ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Low Power Secure and Efficient LBIST for Feedback Shift Register-Based Cryptographic Systems

P. Sampath Kumar
Associate professor & HOD,
Department of ECE,
Malla Reddy Institute of Technology, Hyderabad.

T.Vimala
M.Tech (VLSI &ES),
Department of ECE,
Malla Reddy Institute of Technology, Hyderabad.

Abstract:

Cryptographic methods are used to protect confidential information against unauthorised modification or disclosure. Cryptographic algorithms providing high assurance exist, e.g. AES. However, many open problems related to assuring security of a hardware implementation of a cryptographic algorithm remain. Security of a hardware implementation can be compromised by a random fault or a deliberate attack. The traditional testing methods are good at detecting random faults, but they do not provide adequate protection against malicious alterations of a circuit known as hardware Trojans. For example, a recent attack on Intel's Ivy Bridge processor demonstrated that the traditional Logic Built-In Self-Test (LBIST) may fail even the simple case of stuck-at fault type of Trojans. In this paper, we present a novel LBIST method for Feedback Shift Register (FSR)- based cryptographic systems which can detect such Trojans. In this we also used CUT(circuit under Test) to test the particular circuit. The specific properties of FSR-based cryptographic systems allow us to reach 100% single stuck-at fault coverage with a small set of deterministic tests. The test execution time of the proposed method is at least two orders of magnitude shorter than the one of the pseudo-random pattern-based LBIST. Our results enable an efficient protection of FSR-based cryptographic systems from random and malicious stuck-at faults.

I. INTRODUCTION:

Feedback Shift Register (FSR) based cryptographic systems are the fastest and the most power-efficient cryptographic systems for hardware applications [1].

The speed and the power are two crucial factors for future cryptographic systems, since they are expected to support very high data rates in 5G ultra-low power products and applications. A hardware fault can compromise the security of a cryptographic system. For example, suppose that the output of a pseudo-random pattern generator used in a stream cipher is stuck to the logic 0. A stream cipher encrypts a message by combining it with a pseudo-random pattern, typically by a bit-wise addition modulo 2. Therefore, if the pseudo-random pattern is all-0, the message is sent unencrypted. To make possible periodic fault detection in functional circuits during their lifetime, cryptographic systems often employ Logic Built-In Self-Test (LBIST). However, as shown by a recent attack on Intel's cryptographically secure Random Number Generator (RNG) used in the Ivy Bridge processors [2], traditional LBIST techniques have a limited use against malicious alterations of the original circuit known as hardware Trojans. This is not only due to the fact that a Trojan can be inserted into the LBIST itself, but also because the Trojan can be designed not to trigger the LBIST, since LBIST usually detects only a subset of all possible faults. In this paper, we present a new method for LBIST which makes possible detecting stuck-at fault type of Trojans. The presented method specifically targets FSR-based cryptographic systems. First, we use a deterministic test set which covers 100% of single stuck-at faults in the circuit under test, Test Pattern Generator (TPG) and Test Response Analyser (TRA). Second, we do not compact output responses into a Multiple Input Signature Register (MISR) signature. So, an attack based on selecting suitable values for the Trojan which generate the correct MISR signature for the inputs provided during the LBIST becomes impossible in our case. Furthermore, Trojans inserted into the LBIST circuitry itself (TPG and TRA) will be detected.

The presented method is similar to the traditional scan design in that is provides a simple way of setting and observing each flip-flop in a circuit. However, unlike in the case of scan, we do not connect flip-flops in scan chains. Instead, to support a test mode, we multiplex the input of cells which serve as state variables for the feedback functions and put a switch at the output of cells which correspond to outputs to non-trivial feedback functions. This allows for loading and unloading of flip-flops' contents during the test mode. The size and the number of Boolean functions used in cryptographic systems are typically considerably smaller than the size of an FSR. Therefore, the presented approach has small area overhead. Furthermore, our technique does not affect the propagation delay of the original circuit. In the traditional scan, the propagation delay is always increased by the delay of a multiplexer.

