ISSN (e): 2250 – 3005 || Volume, 07 || Issue, 09|| September – 2017 ||
International Journal of Computational Engineering Research (IJCER)

Application Dependent Testing of S-Ram Based on Test Pattern Generator

[1]S.Lekashri
Assistant Professor, Department of Electronics and Communication Engineering,
Kings Engineering College, Irungattukottai, Sriperumbudur, Chennai
Corresponding Author: [1]S.Lekashri

ABSTRACT
This paper presents an Application-dependent testing of a SRAM-based FPGA interconnects. Here is a method that associates a turn on inputs to numerous nets, which gives rise to test vectors to determine stuck-at, open, and bridging faults. This set up gives us privilege in reducing unnecessary composition that reduces the testing time for application-dependent testing for coverage of faults. As the basic result is in need of a growing complication, polynomial algorithm which is heuristic and acquisitive in nature (based on sorting) and it is used for selecting the nets to be used in the process of generating configurations.

Keywords: Circuit faults, Configuration generation, Field programmable gate array, Look up Table, Test configurations, Test Vectors

---

Date of Submission: 26-01-2017 Date of acceptance: 09-09-2017

I. INTRODUCTION

BIST is an emerging technique for testing complex VLSI systems. To test a design by using a BIST methodology, the design has to be modified (enhanced) in such a way that part of the circuit is used to test the design itself. Therefore, BIST is defined as a DFT technique in which testing is accomplished through built-in hardware components [2], [3]. A general BIST scheme is shown in Figure 2. It consists of a test source block, the CUT, a test response analysis block and a test controller block, which manages the application of the tests. In a classical BIST scheme, the test source consists of a special kind of register, test pattern generator (TPG), which generates on-chip test patterns. Recently, a new hybrid BIST approach [8] has been proposed. It enhances the design with a read only memory (ROM) for storing some deterministic test patterns. These stored test patterns are used to capture faults that cannot be detected by the test patterns generated by the on-chip TPG.

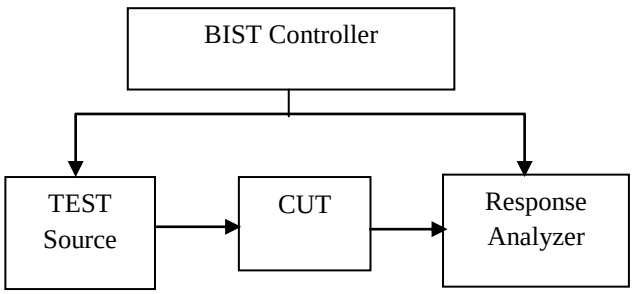


Fig 1: BIST Architecture

II. EXISTING MODEL

Test patterns are generated by using several linear shift registers for testing a Circuit under Test (CUT). On-chip test pattern generators can generate exhaustive or pseudo-random test patterns. Pseudo-random test patterns have many characteristics of random patterns, but are generated deterministically and hence are repeatable. A cellular automata or hardware based linear feedback shift register (LFSR) [2], [5] can be used to generate pseudo-random test patterns. An LFSR will be generally referred to as a TPG. An LFSR is a shift register with feedbacks from the last stage and other stages. The outputs of its flip-flops form the test pattern. Each state of the LFSR corresponds to one test pattern. The number of unique test patterns the LFSR can

generate depends on the number and location of the feedbacks as well as its initial value, which is known as the seed. An example of an LFSR is shown in Figure 3

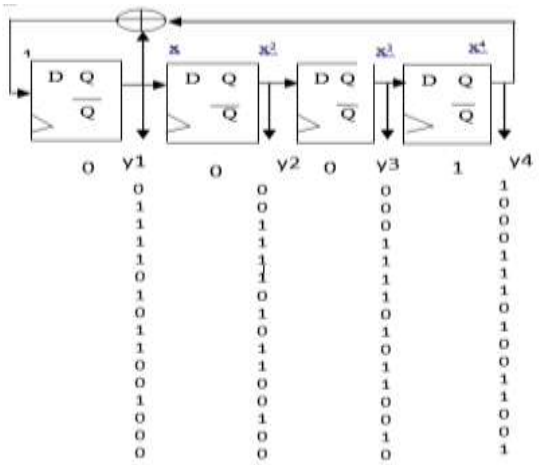


Fig 2: LFSR Example

It is initialized with the seed 0001. In the subsequent clock cycles, a series of test patterns are produced at the outputs of the flip-flops. This LFSR, which has n=4 flip-flops, produces a total of 15 (2n - 1) distinct patterns (except 0000) as shown in Figure.3. The feedback positions are usually described by a characteristic polynomial [2], [3]. In our example, feedbacks are made from the first (x) and the fourth (x4) positions, hence the characteristic polynomial of the LFSR is p(x) = 1 + x + x4. The choice of feedback positions (the choice of the polynomial) determines the length of the test sequences generated. Special polynomials known as primitive polynomials give maximal length sequences (2n-1). A polynomial p(x) = 1 + x + x4, which is used in our example, is primitive. It generates a sequence of 15 distinct test patterns before repetition. Therefore, when designing an LFSR a good choice of seed and polynomial is crucial for generating a good sequence of tests.

