IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

Low Transition Test Pattern Generation for Path Delay and Stuck-at-Faults

Dr.S.Kavitha[1], K.Seeshna Balakrishnan[2]

Professor, Department of ECE, Hindusthan Institute of Technology, Coimbatore, Tamilnadu, India[1]

PG Scholar, Hindusthan Institute of Technology, Coimbatore, Tamilnadu, India[2]

Abstract: A low transition test pattern generation for path delay and stuck-at-faults is proposed. Main challenges in generating compressed tests are reduced test data volume along with low transition pattern sequence. The proposed work uses BS-LFSR (Bit Swapping-Linear Feedback Shift Register) for generating test pattern. The basic approach that the paper uses modifies initially random seeds for the BS-LFSR into seeds that produce tests for detecting target faults. This approach can find seeds even if the available tests cannot be compressed into seeds. In addition, the procedure described in the proposed method also selects detectable path delay and stuck-at faults to address the presence of undetectable path delay and stuck-at faults in the set of target faults. The Bit-Swapping LFSR (BS-LFSR), is composed of an LFSR and a 2 × 1 Multiplexer and used to generate test patterns which reduces the number of transitions that occur at the scan-chain input during scan shift operation by 50% when compared to those patterns produced by a conventional LFSR. Hence, it reduces the overall switching activity in the circuit under test during test applications. Further, undetectable fault are covered using observation point insertion for improving the fault coverage. Experimental results for fault coverage are analyzed for path delay and stuck-at faults in benchmark circuits.

Keywords: BS-LFSR, Test Data Compression, Test Generation, Low-Power Test, Test Point Insertion

I. INTRODUCTION

In recent years, the design for low power has become one of the greatest challenges in high-performance Very Large Scale Integration (VLSI) design. As a consequence, many techniques have been introduced to minimize the power consumption of new VLSI systems. However, most of these methods focus on the power consumption during normal mode operation, while test mode operation has not normally been a predominant concern. However, it has been found that the power consumed during test mode operation is often much higher than during normal mode operation [1]. This is because most of the consumed power results from the switching activity in the nodes of the Circuit Under Test (CUT), which is much higher during test mode than during normal mode operation [1]–[3]. There are techniques that are used for generating test patterns which can efficiently detect all single faults in a circuit for industrial size designs. However, no practical technique for detection of simultaneously occurring multiple faults exists since the early researches addressing the importance of the problem, particularly for large designs. Authors in existing work present a method that is capable of test generation for all multiple faults in circuits of up to hundreds of gates size. The authors proposed methods that can generate tests for multiple faults. However, the method does not guarantee to cover all the faults. Moreover, they cannot handle largest ISCAS benchmarks.

Fault models are used for capturing the behavior of defects, and providing targets for fault simulation and test generation procedures that address these defects. Path delay faults model the situation where small extra delays cause a circuit to fail when they are accumulated along a path [4]. The number of paths in a circuit can be very large. Therefore, the number of path delay faults that can be defined for a circuit is very large. The procedures described in [5]-[6] address this issue by selecting the most important path delay faults as targets for fault simulation and test generation. Typically, these are the path delay faults that are associated with the longest paths. However, many of the path delay faults, especially the ones that are associated with the longest paths, are typically undetectable. Path selection procedures need to take this situation into consideration, and avoid selecting undetectable path delay faults.

The design of low-transition Test-Pattern Generators (TPGs) is one of the most common and efficient techniques for low-power tests [7]–[8]. These algorithms modify the test vectors generated by the LFSR to get test vectors with a low number of transitions. The main drawback of these algorithms is that they aim only to reduce the average-power consumption while loading a new test vector, and they ignore the power consumption that results while scanning out the captured response or during the test cycle.

The paper is organized as follows. Section II describes proposed low transition BS-LFSR. Section III describes the procedure to generate test patterns for target faults for path delay and stuck-at-fault. Section IV explains the test point insertion concept. Section V evaluates the results and Section VI concludes the paper.