Boolean functions used in cryptographic systems are commonly represented in Algebraic Normal Form (ANF) [3]. It was shown by Reddy [4] that a combinational logic circuit implementing an n-variable Boolean function represented in ANF can be tested for all single stuck-at faults using at most 3n+4 tests. We use Reddy's result as a base to derive a minimal complete test set for single stuck-at faults for combinational logic circuits implementing feedback functions of an FSR. The specific properties FSR-based cryptographic systems allow us to reach 100% single stuck-at fault coverage with a test set of size at most (k+2)×(k+3) bits, where k+1 is largest number of variables on which any feedback function of the FSR depends. The expected output responses can be stored using at most (k+3)×m bits, where m is the number of non-trivial feedback functions of the FSR.

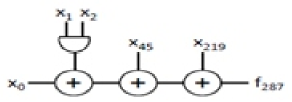


Fig. 1: The logic circuit implementing ANF of the feedback function f287 of Trivium.

The paper is organized as follows. Section II summarises basic notations used in the sequel. Section III describes the presented method. Section IV gives details of procedures used for fault detection. In Section V, the presented method is demonstrated on the example of Trivium stream cipher. Section VI concludes the paper and discusses open problems.

II. PRELIMINARIES:

We use the standard notation from the areas of testing and logic synthesis. For a more detailed description, the reader is referred to [5] and [6].

A. Algebraic normal form of Boolean functions:

An n-variable Boolean function f (x1, . . . ,xn) is a mapping of type f : {0,1}n {0,1}.
The dependence set [5] of a Boolean function f is defined by
dep(f) = {i | f |xi=0 ≠ f |xi=1},
where f |xi=j = f (x0, . . . ,xi−1, j,xi+1, . . . ,xn−1) for j 2 {0,1}.
Any n-variable Boolean function has a unique Algebraic Normal Form (ANF) [3] (also called Reed-Muller canonical form [7]) which is a representation of type:

f (x1, . . . ,xn)= $\sum_{i=0}^{2^n-1} c_i \cdot x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n}$
where ci 2 {0,1} are constants, "•" stands for the Boolean AND and å stands for the Boolean XOR. The vector (i1i2 . . . in) is the binary expansion of i with i1 being the least significant digit, and the term xi j
j denotes the i jth power of the variable xj, j 2 {1, . . . ,n}. ANF is a common representation for Boolean functions
used in cryptographic systems [3]. To be hardware efficient , cryptographic systems typically use ANF of a small size. For example, consider the stream cipher Trivium [8]. It is defined by a 287-bit Feedback Shift Register (FSR) in which all but 3 out of 287 of functions are trivial functions of type fi = xi+1.

The remaining 3 functions are:
f287 = x0 x1x2xorx45xorx219
f194 = x195xorx196x197xorx117xorx222
f110=x111xorx112x113xorx24xorx126 (2)
The primary output of Trivium is computed by adding the values from cells 110, 194 and 287:
fout put = f287xor f194xor f110. We can see that functions f287, f194 and f110 use only 15 out of 287 possible state variables in their ANFs in total.
As another example, consider the 128-bit FSR used in the stream cipher Grain-128 [9]. All its feedback functions except the function f127 are of type fi = xi+1. The function f127 is given by:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

f127=x0xorx26xorx56xorx91xorx96xorx3x67xorx11x13 xorx17x18xorx27x59xor x40x48xorx61x65xorx68x84
This function uses 19 out of 128 possible state variables.

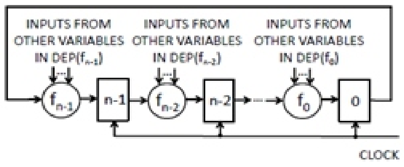

INPUTS FROM OTHER VARIABLES IN DEP(f_{n-1})
INPUTS FROM OTHER VARIABLES IN DEP(f_{n-2})
INPUTS FROM OTHER VARIABLES IN DEP(f_0)
f_{n-1}
n-1
f_{n-2}
n-2
f_0
0
CLOCK

Fig. 2: The general structure of an n-bit FSR.

From the examples above, the reader may see that none of ANFs uses the same variable twice. Furthermore, the same variable does not occur in more than one ANF. In addition, the same index is not used as both input and output, i.e. if fi is non-trivial, then the state variable xi is not used. These typical features of ANFs used in cryptographic systems follow from the requirements for the cryptographic security of Boolean functions [3].

Any n-variable Boolean function represented in ANF can be implemented by a logic circuit consisting of a linear cascade of two-input XOR gates fed by AND gates, one corresponding to each product-term of the expression (1) with a non-zero constant ci, i 2 {1, . . . ,2n−1}. For example, the function f287 of Trivium can be implemented by a circuit shown in Fig. 1.

