International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

VHDL Implementation of Built in Generation of Functional Broadside Tests

N. A. Raju D.[1], G. Sita Annapurna[2]

[1]M. Tech (VLSI & SD) Student, Department of Electronics and Communication Engineering, Sri Vasavi Institute of Engineering and Technology, Nandamuru, Machilipatnam, JNTUK, Kakinada, A.P., India

[2]Assistant Professor, Department of Electronics and Communication Engineering, Sri Vasavi Institute of Engineering and Technology, Nandamuru, Machilipatnam, JNTUK, Kakinada, A.P., India

Abstract*: Functional broadside tests are two-pattern scan-based tests that avoid over testing by ensuring that a circuit traverses only reachable states during the functional clock cycles of a test. In this paper, we test the s27 sequential circuit by using Built in Self Test. The hardware was based on the application of primary input sequences initial from a well-known reachable state. Random primary input sequences were changed to avoid repeated synchronization and thus yield varied sets of reachable states by implementing a decoder in between circuit and LFSR. This paper shows the on chip test Generation for a bench mark circuit using simple fixed hardware design with small no of parameters altered in the design for the generation of number of patterns. If the patterns of the input test vector results a fault simulation then circuit under test is going to fail.*

Keywords: Built-in test generation, functional broadside tests, Transition faults, LFSR, Reachable states.

1. Introduction

Very Large Scale Integration (VLSI) has made a dramatic impact on the growth of integrated circuit technology. It has not only reduced the size and the cost but also increased the complexity of the circuits. The positive improvements have resulted in significant performance/cost advantages in VLSI systems. There are, however, potential problems which may retard the effective use and growth of future VLSI technology. Among these is the problem of circuit testing, which becomes increasingly difficult as the scale of integration grows. Because of the high device counts and limited input/output access that characterize VLSI circuits, conventional testing approaches are often ineffective and insufficient for VLSI circuits.

Built-in self-test (BIST) is a commonly used design technique that allows a circuit to test itself. BIST has gained popularity as an effective solution over circuit test cost; test quality and test reuse problems. In this paper we are presenting an implementation of a tester using VHDL. Test time is a significant component of IC cost. It needs to be minimized and yet has to have maximum coverage to ensure zero-defect. There is a need for design for testability techniques.

For any testing methodology, the following factors should be considered- high and easily verifiable fault coverage, minimum test pattern generation, minimum performance degradation, at-speed testing, short testing time and reasonable hardware overhead. With increasing integration density, the amount of manufacture faults is increasing. Thus we have to test the chip. With increasing complexity of the design, it becomes impossible to test the chip externally. Thus, we have to use BIST Built-In Self-Test (BIST) provides a feasible solution to the above demands. Another advantage of this methodology is that the test patterns are not applied by external Automatic Test Equipments (ATEs) but generated by in built testing circuit. It saves the memory requirement during test. BIST is a design technique in which parts of a circuit are used to test the circuit itself.

Test generation procedures for functional and pseudo-functional scan-based tests were described in [4] and [6]–[13]. The procedures generate test sets offline for application from an external tester. Functional scan-based tests use only reachable states as scan-in states. Pseudo-functional scan-based tests use functional constraints to avoid unreachable states that are captured by the constraints.

This work considers the on-chip (or built-in) generation of functional broadside tests. On-chip test generation reduces the test data volume and facilitates at-speed test application. The on-chip test generation method from [16] applies pseudo-functional test generation based on LFSR. The on-chip test generation process described in this work guarantees that only reachable states will be used. However, the tests that are needed for achieving this higher fault coverage are also ones that can cause over testing.

If a primary input sequence A is applied in functional mode starting from a reachable state, all the states traversed under A are reachable states [19][20]. Any one of these states can be used as the initial state for the application of a functional broadside test. By generating an on-chip and ensuring that it takes the circuit through a varied set of reachable states, the on-chip test generation process is able to achieve high transition fault coverage using functional broadside tests based on A. Previous works are mentioned below.