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

II. THE PROPOSED LOW TRANSITION BS-LFSR

LFSR

The sequence of 1's and 0's that is followed by1 bit position of a maximal length LFSR commonly referred as m-sequence , each bit within the LFSR follow the m-sequence with the one time step delay . The m-sequence generated by a LFSR of length m has a periodicity of2^n-1 bits. The beginning of each sequence generator r is marked by a transition between 0's and 1's. Therefore the total number of transitions for each stages of LFSR is 2^n-1 in LFSR consider a maximal length –stage for internal LFSR .n value should be greater than 2.
The proposed LFSR is a combination of LFSR and 2*1 multiplexer. Swapped output is obtained from the final value of BS LFSR .we see how the swapped output is obtained is explained bellow In this we choose one of the cells and swap its value with its adjacent cell , if the current value of 3rd cell in the LFSR is 0 and leave the cells and swapped if the 3rd cell has a value I 1.the value of third cell is described as selection line value The selection line is linked to one of the swapped cells through an xor gate in this configuration a single cell can save 50% transition that where originally produced by an LFSR cell.

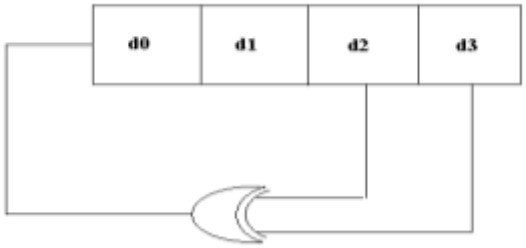

Fig.1 Conventional LFSR

Fig.1 shows of the design of conventional LFSR. A Linear Feedback Shift Register (LFSR) is a shift register whose input bit is a linear function of its previous state. The only linear functions of single bits are xor and inverse-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. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the sequence of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences.

- Ones and zeroes occur in 'runs'. The output stream 0110100, for example consists of five runs of lengths 1, 2,1,1,2, in order. In one period of a maximal LFSR, 2n − 1 runs occur (for example, a six bit LFSR will have 32 runs). Exactly 1 / 2 of these runs will be one bit long, 1 / 4 will be two bits long, up to a single run of zeroes n − 1 bits long, and a single run of ones n bits long. This same property is statistically expected in a truly random sequence.
- LFSR output streams are deterministic. If we know the present state, we can predict the next state. This is not possible with truly random events such as nuclear decay.
- The output stream is reversible; an LFSR with mirrored tap sequence will cycle through the states in reverse order.

Bit-Swapping LFSR

The Bit-Swapping LFSR (BS-LFSR), is composed of an LFSR and a 2 × 1multiplexer as shown in Fig.2. When used to generate test patterns for scan-based built-in self-tests, it reduces the number of transitions. The proposed BS-LFSR generates the same number of 1s and 0s at the output of multiplexers after swapping of two adjacent cells; hence, the probabilities of having a 0 or 1 at a certain cell of the scan chain before applying the test vectors are equal.
Hence, the proposed design retains an important feature of any random TPG. In the BS-LFSR, consider the case that c1 will be swapped with c2 and c3 with c4, . . . , cn−2 with cn−1 according to the value of cn which is connected to the selection line of the multiplexers. In this case, we have the same exhaustive set of test vectors as would be generated by the conventional LFSR, but their order will be different and the overall transitions in the primary inputs of the CUT will be reduced. The proposed BS-LFSR for test-per-scan BISTs is based upon some new observations concerning the number of transitions produced at the output of an LFSR. Two cells in an n-bit LFSR are considered to be adjacent if the output of one cell feeds the input of the second directly (i.e., without an intervening XOR gate). Each cell in a maximal-length n-stage LFSR (internal or external) will produce a number of transitions equal to 2n−1 after going through a sequence of 2n clock cycles. The sequence of 1s and 0s that is followed by one bit position of a maximal-length LFSR is commonly referred to as an m sequence.