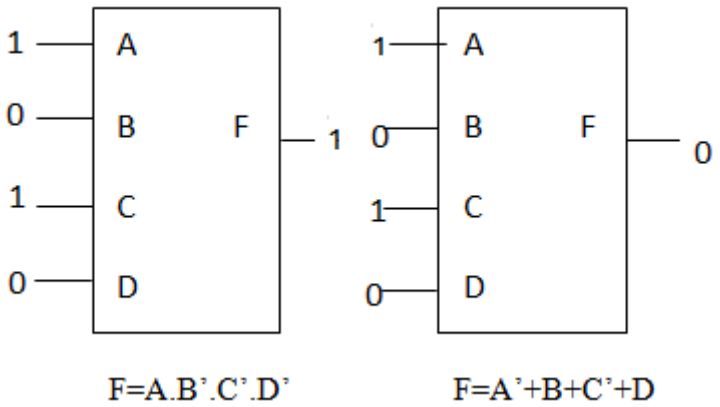
III. PROPOSED MODEL

Application-dependent testing of a SRAM-based FPGA interconnect is been proposed. The novelty of this comprehensive method is that it connects an activating input to multiple nets, thus generating a compact set of activating test vectors and requiring a reduced numbers of configurations. The faults covered in this technique include all possible stuck-at, open, and pair wise bridging faults. As in [1], [3], [7], detection is not based on physical information (such as layout) of the FPGA interconnect; in this latter case, the possible adjacencies could be found and the number of pair wise bridging faults could be reduced. However, to allow a fair comparison with existing works [1], [3], [6] logic simulation is therefore employed also in this paper; therefore, all possible stuck-at, open, and pair wise bridging faults are considered and detected by the proposed approach. So, for detecting the faults at the primary outputs, the induction fault detection method presented in [1] is adopted. The proposed method is heuristic in execution (the underlying problem has exponential complexity). This is lower than a previous comprehensive method [1] as well as by combining different methods (such as [3] and [8] using a fixed (constant) number of configurations). The proposed method has a computational algorithm execution with L is the number of LUTs in the design. Therefore, the proposed method differs from previous approaches for application-dependent testing [1], [3], [6] by utilizing the following novel features:

1. Differently from [1], [3] in which every net is driven by a single input, in the proposed approach test inputs simultaneously drive multiple nets; hence, application testing is very efficient in terms of requiring a lower number of configurations (thus saving time and cost). The proposed approach reduces redundancy in test configurations.
2. Input test assignment reflects a novel heuristic criterion (i.e., net sorting) that simulation has shown to be very effective for the ISCAS 89 benchmarks.
3. In the proposed method, the test generation algorithm exploits specific features (such as the occurrence of each net in connecting the LUTs) of the interconnect at a polynomial execution complexity to avoid and resolve so-called conflicts, thus resulting in a reduction in the number of test vectors (while preserving full coverage).

This paper is organized as follows: Section 2 presents a review of existing methods and the preliminaries of the proposed method; Section 3 outlines the basic principles of the proposed method, while Section 4 presents the configuration algorithm. Section 5 presents the logic simulation results and a comparative

discussion with previous methods found in the technical literature. Section 6 concludes this manuscript. The general form of a single-term function is a logic AND or OR function with possibly some inversions at the input(s) and/or the output [1]. Therefore, the output value of a single-term function is the same for all combinations of input vectors, except for one set of input vector; such vector is referred to as an activating input vector. Figs. 1a and 1b show examples of a single-term function for a max term and a min term, respectively.



Figs 3: (a) Max function. (b) Min function

In Fig.3 (a), an activating input vector (1010) is applied to the inputs of a LUT such that 1 is generated at the output for the max function F ¼ A: B0: C: D0. Among the nets connected to the inputs of the LUT, if anyone of them is faulty, then the output of the corresponding LUT is inverted (as outcome of that fault). Using induction, the fault is propagated to the primary output(s), where detection is accomplished. If the given input vectors are the stimulating input vectors, then all the faults that are sensitized are detected [1]. For sequential circuits, all Flip Flops in the design must be configured to be transparent; this is accomplished by presetting each FF to either 1 or 0 depending on the default value of the max term or min term for the LUT connected to its input. To test the nets, several sets of activating input vectors and their corresponding single-term functions are required. For example, in addition to the activating input vector (1010), the activating input vectors (0110), (0001), and the corresponding single-term functions (A'.B.C.D') (A'.B'.C'.D) detect all possible stuck-at, open, and pair wise bridging faults of the nets connected to the input of the LUT shown in Fig. 3(a).