1) Running the test at a slower frequency than in normal mode. This technique of reducing power consumption, while easy to implement, significantly increases the test application time.

2) Reduce the power consumption in scan-based built-in self-tests (BISTs) is by using scan chain ordering techniques. These techniques aim to reduce the average-power consumption when scanning in test vectors and scanning out captured responses. Although these algorithms aim to reduce average-power consumption, they can reduce the peak power that may occur in the CUT during the scanning cycles, but not the capture power that may result during the test cycle (i.e., between launch and capture)[1].
3) Modifying the test vectors generated by the LFSR to get test vectors with a low number of transitions. The main drawback of these techniques is, it results in lower fault coverage and higher test application time.

The hardware used in this paper for generating the primary input sequence A consists of a pseudo- functional scan based test with linear-feedback shift-register (LFSR) as a random source [17], and of a small number of gates (at most six gates are needed for every one of the benchmark circuits considered). The gates are used for modifying the random sequence in order to avoid cases where the sequence takes the circuit into the same or similar reachable states repeatedly. This is referred to as repeated synchronization [13]. In addition, the on-chip test generation hardware consists of a single gate that is used for determining which tests based on will be applied to the circuit. The result is a simple and fixed hardware structure, which is tailored to a given circuit only through the following parameters.

1) The number of LFSR bits.
2) The length of the primary input sequence.
3) The specific gates used for modifying the LFSR sequence into the sequence.
4) The specific gate used for selecting the functional broadside tests that will be applied to the circuit based on.
5) Seeds for the LFSR in order to generate several primary input sequences and several subsets of tests.

2. Block Diagram of BIST Structure

Fig.1 shows a simple BIST block diagram which uses a linear feedback shift register (LFSR) to generate the test vectors for circuit under test. LFSR is used as a pseudorandom sequence generator.

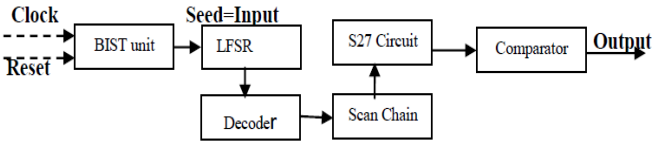

Figure 1: Block diagram of BIST Structure

In this paper we are testing the functional operation of reference circuits. For this we need to check Maximum possible input conditional that may activate the each and every element in the IC. By this we can check each and every transistor in the IC and will know is that IC working perfectly or not. We use 12 bit LFSR to generate 12 bit random patterns. By using some basic digital gates we convert these 12 bit patterns into 4 bit patterns. These patterns are applied to the s27 benchmark circuit. Fig.4 is reference circuit in our project. It needs four inputs. For 4 bit inputs there are 16 different combinations in digital. But we don't need those 16 combinations we require 4 patterns to check the complete IC and its functional operation. We generate that particular input patters which are required to check IC. We generate these test patterns by seed of LFSR.

The logic that produces the primary input sequence is designed in this paper to reduce the dependencies between the values assigned to the primary inputs, considering the following sources of dependency. In [19], for a circuit with n primary inputs and a parameter mod, the LFSR used for producing A has n+mod bits. The n left-most bits are used for driving the primary inputs of the circuit, and the mod right-most bits are used for modifying the random sequence in order to avoid repeated synchronization. With this structure, all the primary input values are modified using the same function of the mod right-most bits of the LFSR. Thus, they are always modified together and to the same values. In addition, some primary inputs receive shifted values of the primary inputs immediately preceding them. The structure used in this paper reduces these dependencies between primary input values by using a (d.n)-bit LFSR for a circuit with primary inputs, where is a parameter such that d>mod. Every d consecutive bits of the LFSR are used for producing the value of a different primary input. At most *mod* of the bits dedicated to a primary input are actually used for producing values for the input, including the modification of the input values in order to avoid repeated synchronization. Since the modification is done using different bits for every primary input, the dependencies between primary input values are reduced.