B. Feedback Shift Registers

An n-bit Feedback Shift Register (FSR) [10] consists of n binary storage elements, called cells or stages (see Fig. 2). Each cell i 2 {0,1, . . . ,n−1} has an associated state variable xi 2 {0,1} which represents the current value of the cell i and an feedback function fi : {0,1}n !{0,1} of type

fi(x0,x1, . . . ,xn−1) = xi+1xorgi(x0,x1, . . . ,xn−1)

which determines how the value of i is updated, where ”+” is modulo n. If gi = 0, fi is called trivial. The variable xi+1 of fi is called the free variable.

A cell with index i such that i€dep(f j) for some non-trivial function f j, i, j € {0,1, . . . ,n−1}, is called an controllable cell.

A cell with index j such that f j is non-trivial, j €{0,1, . . . ,n−1}, is called an observable cell.

A cell with which is neither controllable cell, nor observable cell, is called an internal cell.

The state of an FSR is a binary vector of values of its state variables (x0,x1, . . . ,xn−1). At every clock cycle, the next state is determined from the current state by updating the values of all cells simultaneously to the values of the corresponding feedback functions.

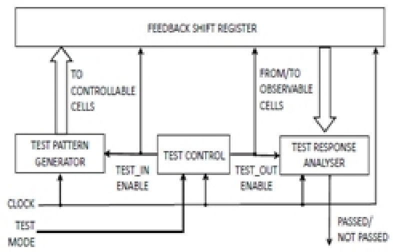


Fig. 3: A block diagram illustrating the presented method.

C. Logic Built-In Self Test:

The traditional LBIST typically employs a Linear FSR (LFSR) to generate pseudo-random test patterns that are applied to the circuit under test and a Multiple Input Signature Register (MISR) for obtaining the compacted response of the circuit to these test patterns [11]. An incorrect MISR output indicates a fault in the circuit. Various techniques can be used to complement pseudo-random test patterns [12], [13]. A problem with the traditional LBIST is that many pseudorandom patterns (several thousands or more) need to be applied to reach a satisfactory fault coverage. This implies that test execution time can be too long for some applications [14].

III. THE PRESENTED METHOD:

The method presented in this paper is similar to the traditional scan design in that is provides a simple way of setting and observing each flip-flop in a circuit. However, unlike in scan, we do not connect flip-flops in scan chains. Instead, to support the test mode, we modify the original FSR as follows (see Fig. 3):1) We multiplex the input of each controllable cell as shown in Fig. 4(b). The input of the original flipflop becomes the functional input of the multiplexer (MUX). The test input of MUX is connected to the Test Pattern Generator (TPG).

2) We duplicate the output of each observable cell as shown in Fig. 4(c). The duplicated output is connected to the Test Response Analyser (TRA) through a switch. When the test mode is selected, the flip-flops with multiplexed inputs become inputs to the combinational logic. The flip-flops which have a switch on the output become outputs of the combinational logic. As in a scan design, this increases controllability and observability, making possible testing a sequential circuit with tests for combinational logic.

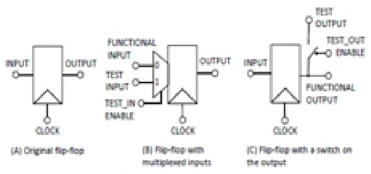


Fig. 4: Modifications of FSR flip-flops to support test mode.

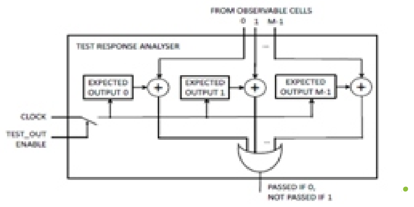
Note that such a technique does not affect the propagation delay of the original circuit. In the traditional scan, the propagation delay is always increased by the delay of a MUX. We add MUXes only to the controllable cells, whose feedback functions are trivial. Therefore, the propagation delay is still determined by the observable cells, whose feedback functions are non-trivial.
The following signals are added to the FSR to control and observe its cells:
1) Test in enable signal controls the application of test vectors. When it is asserted, controllable cells are connected to the TPG and TPG is connected to the clock. Otherwise, controllable cells are connected to their predecessor cells and TPG is not connected to the clock.
2) Test out enable signal controls output response analysis. When it is asserted, observable cells are connected to both the TRA and their successor cells and TRA is connected to the clock. Otherwise, observable cells are connected to their successor cells only, and TRA is not connected to the clock.The comparison of expected and computed responses can be done using TRA shown in Fig. 5. For each observable cell i, the value at the test output is compared to the expected value of fi using an XOR gate. The outputs of all XORs are fed into an OR gate the output of which indicates the presence/absence of a fault.