The activating input vectors required to sensitize the faults in the nets are generated using the Walsh code [1].The Walsh code for an interc onnect with N nets is generated by its binary representation as a number. As an example, consider the circuit shown in Fig. 4 (9 LUTs and 30 nets). The corresponding Walsh code for the 30 nets is shown in Table 1. Each row of the Walsh code represents an activating input that is dedicated to one of the nets over five configurations. For example, net n1 is driven by the activating input k1 (0, 0, 0, 0, 1) over the five configurations C1 through C5. An activating input vector and the corresponding single-term function for a LUT for a configuration are derived from a set of activating inputs. As shown in Fig. 2, the nets n1, n2, n3, n4, and n5 that are connected to LUT L1, are driven by the activating inputs k1, k2, k3, k4, and k5, respectively, over five configurations. Table 1 shows the value of the activating inputs k1, k2, k3, k4, and k5 for the configuration C1; they are given by 1, 0, 1, 0, and 1, respectively. Thus, using the four activating inputs (k1, k2, k3, k4), an activating input vector (1010) is obtained, while the value of k5 determines the type of single-term function (min or max). Thus, for C1, L1 is configured to the single-term function. Table 2 shows the complete set of single-term functions generated for each of the LUTs over the five configurations using [1]. The following definitions are therefore introduced:

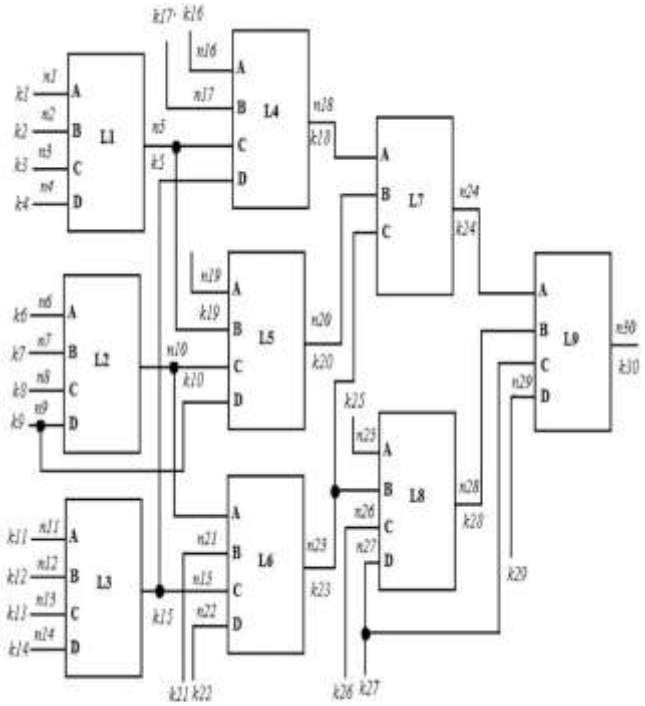


Fig 4: Activating input assignment

Walsh Code for 30 Nets:

Table 1 Walsh Code for 30 nets

Configurations					Activating inputs	Configurations					Activating inputs
C5	C4	C3	C2	C1		C5	C4	C3	C2	C1	
0	0	0	0	1	*k1*	1	0	0	0	0	*k16*
0	0	0	1	0	*k2*	1	0	0	0	1	*k17*
0	0	0	1	1	*k3*	1	0	0	1	0	*k18*
0	0	1	0	0	*k4*	1	0	0	1	1	*k19*
0	0	1	0	1	*k5*	1	0	1	0	0	*k20*
0	0	1	1	0	*k6*	1	0	1	0	1	*k21*
0	0	1	1	1	*k7*	1	0	1	1	0	*k22*
0	1	0	0	0	*k8*	1	0	1	1	1	*k23*
0	1	0	0	1	*k9*	1	1	0	0	0	*k24*
0	1	0	1	0	*k10*	1	1	0	0	1	*k25*
0	1	0	1	1	*k11*	1	1	0	1	0	*k26*
0	1	1	0	0	*k12*	1	1	0	1	1	*k27*
0	1	1	0	1	*k13*	1	1	1	0	0	*k28*
0	1	1	1	0	*k14*	1	1	1	0	1	*k29*
0	1	1	1	1	*k15*	1	1	1	1	0	*k30*

For example, consider LUT L1 shown in Fig. 2; this LUT has four inputs which are connected to the nets n1, n2, n3, and n4. The activating inputs k1(00001), k2(00010), k3(00011), and k4(00100) that drive nets n1, n2, n3, and n4, form five activating input vectors (1010), (0110), (0001), (0000), and (0000).