2.1 Linear Feedback Shift Registers(LFSRs)

Linear feedback shift register (LSFR) is a shift register whose input bit is a linear function of its previous state. The only linear function of single bits is XOR, thus it is a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value flops. The initial value of the LFSR is called the seed, and the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. The seed is used to generate a test pattern and their corresponding test cube. Reseeding is a very powerful method for reducing test data. Most of the test data reduction is mainly concentrating on LFSR reseeding. The basic idea in LFSR reseeding is to generate deterministic test cubes by expanding seeds. A seed is an initial state of the LFSR that is expanded by running the LFSR in autonomous mode. An LFSR generates periodic sequence must start in a non-zero state, the maximum length of an LFSR sequence is 2^n -1 does not generate all 0s pattern.

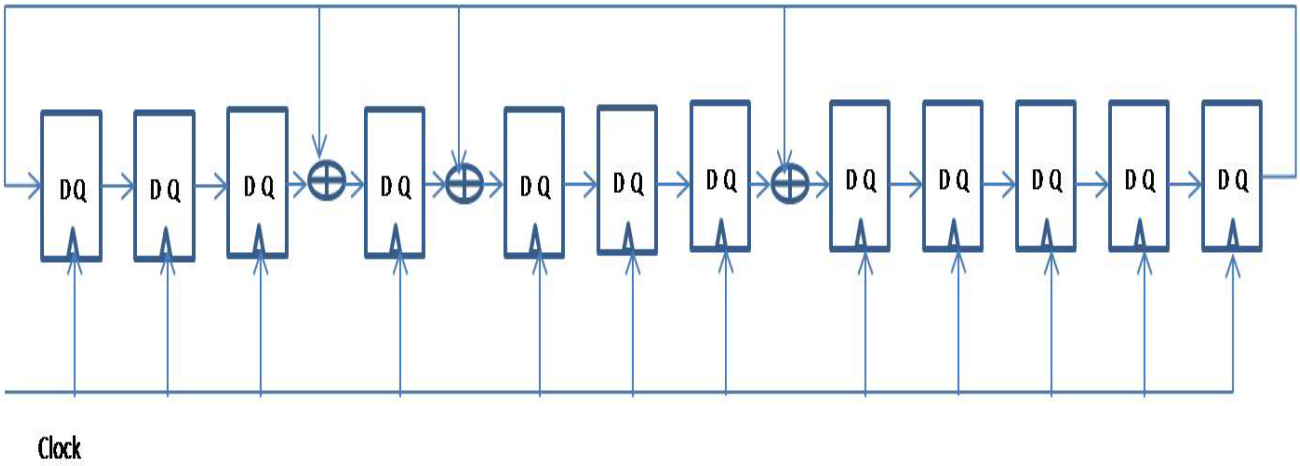

Figure 2: 12-bit LFSR Circuit

2.2 Primary Input Sequence

Fig. 3 shows the hardware used for s27 with parameters d=3 and mod=2.Primary input cube I0I1I2I3=0xxx applied in present state y0y1y2=xxx results in next state Y0Y1Y2=0xx synchronizing state variable y0. In addition, Primary input cube I0I1I2I3=xx1x applied in present state y0y1y2=xxx results in next state Y0Y1Y2=xx0 synchronizing state variable y2. For the circuit s27, number of bits in primary input=4 and combined single cube c=1x0x which is useful for avoiding repeated synchronization.

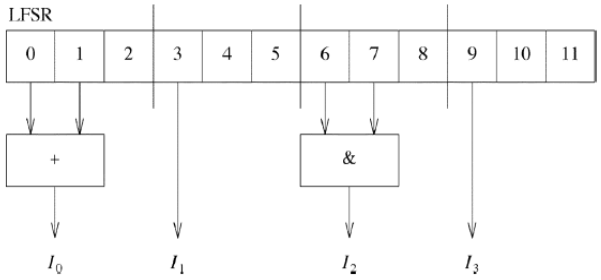

Figure 3: Generation of Primary input Sequence A.