IV. FAULT DETECTION:

In this section, we show that it is possible to detect all single stuck-at faults in an FSR whose feedback functions are represented in ANF using a test set of size (k+2)×(k+3) bits, where k is size of the largest dep(gi), i 2 {0,1, . . . ,n−1}. We target cryptographic systems, therefore we assume that the feedback functions satisfy the following two properties:
1) If xi€dep(gj), then xi 62 dep(gk), for any i, j,k 2 {0,1, . . . ,n − 1}, j ≠k. This means that the same variable does not occur more than once in ANFs of non-trivial functions.
2) If gi ≠0, then xi £ dep(gj), for any i, j € {0,1, . . . ,n− 1}, i ≠j. This means that, the same cell is not used as both input an output of non-trivial functions.

A. Detecting faults in the combinational logic:

Suppose that each non-trivial function fi, i € {0,1, . . . ,n−1}, is implemented by a logic circuit consisting of a linear cascade of two-input XOR gates fed by AND gates, one corresponding to each product-term of the ANF with a nonzero constant ci, i € {1, . . . ,2n−1}. Let the size of the largest dependence set of functions fi be k + 1. For example, for Trivium, the size of the largest dependence set is 5. For Grain-128, the size of the largest dependence set is 19.



The TPG has k +2 outputs 0E,0D,1,2, . . . ,k which are connected to the test inputs of controllable cells of the FSR as follows. For all gi ≠0, i € {0,1, . . . ,n−1}:
1) If the number of product-terms in the ANF of fi is even, the output 0E of the TPG is connected to the cell i+1. Otherwise, the output 0D of the TPG is connected to the cell i+1, where ”+” is modulo n.
2) If |dep(gi)| xor j, then the output j of the TPG is connected to the cell corresponding to the jth variable in dep(gi), for j € {1,2, . . . ,k}. The TPG generates a test set T = T1[T2 of size (k+3)×(k+2) bits which is constructed as follows.The set T1 consists of three tests listed in the table below.

Test set T_1

	Outputs of TPG						Expected
	$0E$	$0D$	1	2	…	k	output responses
t_1	0	0	0	0	…	0	0
t_2	0	1	1	1	…	1	1
t_3	1	0	1	1	…	1	0

T1 detects all single stuck-at faults at the inputs and outputs of all XOR gates because it applies both 0 and 1 to every input and output of each XOR gate and an XOR cascades always propagates any change to its outputs. Either of 2nd and 3rd tests also detects all stuck-at-0 faults at the inputs of all AND gates, since its set all inputs of all AND gates to 1s.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The set T2, consists of k tests listed in the table below. The test ti+3, i∈{1,2, . . . ,k}, sets ith output of the TPG to 0 and all other outputs j ∈ {1,2, . . . ,k}, j ≠i, to 1. T2 detects all single stuck-at-1 faults on the inputs of all AND gates. In general, the values set to 0E and 0D do not matter for the detection of faults. We set them to 0 and 1, respectively, to make the output response the same for the cases of ANF with and even and odd number of product-terms.

Test set T_2

	Outputs of TPG						Expected output responses for	
	0E	0D	1	2	...	k	$\lvert dep(f_i)\rvert \geq j$	$\lvert dep(f_i)\rvert < j$
t_4	0	1	0	1	...	1	0	1
t_5	0	1	1	0	...	1	0	1
...			...				...	
t_{k+3}	0	1	1	1	...	0	0	1

For arbitrary Boolean functions, it is also necessary to detect faults on the inputs of the logic circuit implementing ANF by sensitizing an odd number of paths from each input through the AND gates to the output of the circuit. Since XOR gates are modulo 2 adders, an even number of changes at the input of an XOR cascade cancel out and do not cause a change on the output. However, since we made an assumption that no variable occurs more than once in ANFs of non-trivial functions, in our case only one path is sensitized by a change at some input. Therefore, no additional tests are required for fault detection on inputs.
To summarise, the set T = T1 [T2 of k+3 tests detects all single stuck-at faults in the combinational logic implementing all non-trivial feedback functions of an FSR. It also detect all single stuck-at faults:
1) At the test input of each controllable cell.
2) At the output of each controllable cell.
3) At the input of each observable cell.
4) At the output of each observable cell which is connected to the TRA.
The detection of faults can be carried out using the following procedure.