IV. CONFIGURATION GENERATION ALGORITHM

The process of assigning activating inputs and test vectors in the configurations for the nets starts by extracting the interconnect features of a specific circuit mapped to an FGPA (in this case, this is given by the Virtex 4) from the Native Generic Database (NGD) of the design. The NGD is converted to a text file and then the nets are sorted based on the number of connected LUTs. A greedy criterion for selecting the activating inputs and test vectors is used as follows: The first activating input is assigned to a net that is connected to the

largest number of LUTs, followed by assigning the second and third activating inputs to the other nets in a descending order. However, during each activating input assignment, it must be ensured that no two (or more) nets connected to a LUT are driven by the same activating input. In the case of two nets connected to the same LUT having the same activating input, one of the activating inputs is assigned to the next available activating input to resolve the conflict. The pseudocode for the above process as given in the induction method of Tahoori [3] is utilized for propagating the sensitized faults to the primary output, thus accomplishing observability in detection.

At completion of the algorithm, the activating inputs to the LUTs are found and hence, the corresponding test configurations can be determined. For example, the activating inputs for LUT L1 are k1, k2, k3, k4 and their Walsh code are 001, 010, 011, 100. Rearranging the Walsh code, grouping the first, second, and third bits of k1, k2, k3, k4 will generate three activating vectors: namely 1010, 0110, 0001. Also, by knowing the activating input assigned to the LUT output net, it is possible to determine the test configurations. In this example, the output of the LUT is assigned to k1 (001). Therefore, the first configuration is given by max (A.B'.C.D') while the remaining two configurations are min (A.B'.C'.D) and (A.B.C.D'). The test configurations (single-term functions) are determined in Step for all LUTs in the circuit. Table 2 shows the complete list of configurations generated for all LUTs in the circuit of Fig. 3.5, i.e., three for the proposed method versus five by using [5]. Just as in [5], the proposed method is comprehensive because it considers all faults (under the same assumed fault model) within a single algorithmic framework.

V. RESULTS & DISCUSSION

To avoid redundancy and reduce the number of configurations the so-called activating input vectors and test configurations using Walsh coding are utilized as per the following feature. In our method, an activating input is used to drive multiple nets, provided no two (or more) nets connected to a LUT are driven by the same activating input. From previous methods in which each of the nets is driven by a dedicated activating input and the corresponding single-term functions for the circuit are shown in Table 2. In this case, only three configurations are required; thus, by assigning an activating input to multiple nets (such as k1 to drive n1, n5, n6, n11, n21, n23, n24, and n30), a substantial reduction in the number of configurations can be accomplished. As a simpler example to clarify the above definitions and process for the proposed method, consider the circuit shown in the Fig. 3 with three LUTs and 13 nets (n1-n13).

Table 2 Single-Term Functions for LUTs

LUTs	Configurations		
	C1	C2	C3
L1	A.B'.C.D'	A+B'+C'+D	A+B+C+D'
L2	A'+B+C'+D	A+B'+C'+D	A'.B'.C.D
L3	A.B'.C.D'	A+B'+C'+D	A+B+C+D'

Table 3 Configurations Generated for the Circuit Fig 4

Activating Input				Activating Input Vector				Configs	List of Faults Covered
				A	B	C	D		
K1	0	0	1	1	1	1	0	C1	A/0, B/1, C/0, D/1, AB_{AND}, AD_{AND}, BC_{AND}, $CD_{AND,}$ AB_{OR}, AD_{OR}, BC_{OR}, CD_{OR}
K2	0	1	0	0	1	1	0	C2	A/1, B/0, C/0, D/1, AB_{AND}, AC_{AND}, BD_{AND}, $CD_{AND,}$ AB_{OR}, AC_{OR}, BD_{OR}, CD_{OR}
K3	0	1	1	0	0	0	1	C3	A/1, B/1, C/1, D/0, AD_{AND}, BD_{AND},