Bits 0, 1 and 2 of the LFSR are used for producing the values of I_0. Since c(0)=1, an OR gate is used for increasing the probability that I_0 will be assigned the value 1. The OR gate is driven by bits 0 and 1 of the LFSR. Bit 2 of the LFSR reduces the dependencies between the values of I_0 and the values of I_1. Bits 3, 4, and 5 of the LFSR are used for producing the values of I_1. For I_1, c(1)=x . Therefore, I_1 is driven directly by bit 3 of the LFSR. Bits 4 and 5 reduce the dependencies between the values of I_1 and the values of I_2. Bits 6, 7, and 8 are used for producing the values of I_2. Since c(2), an AND gate is used for increasing the probability that I_2 will be assigned the value 0. The AND gate is driven by bits 6 and 7. Bit 8 of the LFSR reduces the dependencies between the values of I2 and those of I3. Finally, bits 9, 10, and 11 are used for producing the values of I3. For I3, c(3)=x. Therefore, I3 is driven directly by bit 9 of the LFSR. In general, if there are N primary inputs Ij with c(j) not equal to x , the implementation illustrated by Fig. 3 requires a (d.n)-bit LFSR, and N AND or OR gates with mod inputs. LFSR sequence for s27 The primary input sequence shown in Table I was produced by the structure shown in Fig. 2 using the 12-bit primitive LFSR $X^{12}+X^7+X^4+X^3+1$ with seed 101 011 100 100. The states of the LFSR are shown in Table 1.

Table 1: LFSR Sequence for s27

u	lfsr(u)
0	101 011 100 100
1	010 101 110 010
2	001 010 111 001
3	100 011 001 100
4	010 001 100 110
5	001 000 110 011
6	100 010 001 001
7	110 111 010 100
8	011 011 101 010
9	001 101 110 101
10	100 000 101 010
11	010 000 010 101
12	101 110 011 010
13	010 111 001 101
14	101 101 110 110
15	010 110 111 011

2.3 Sequential Benchmark Circuit s27

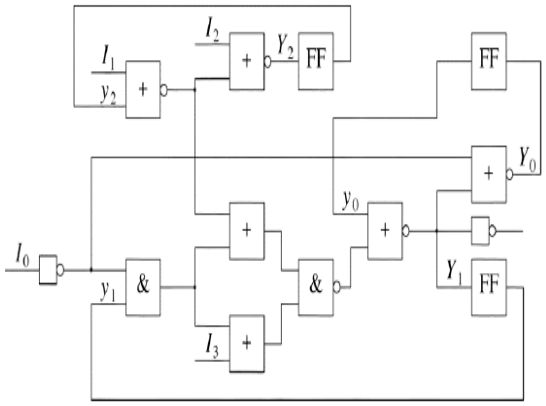

Figure 4: s27 circuit.

We are used s27 bench mark circuit which is a standard sequential circuit as a testing circuit. Applying test vectors as input to the s27 bench mark sequential circuit. I_0, I_1, I_2, I_3 are the input of this circuit.

The initial state of the circuit is denoted by s_r.The discussion also assumes that functional operation consists of the application of primary input sequences starting from state sr. With s_r as the initial state for functional operation, s_r is a reachable state. In addition, the set of reachable states consists of every state s_i such that there exists a primary input sequence that takes the circuit from s_r to s_i. Since s_i can be entered during functional operation starting from s_r, s_i is a reachable state. It is possible to obtain reachable states on- chip by placing the circuit in state and applying a primary input sequence A=a(0),a(1)....a(L-1) of length L in functional mode. The circuit can be brought into state s_r by using a scan-in operation, or by using its initializing sequence. Let s(u) be the state that the circuit reaches at time unit u under A , for 0<=u<=L . We have that s(0)=s_r . In addition, s(u) is a reachable state for 0<=u<=L. Therefore, every state s(u) can be used as the initial state for a functional broadside test (s(u),a1,a2), where s(u) plays the role of a scan-in state. As in a broadside test, a1 and a2 are primary input vectors that are applied in two consecutive functional clock cycles starting from s(u) using a slow and a fast clock, respectively. In addition to producing reachable states, the primary input Sequence A can also be used as a source for the primary input vectors of functional broadside tests.