Procedure 1:

1) Assert the test in enable signal to connect test inputs of controllable cells controllable cells to the TPG.
2) Apply one clock to load the test vector ti 2 T, i ∈{1,2, . . . ,k+3}, from the TPG to all controllable cells in parallel.
3) Apply one clock to evaluate the non-trivial feedback functions for the input assignment defined by ti. The resulting output responses are captured at the observable cells. At the same clock cycle, the next test vector ti+1 from T is loaded from the TPG to the controllable cells.
4) Assert the test out enable signal to connect test outputs of observable cells to the TRA.
5) Apply one clock to upload the responses to ti from all observable cells to the TRA in parallel. The TRA compares the computed responses to the expected responses. If they agree, TRA outputs ”passed”. Otherwise, it outputs ”not passed”. At the same clock cycle, non-trivial feedback functions are evaluated for the input assignment defined by ti+1. The resulting output responses are captured at the observable cells. The next test vector ti+2 from T is loaded from TPG to the controllable cells.
6) Repeat the steps 2, 3 and 5 until all test vectors from T are applied.
The Procedure 1 completes the application of all tests from T and evaluation of all output responses in k+5 clock cycles.

B. Detecting remaining faults in FSR:

The set T does not detect stuck-at faults at internal cells. To detect these faults, we use the tests t1 and t2 of the test set T1 and the Procedure 2 described below.
Let I = {i1, i2, . . . , i|I|}, where i j ∈ {0,1, . . . ,n−1} for j 2{0,1, . . . , |I|}, be the union of dependence sets of all non-trivial functions:
I = {i | i ∈dep(fi)^(gi 6= 0)}.
Suppose that I is ordered as i1 > i2 > . . . > i|I|. Then the maximum distance between two controllable cells is defined by d = max(i j −i j+1)
for all i j ∈ I, j ∈ {1,2, . . . , |I|}, where ”+” is modulo n.
For example, for Trivium, d =69 (between the controllable cells 195 and 126). For Grain-128, d = 32 (between the controllable cells 0 and 96).

Procedure 2:

1) Set the test out enable signal low to disconnect test outputs of observable cells from the TRA.
2) Assert the test in enable signal to connect test inputs of controllable cells to the TPG.
3) Apply one clock to load the test vector t1 2 T1 from the TPG into all controllable cells in parallel.
4) Repeat d−1 times: Apply one clock to evaluate the non-trivial feedback functions for the input assignment defined by t1. The resulting output responses are captured at the observable cells. All internal cells capture the value of their predecessors. At the same clock cycle, the same test vector t1 from T is loaded again from the TPG to the controllable cells.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

5) Set the test in enable signal low to connect functional inputs of controllable cells to their predecessors.
6) Apply one clock to capture the value of the predecessors of controllable cells into the controllable cells.
7) Apply one clock to evaluate the non-trivial feedback functions for the input assignment defined by the controllable cells. The resulting output responses are captured at the output flip-flops.
8) Assert the test out enable signal to connect test outputs of observable cells to the TRA.
9) Apply one clock to upload the responses from all observable cells to the TRA in parallel. The TRA compares the computed responses to the expected responses. If the two responses agree, TRA outputs ”passed”. Otherwise, it outputs ”not passed”.
10) Repeat the steps 1-9 for the test vector t2 € T1.
The Procedure 2 completes the application of tests and evaluation of all output responses in 2d+6 clock cycles.

