULTS:

The simulation results for the 4 BIT, 8 BIT, 16 BIT LFSR are represented below. Here the initial value that is having less number of functional clock cycles value is taken as a seed value and the seed value is able to find out as many faults as possible. Therefore by using the seed value of LFSR, we can reduce the test application time and also test data volume by reducing a maximum number of clock cycles.

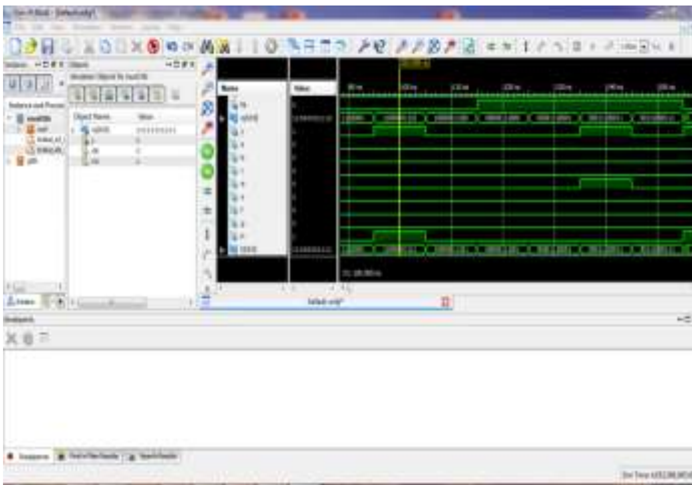
A. INTERNAL 4 BIT DECODER FOR SA 0 FAULT AT T2:



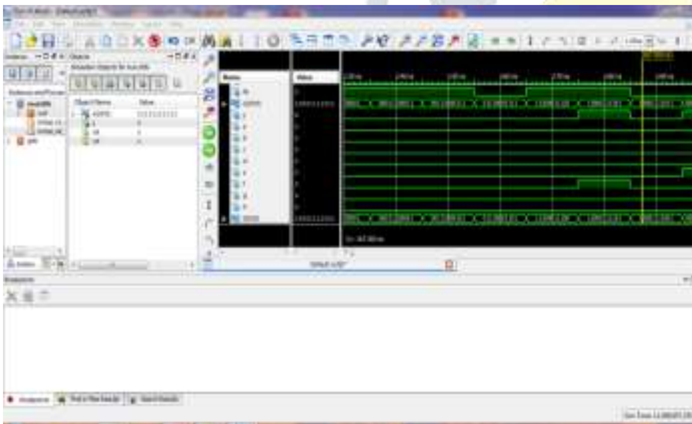
INTERNAL 4 BIT DECODER FOR SA 1 FAULT AT T2:



EXTERNAL LFSR FOR 8X1 MUX FOR SA1 AT T0:



EXTERNAL LFSR FOR 8X1 MUX FOR SA0 AT T0:



6. CONCLUSION:

This paper describes that by using customized LFSR we can reduce the test application time and test data volume. Here we will find the appropriate seed value of the LFSR that reduces the maximum number of clock cycles and at the same time, it should be able to give efficient fault coverage.

7. REFERENCES:

[1] I.Pomeranz "LFSR-Based Generation of Multicycle Tests," IEEE transactions on computer-aided design of integrated circuits and systems, vol. 36, no. 3, March 2017

[2] A. Chandra, J. Saikia, and R. Kapur, "Breaking the test application time barriers in compression: Adaptive scan-cyclical (AS-C)," in *Proc. AsianTest Symp.*, New Delhi, India, 2011, pp. 432–437.

[3] O. Acevedo and D. Kagaris, "Using the Berlekamp–Massey algorithmto obtain LFSR characteristic polynomials for TPG," in *Proc. Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Austin, TX, USA, 2012,pp. 233–238.

[4] X. Lin and J. Rajski, "On utilizing test cube properties to reduce testdata volume further," in *Proc. Asian Test Symp.*, Niigata, Japan, 2012, pp. 83–88.

[5] T. Moriyasu and S. Ohtake, "A method of one-pass seed generationfor LFSR-based deterministic/pseudo-random testing of static faults," in*Proc. Latin-Amer. Test Symp.*, Puerto Vallarta, Mexico, 2015, pp. 1–6.

[6] Rajski, J., Moghaddam, E.K., Reddy, S.M.: 'Low power compressionutilizing clock-gating'. Proc. Int. Test Conf., 2011, pp. 1–8

[7] Tenentes, V., Kavousianos, X.: 'Test-data volume and scan-powerreduction with low ATE interface for multi-core SoCs'. Proc. Int.Conf. Computer-Aided Design, 2011, pp. 747–754

[8] Tenentes, V., Kavousianos, X.: 'Low power test-compression for hightest-quality and low test-data volume'. Proc. Asian Test Symp., 2011,pp. 46–53

[9] Chandra, A., Saikia, J., Kapur, R.: 'Breaking the test application timebarriers in compression: adaptive scan-cyclical (AS-C)'. Proc. AsianTest Symp., 2011, pp. 432–437

[10] Liu, X., Xu, Q.: 'On X-variable filling and flipping for capture-powerreduction in linear decompressor-based test compressionenvironment', IEEE Trans. Comput. Aided Des., 2012, 31, (11),pp. 1743–1753