We consider ISCAS-89 benchmark s27 with initial state sr=000 is shown in Fig. 4. A primary input sequence for the circuit is shown in Table 2. For every time unit u, Table 2 shows the state s(u) and the primary input vector a(u) . Table 2 yields the functional broadside tests t(0)=(000,1001,1110),t(1)=(010,1110,0010).....t(14)=(101,1111,1110).The proposed on-chip generation method of functional broadside tests is based on placing the circuit in the initial state , applying a primary input sequence A , and using several of the functional broadside tests that can be extracted from in order to detect target faults. Next, we discuss how the application A is affected by the need to observe fault effects created by a test t(u)=(s(u),a(u),a(u+1)).

A fault can be detected in one of the following two ways.1) Based on the primary output vector z(u+1) obtained in response to a(u+1), if this vector is different from the expected fault free primary output vector.2) Based on the final state s(u+2)of the test, if this state is different from the expected fault free state. In the context of built-in self-test, z(u+1)and s(u+2)need to be captured by an output response compactor. In the case of s(u+2) , the state needs to be shifted into the output response compactor over a number of clock cycles equal to the length of the longest scan chain. For the s27 circuit, n=4, d=3,mod=2 .Here 2 LFSR bits are for modifying the value of a primary input. The probability of changing a primary input value is ¾. The parameters of the on chip test generation hardware are summarized in Table 3 for ease of reference.

The circuit then needs to be brought back to state s(u+2) in order to continue the test A. application process under A. Bringing the circuit back to state s(u+2)can be done by using circular shift of s(u+2).

Table 2: Primary input sequence for s27.

u	s(u)	a(u)	Op
0	000	1001	0
1	010	1110	1
2	100	0010	1
3	000	1001	0
4	010	1001	0
5	010	0010	0
6	010	1000	1
7	100	1101	1
8	101	1000	1
9	101	0111	1
10	000	1000	1
11	100	1001	1
12	100	1100	1
13	101	1101	1
14	101	1111	1
15	100	1110	1

Table 3: Parameters.

Parameter	Meaning
L	Length of primary input sequences
D	Number of LFSR bits per primary input
Mod	Number of LFSR bits modifying the value of a primary input. the probability of changing a primary input value $1-1/2^{mod}$
Sel	Tests starting at time units that are divisible by sel are applied to the circuit. Sel=2^m

As s(u+2) is scanned out, it can also be scanned in. If s(u+2)is faulty, the output response compactor will capture the fault effect, and observation of the final signature will indicate that a fault is present. If s(u+2) is fault free, the remaining tests based on A will be applied as required. It should be noted that the tests starting in two consecutive time units, u and u+1, are overlapping in the following sense. Application of t(u) takes the circuit through states s(u),s(u+1) and s(u+2). Application of t(u+1) takes the circuit through states , s(u+1),s(u+2)and .s(u+3). The application of both t(u) and t(u+1) requires special hardware to bring the circuit back to state s(u+1)after t(u) is applied. To avoid the need for this hardware, the proposed test generation hardware applies subsets of non-overlapping tests of the form{t(u0),t(u1),....,t(uk-1)},where ui+1<ui+1 for 0<=i<k-1.

2.4 Comparator

The comparator will produce logic 1 in the case of any mismatch between the expected and actual output responses. Comparator is used for detecting mismatches in the fault-free and faulty circuits .Comparator is used to compare the two outputs coming from fault circuit and fault free circuit. If both the outputs are same there is no fault and fault signal is zero, otherwise there is a fault and fault signal is one.

3. Experimental Results

3.1 Result Analysis

Here test vectors generated by the LFSR with decoding circuit are given to the input of circuit under test and fault is detected by comparing the output of fault free circuit and faulty circuit by using comparator. The figures 6-10 are the simulation results done in the Xilinx ISE 12.1.The figure 9 shows no fault case in the s27 circuit and figure 10 shows there is a fault in the circuit. The clock and reset signals are inputs to the top level entity. If the comparator output is zero then no fault in the circuit else there is a fault in the circuit. Corresponding output waveforms are given below.