To detect stack-at-1 faults, all-0 test vector t1 is applied from the TPG. During the step 4 of Procedure 2, value 0 which is computed as the single fault occurred in a cell which is not a controllable cell, all other inputs on which this observable cell depends have 0 value and cannot cancel out the change. Therefore, at step 9, the change 0/1 will propagate to the TRA and will be detected.The detection of stuck-at-0 faults is more complicated since the function values differ for ANFs with an even and an odd number of product-terms. If the ANF has an odd number of product-terms, we can set the function to 1 by setting all its input variables to 1. If the ANF has an even number of productterms, we can set the function fi to 1 by setting all its input variables except xi+1 to 1.To be able to set different values to the controllable cells xi+1 of different functions fi, we need to use two outputs of TPG - one for functions whose ANF has an even number of product-terms, 0E, and the other for functions whose ANF has an odd number of product-terms, 0D.To detect stack-at-0 faults, the test vector t2 is applied from the TPG. During the step 4 of Procedure 2, the value 1 which is computed at the observable cells as a response to t2 is shifted from the observable cells through the chains of internal cells. In at most d –1 clock cycles, all internal cells are set to the 1 value. Suppose that a single stuck-at-0 fault occurs at some cell j which is not a controllable cell. During the step 4 of Procedure 2, this change will propagate to the predecessor of the nearest controllable cell i after the cell j.

At step 6, the change 1/0 will shift to the cell i. At step 7, the change 1/0 will propagate to the observable cell k which depends on the cell i.At the observable cell j, the change 1/0 can be potentially cancelled out only if the variable xi occurs in the ANF of fk in a product-term containing one or more other variables which have values 0. However, this is not possible since the only input variables which are loaded with 0 from the TPG are free variables. Therefore, at step 9, the change 1/0 will be propagated to the TRA and detected.observable cells as a response to t1 is shifted from the controllable cells through the chains of internal cells. In at most d clock cycles, all cells in the FSR are set to the 0 value. Suppose that a single stuck-at-1 fault occurs at some cell j which is not a controllable cell. During the step 4 of Procedure 2, in at most d–1 clock cycles the change 0/1 will propagate to the predecessor of the nearest controllable cell i after the cell j. At step 6, the change 0/1 will shift to the cell i. At step 7, the change 0/1 will propagate to the observable cell which depends on cell i. Since we assumed that a

C. Detection of faults in the TPG:

Consider the case when a single stuck-at fault occurs of the output j of the TPG, j 2 {0E,0D,1, . . . ,k}. Such a fault will manifest itself as a multiple stuck-at fault at the controllable cells connected to the output j. Since none of the state variables occurs in more than one ANF, each faulty input will affect only one fi. Therefore, the change in values caused by the fault will not be canceled out and the fault will be detected by the Procedure 1.

D. Detection of faults in the TRA :

TRA stores the expected responses to the tests and compares them to the computed responses. We have shown in the previous Section that for T2 the expected responses may differ for functions whose dependence set have different sizes. In the worse case, all non-trivial functions may have dependence sets of different sizes. Then, in order to store the expected responses, we need (k+3)×m bits, where m is the number of non-trivial feedback functions.Note that the TRA circuit shown in Fig. 5 handles not only single stuck-at faults in the FSR, but also single stuck-at faults which occurs in the TRA itself, except the stuck-at-0 and stuck-at-1 fault at the output of the OR gate. To allow for detection of these faults, the OR gate can be duplicated.

ISSN No: 2348-4845
International Journal & Magazine of Engineering,
Technology, Management and Research
A Peer Reviewed Open Access International Journal

V. PROPOSED MODEL WITH CUT(CIRCUIT UNDER TEST):

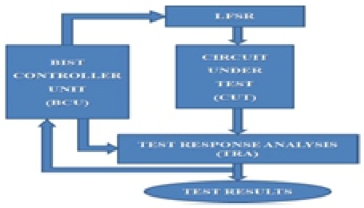


Fig 6. Presented method with CUT.

Circuit Under Test (CUT):

It is the portion of the circuit tested in BIST mode. It can be sequential,

Test Response Analysis (TRA):

It analyses the value sequence on PO and compares it with the expected output.

BIST Controller Unit (BCU):

It controls the test execution; it manages the TPG, TRA and reconfigures the CUT and the multiplexer.

Test Pattern Generator (TPG):

It generates patterns for the CUT. It is a dedicated circuit or a microprocessor. The patterns may be generated in pseudorandom or deterministically.