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

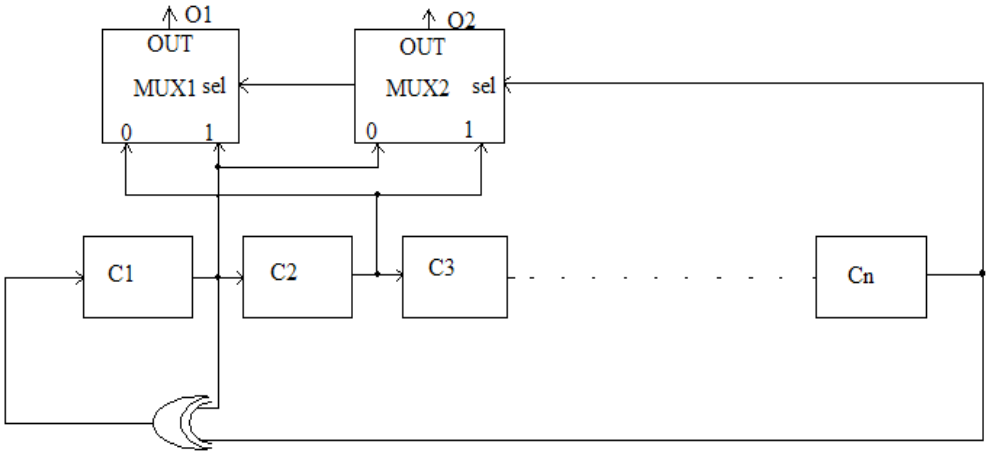

Fig.2 The proposed bit-swapping LFSR with polynominal x^n + x + 1

III. TEST PATTERN GENERATION FOR PATH DELAY AND STUCK-AT-FAULT

Tests for Delay Faults

Both a transition fault and a path delay fault are detected by a two-pattern test <p1, p2>. For a transition fault, the first pattern p1 assigns the initial transition value at the faulty line. The second pattern p2 assigns the final transition value at the faulty line and propagates the fault effect to the outputs. To detect the transition fault, a test <001, 101> is applied to "abd" as shown in Fig. 3. The value of a line under p1(p2) is shown on the left(right) of the arrow. The value shown on the left of the slash is the expected value under p2 if the circuit is fault free, and the value on the right of the slash is the faulty value under p2 if a fault exists. The transition fault can be detected if a 0 instead of a 1 is observed at output e at the required time point. For a path delay fault, p1 and p2 create a transition at the source of the target path, and p2 propagates it along the path. To detect the path delay fault, a test <0010, 1010> is applied to "abdf" as shown in Fig.4. The path delay fault can be detected if a 0 instead of a 1 is observed at output g at the required time point.

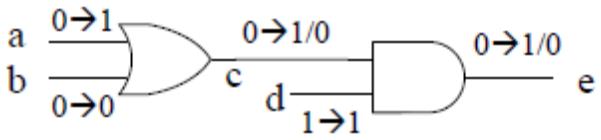

Fig.3. A test for the slow-to-rise transition fault at c

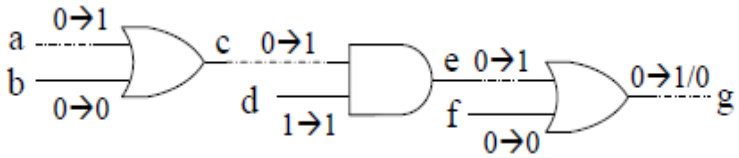

Fig.4. A test for the path delay fault associated with path a-c-e-g

Based on the propagation conditions used for the detection of path delay faults, tests for path delay faults can be categorized as robust and non-robust. A robust test guarantees the detection of a path delay fault regardless of the delays in the rest of the circuit.

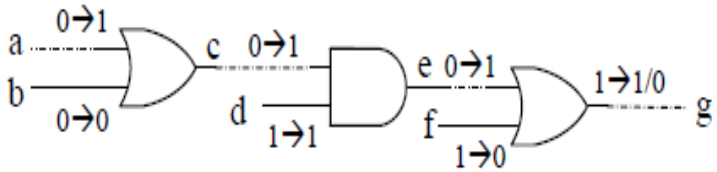

Fig.5 A non-robust test for the path delay fault associated with path a-c-e-g

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

For example, the test shown in Fig.4 is a robust test for the path delay fault. A non-robust test requires that the desired transition is created at the source of the target path and p2 statically sensitizes the path to enable the propagation of the transition along the path. A non-robust test detects a path delay fault if none of the off-path input signals arrive late. Otherwise, the test may be invalid. A non-robust test <0011, 1010> for the path delay fault is applied to "abdf" as shown in Fig.5. Different from the robust test in Fig.4, a falling transition occurs at off-path input f. If the transition at f does not arrive late, the path delay fault is detected if a 0 instead of a 1 is observed at g at the required time point. Otherwise, the value of g will always be 1 at the required time point even if the delay of path a-c-e-g exceeds the clock period. In this case, the test is not valid for the path delay fault.