									CD_{AND}, AB_{OR}, BD_{OR}, CD_{OR}
K4	1	0	0						

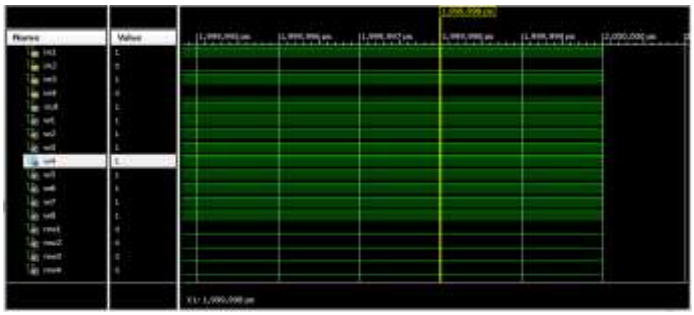


Fig 5: Simulation Result for Configuration 1

Fig 6: Simulation Result for Configuration 2

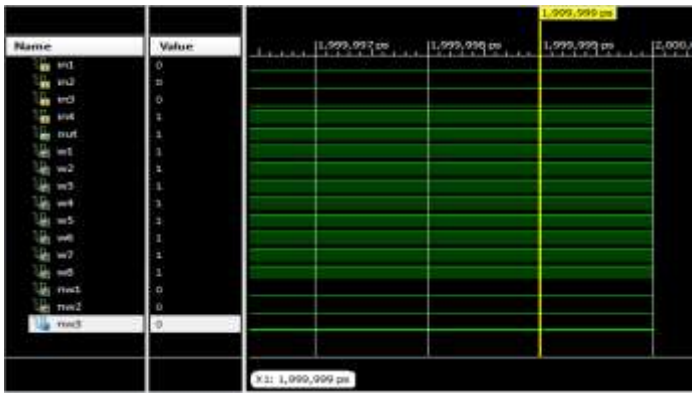


Fig 7: Simulation Result for Configuration 3

Table 4 Detected Faults for the Circuit in Fig. 4

Stuck-at-Fault				Pair-wise Bridging Fault			
Stuck-at-1	Status	Stuck-at-0	Status	Wire-OR	Status	Wire-AND	Status
A/1	✓	A/0	✓	AB	✓	AB	✓
B/1	✓	B/0	✓	AC	✓	AC	✓
C/1	✓	C/0	✓	AD	✓	AD	✓
D/1	✓	D/0	✓	BC	✓	BC	✓
✓ - Fault is covered				BD	✓	BD	✓
				CD	✓	CD	✓

VI. CONCLUSION

Thus a new algorithmic method for application-dependent testing of a SRAM-based FPGA interconnect is been proposed. The proposed method relies on generating and utilizing so-called activating inputs connected through multiple nets with Walsh coding. This algorithmic-based method detects all stuck-at, open, and pair wise bridging faults in the interconnect resources of an FPGA. Analysis and simulation have shown that the heuristic criterion used in the proposed method results in an efficient generation of test configurations. For further works, the same three configurations can be implemented even if the numbers of LUTs are increased and thus the testing time can be decreased

REFERENCES

[1]. M.B. Tahoori, "Application-Dependent Testing of FPGAs," IEEE Trans. Very Large Scale Integration Systems, vol. 14, no. 9, pp. 1024- 1033, Sept. 2006.
[2]. M.B. Tahoori, "Application-Dependent Testing of FPGA Interconnects," Proc. 18th IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems, pp. 409-416, Nov. 2003.
[3]. M.B. Tahoori, E.J. McCluskey, M. Renovell, and P. Faure, "A Multi-Configuration Strategy for an Application Dependent Testing of FPGAs," Proc. 22nd IEEE VLSI Test Symp., pp. 154- 159, Apr. 2004.
[4]. M.B. Tahoori, "Application-Dependent Diagnosis of FPGAs," Proc. IEEE Int'l Test Conf., pp. 645-654, Oct. 2004.
[5]. A. Doumar and H. Ito, "Testing the Logic Cells and Interconnect Resources for FPGAs," Proc. Eighth Asian Test Symp., pp. 369-374, Nov. 1999.
[6]. Y. Yu, J. Xu, W.K. Huang, and F. Lombardi, "A Diagnosis Method for Interconnects in SRAM Based FPGAs," Proc. Seventh Asian Test Symp., pp. 278-282, Dec. 1998.
[7]. W.K. Huang, X.T. Chen, and F. Lombardi, "On the Diagnosis of Programmable Interconnect System: Theory and Application," Proc. 14th VLSl Test Symp., pp. 204-209, Apr. 1996.
[8]. F. Lombardi, D. Ashen, X. Chen, and W.K. Huang, "Diagnosing Programmable Interconnect System for FPGAs," Proc. Fourth ACM Int'l Symp. Field-Programmable Gate Arrays, pp. 100-106, 1996.