VI. SIMULATION RESULTS:

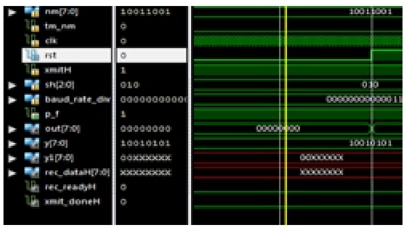


Fig 7. Simulation Result for top module (rst=0,tm_nm=0)

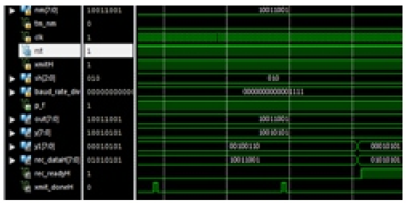


Fig 8. Simulation Result for top module (rst=1,tm_nm=0)

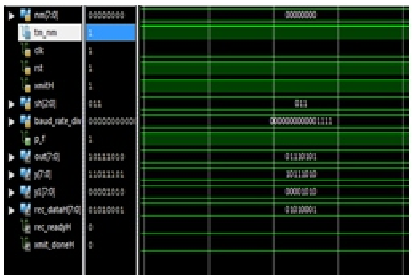


Fig 9. Simulation Result for top module (rst=1,tm_nm=1)

combinational or a memory. It is delimited by their Primary Input (PI) and Primary Output (PO).

VII. CONCLUSION:

To summarize, in k+2d +11 clock cycles, we can detect all single stuck-at faults in FSR, TPG and TRA using the test set of size at most (k +2)×(k +3) bits and a set of expected output responses of size at most (k +3)×m bits, where k + 1 is largest number of variables on which any feedback function of the FSR depends, m is the number of non-trivial feedback functions, and d is the largest distance between any two controllable cells.
In presented method we also used circuit under test to test the particular circuit with such Trojans.
The presented method has the following advantages compared to the traditional pseudo-random pattern-based LBIST using scan:
1) It causes no performance degradation.
2) It requires a small set of deterministic tests to cover 100% of single stuck-at faults. Thus, the test execution time is much shorter (at least two orders of magnitude).
3) It has a higher resistance against stuck-at fault type of hardware Trojans.

ACKNOWLEDGEMENT:

The first author was supported in part by the research grant No SM12-0005 from the Swedish Foundation for Strategic Research.

REFERENCES:

[1] T. Good and M. Benaissa, "ASIC hardware performance," New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986, pp. 267–293, 2008.

[2] G. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopantlevel hardware Trojans," Proceedings of Cryptographic Hardware and Embedded Systems (CHES'2013), LNCS 8086, pp. 197–214, 2013.

[3] T. W. Cusick and P. Stˇanicˇa, Cryptographic Boolean functions and applications. San Diego, CA, USA: Academic Press, 2009.

[4] S. Reddy, "Easily testable realizations for logic functions," IEEE Transactions on Computers, vol. 21, no. 11, pp. 1183–1188, 1972.

[5] R. K. Brayton, C. McMullen, G. Hatchel, and A. Sangiovanni- Vincentelli, Logic Minimization Algorithms For VLSI Synthesis. Kluwer Academic Publishers, 1984.

[6] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. Jon Willey and Sons, New Jersey, 1994.

[7] D. H. Green, "Families of Reed-Muller canonical forms," International Journal of Electronics, vol. 70, pp. 259–280, 1991.

[8] C. Canni`ere and B. Preneel, "Trivium," New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986, pp. 244–266, 2008.

[9] M. Hell, T. Johansson, A. Maximov, andW. Meier, "The Grain family of stream ciphers," New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986, pp. 179–190, 2008.

[10] S. Golomb, Shift Register Sequences. Aegean Park Press, 1982.

[11] E. McCluskey, "Built-in self-test techniques," IEEE Design and Test of Computers, vol. 2, pp. 21–28, 1985.

[12] H.-J. Wunderlich, "BIST for systems-on-a-chip," Integration, the VLSI Journal, vol. 26, no. 1-2, pp. 55 – 78, 1998.

[13] K. Chakrabarty, "Modular testing and built-in self-test of embedded cores in system-on-chip integrated circuits," in The Embedded Systems Handbook (R. Zurawski, ed.), pp. 27–2–27–27, CRC Press, 2006.

[14] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: real issues and case studies," in Proceedings of International Test Conference (ITC'1999), pp. 358 – 367, 1999.

[15] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," IEEE Design Test of Computers, vol. 27, no. 1, pp. 10–25, 2010.