Non-robust tests can be further categorized as strong non-robust and weak non-robust. Under a strong non-robust test, there is a transition that matches the transition at the source of the path on every line along the path, and every off-path input has a non-controlling value for the gate it drives under p2. Under a weak non-robust test, it is only required that every off-path input has a non-controlling value for the gate it drives under p2. A robust test for a path delay fault can detect all the transition faults along the path. However, a non-robust test for a path delay fault does not necessarily detect the transition faults along the path. Fig.6 shows a non-robust test for the path delay fault associated with path b-d-f-h and a rising transition at its source b. The rising transition fault at b is then targeted under the non-robust test, as shown in Fig.7

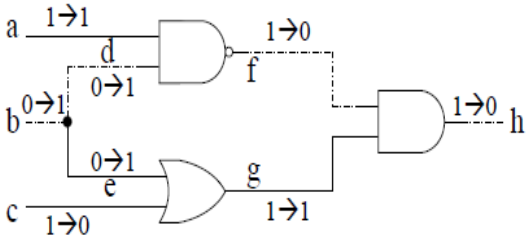

Fig.6 A non-robust test for the path delay fault associated with path b-d-f-h

Under the test, a 0 is observed at output h in both the faulty and fault free circuit. Therefore, the rising transition fault at b cannot be detected by the non-robust test.

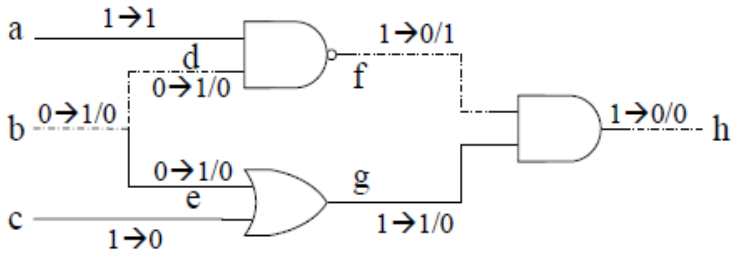

Fig.7 A transition fault under a non-robust test for a path delay fault

Tests for Stuck-at-fault

A stuck-at fault is a particular fault model used by fault simulators and automatic test pattern generation (ATPG) tools to mimic a manufacturing defect within an integrated circuit. Individual signals and pins are assumed to be stuck at Logical '1', '0' and 'X'. For example, an input is tied to a logical 1 state during test generation to assure that a manufacturing defect with that type of behavior can be found with a specific test pattern. Likewise the input could be tied to a logical 0 to model the behavior of a defective circuit that cannot switch its output pin. Not all faults can be analyzed using the stuck-at fault model. Compensation for static hazards, namely branching signals, can render a circuit untestable using this model. Also, redundant circuits cannot be tested using this model, since by design there is no change in any output as a result of a single fault.

Algorithm Flow for Stuck-at- Faults

A explanation of the ATPG method for up to double faults is as follows.

1) Generate a list of all eligible SA faults.
2) Pick a "Target fault", fi (If possible, give high priority to faults at fanout wires closest to primary outputs and low priority to primary inputs).
3) Generate a test pattern, vi, for fi.
4) Based on fi and vi, find the necessary path constraints for the fault propagation.
5) Find other stuck-at faults which violate the constraints and also mask the focus fault.

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

6) Add all DSA faults which consist of the focus fault paired with each of the path constraint violating faults to a temporary list of potentially undetected DSA faults.
7) Remove focus fault from list of remaining SSA faults.
8) If there are remaining SSA faults, repeat from step
2. Else go to step 9.
9) Go through the temporary list of potentially undetected DSA faults, and keep only the faults which are listed twice, i.e. {fi, fj} and {fj, fi}
10) Perform TPG on each remaining DSA fault.