3.1 RTL Schematic

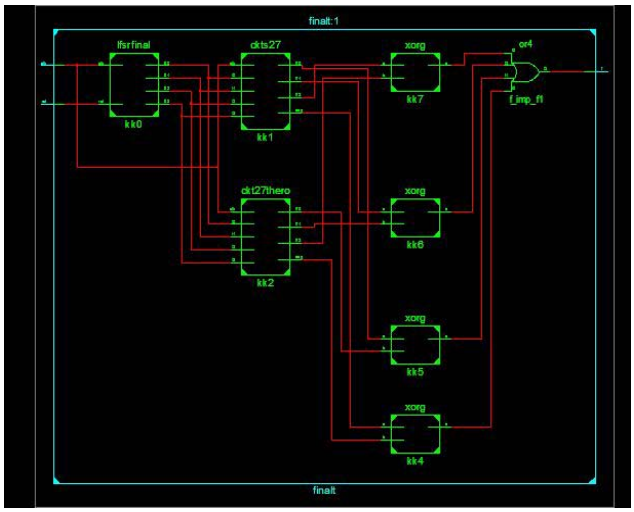

Figure 5: RTL Schematic

3.2 Output waveforms of LFSR

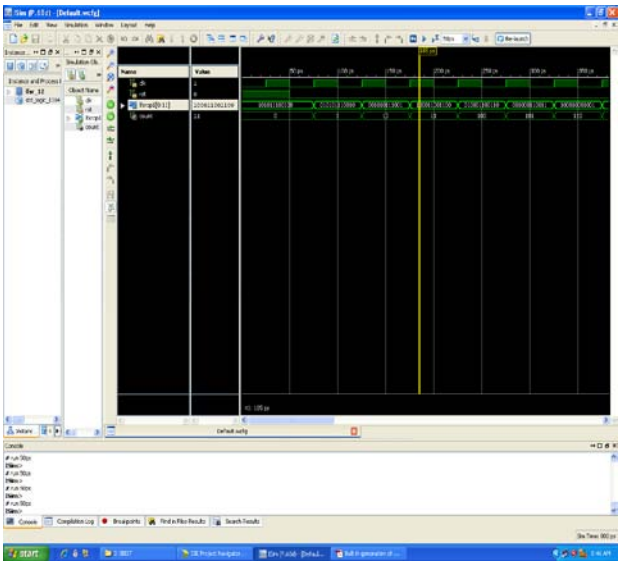

Figure 6: Output pattern at LFSR output

3.3 Generation of primary input sequence

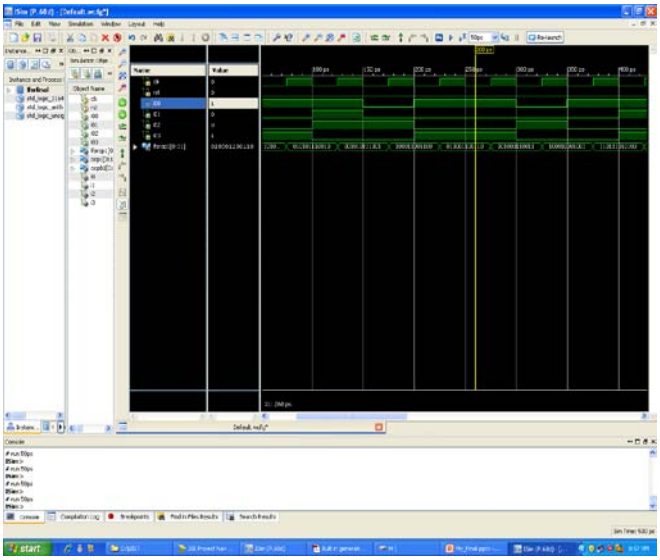

Figure 7: Generation of primary input sequence from the LFSR output

3.4 Output waveforms of s27 circuit

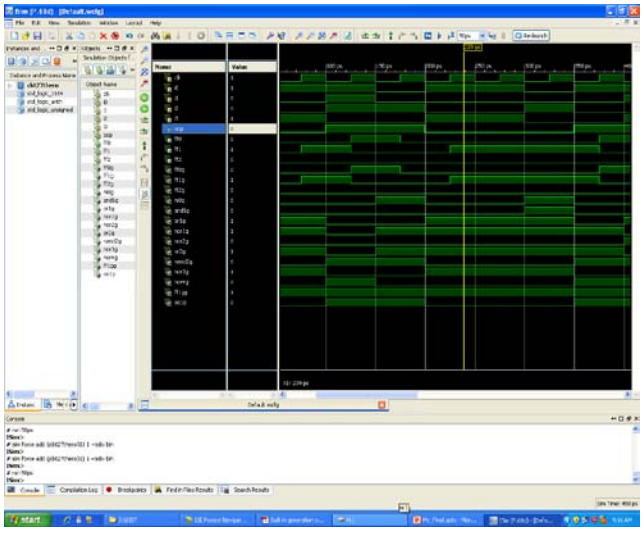

Figure 8: Output waveforms of s27 circuit

3.5 Waveforms represent no fault in the circuit

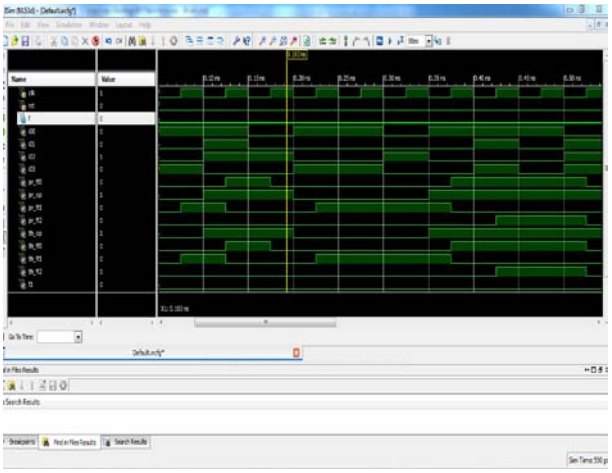

Figure 9: Waveforms represent no fault in the circuit

3.6 Waveforms represent fault in the circuit

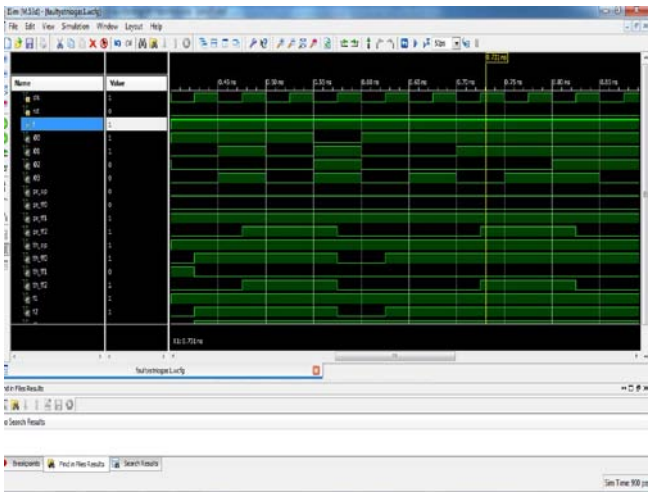

Figure 10: Waveforms represent fault in the circuit

4. Conclusion

The presence of delay-inducing defects is causing increasing concern in the semiconductor industry today. To test for such delay-inducing defects, on chip testing techniques are being implemented. On-chip test generation has the advantage it reduces test data volume, facilitates at-speed test application and achieves high fault coverage with low power estimated. The hardware used in this paper for generating the primary input sequence A consists of a linear-feedback shift-register (LFSR) as a random source and of a small number of gates to focus on reducing test pattern by avoiding repeated synchronization. The design is coded using VHDL language. The design is synthesized and simulated on Xilinx ISE 12.1 software.

5. Acknowledgement

I sincerely thank my guide G. Sita Annapurna, Assistant Professor in ECE Department of Sri Vasavi Institute of Engineering and Technology for her kind advice, support, encouragement as well as guidance for the preparation of research manuscript. I'm really grateful to acknowledge the support provided by my Mom & Dad.

References

[1] J. Rearick, "Too much delay fault coverage is a bad thing," in Proc. Int. Test Conf., 2001, pp. 624–633.
[2] J. Saxena, K. M. Butler, V. B. Jayaram, S. Kundu, N. V. Arvind, P.Sreeprakash, and M. Hachinger, "A case study of IR-drop in structured at-speed testing," in Proc. Int. Test Conf., 2003, pp. 1098–1104.
[3] S. Sde-Paz and E. Salomon, "Frequency and power correlation between at-speed scan and functional tests," in Proc. Int. Test Conf., 2008, pp. 1–9, Paper 13.3.
[4] I. Pomeranz and S. M. Reddy, "Generation of functional broadside testsfor transition faults," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 10, pp. 2207–2218, Oct. 2006.
[5] J. Savir and S. Patil, "Broad-side delay test," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 13, no. 8, pp. 1057–1064, Aug.1994.
[6] I. Pomeranz, "On the generation of scan-based test sets with reachable states for testing under functional operation conditions," in Proc. Design Autom. Conf., 2004, pp. 928–933.
[7] Y.-C. Lin, F. Lu, K. Yang, and K.-T. Cheng, "Constraint extraction for pseudo-functional scan-based delay testing," in Proc. Asia South Pacific Design Autom. Conf., 2005, pp. 166–171.
[8] Z. Zhang, S.M. Reddy, and I. Pomeranz, "On generating pseudo-functional delay fault tests for scan designs," in Proc. Int. Symp. Defect Fault Toler. VLSI Syst., 2005, pp. 398–405.
[9] I. Polian and F. Fujiwara, "Functional constraints vs. test compression in scan-based delay testing," in Proc. Design, Autom. Test Euro. Conf., 2006, pp. 1–6.
[10] M. Syal et al., "A study of implication based pseudo functional testing,"in Proc. Int. Test Conf., 2006, pp. 1–10.
[11] A. Jas, Y.-S. Chan, and Y.-S. Chang, "An approach to minimizing functional constraints," in Proc. Defect Fault Toler. VLSI Syst., 2006, pp.215–226.
[12] H. Lee, I. Pomeranz, and S. M. Reddy, "On complete functional broadside tests for transition faults," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., pp. 583–587, 2008.
[13] I. Pomeranz and S. M. Reddy, "On reset based functional broadside tests," in Proc. Design Autom. Test Euro. Conf., 2010, pp. 1438–1443.
[14] H. Lee, I. Pomeranz, and S.M. Reddy, "Scan BIST targeting transition faults using a Markov source," in Proc. Int. Symp. Quality Electron. Design, 2004, pp. 497–502.
[15] V. Gherman, H.-J. Wunderlich, J. Schloeffel, and M. Garbers, "Deterministic logic BIST for transition fault testing," in Proc. Euro. Test Symp., 2006, pp. 123–130.
[16] Y.-C. Lin, F. Lu, and K.-T. Cheng, "Pseudofunctional testing," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., pp. 1535–1546, 2006.
[17] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. Piscataway, NJ: IEEE Press, 1995.
[18] I. Pomeranz and S. M. Reddy, "Primary input vectors to avoid in random test sequences for synchronous sequential circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., pp. 193–197, 2008.
[19] I. Pomeranz, "Built-in generation of functional broadside tests," presented at the Design Autom. Test Euro. Conf., Grenoble, France, 2011.
[20] Built-In Generation Of Functional Broadside Tests Using A Fixed Hardware Structure Irith Pomeranz, Fellow, IEEE, Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 21, No. 1, January 2013.