IV. TEST POINT INSERTION

Test point insertion involves adding control and observation points to the circuit-under-test in a way that the system function remains the same, but the testability is improved. An observation point is an additional primary output that is inserted in the circuit to increase the observability of faults in the circuit. In the example, an observation point is inserted at the output of gate G1 such that faults are observable regardless of the logic value at node y. Control points inserted in the circuit such that when it is activated, it fixes the logic value at a particular node to increase the controllability of some faults in the circuit. A control point can also affect the observability of some faults in the circuit because it can change the propagation paths in the circuit.

Test points can be inserted to put the uncontrollable or unobservable logics into the scan chain. By using this technique, these logics can be tested, and test coverage and test efficiency can be improved greatly. Test Point Insertion (TPI) is a useful technique for solving the potential testability problem and improving the test coverage of a design by making its uncontrollable logic controllable and unobservable logic observable. This technique also helps to improve the test efficiency since the higher coverage can be derived with few test vectors increasing. This technique is very easy to put into application since only a few commands are added in the existing scripts.

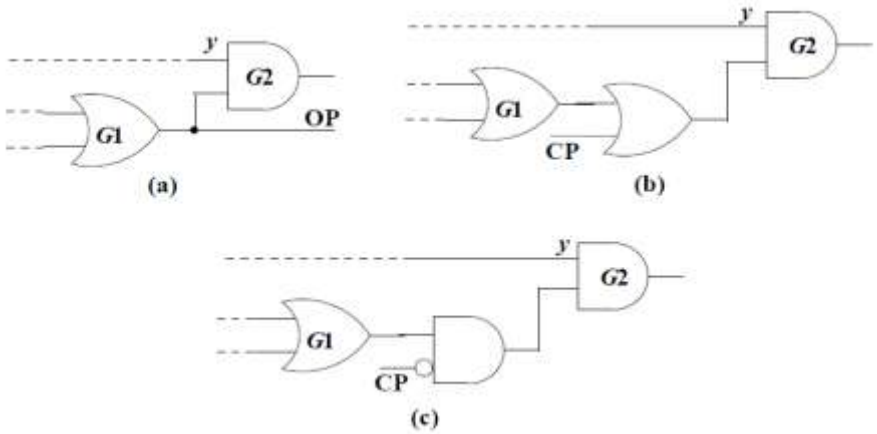

Fig.8. Example Illustrating Test Points: a) Observation Point, b) 1-Control Point, and c) 0-Control Point

An observation point is an additional primary output that is inserted in the circuit to increase the observability of faults in the circuit. Fig.8 shows an observation point is inserted at the output of gate G1 such that faults are observable regardless of the logic value at node y. A control point is inserted in the circuit such that when it is activated, it fixes the logic value at a particular node to increase the controllability of some faults in the circuit. A control point can also affect the observability of some faults in the circuit because it can change the propagation paths in the circuit. In the example in Figure (b), a control point is inserted to fix the logic value at the output of gate G1 to a 1 when the control point is activated (this is called a control-1 point). This is accomplished by placing an OR gate at the output of gate G1. In the example in Figure (c), a control point is inserted to fix the logic value at the output of gate G1 to a 0 when the control point is activated (this is called a control-0 point). This is accomplished by placing an AND gate at the output of gate G1. During system operation, the control points are not activated and hence, they do not affect the system function. However, control points do add an extra level of logic to some paths in the circuit. If a control point is placed on a critical timing path, it can increase the cycle time of the circuit. Since test points add both area and performance overhead, it is important to try to minimize the number of test points that are inserted to achieve the desired fault coverage. Optimal test point placement for circuits with reconvergent fan-out has been shown to be NP-complete

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

V. EVALUATION RESULTS

Table I show the comparison result for various benchmark circuit using normal LFSR and bit-swapping LFSR. Bit-swapping LFSR along with observation point insertion results in improved fault coverage for stuck-at and path delay faults. The proposed method also results in low power consumption compares with existing method.

Table I. Comparison Results of Benchmark circuits

Circuits	S27 LFSR	S27 BS-LFSR	S208 LFSR	S208 BS-LFSR	S298 LFSR	S298 BS-LFSR
Gates	10	10	112	112	133	133
FFS	3	3	8	8	14	14
PIs	4	4	10	10	3	3
Pos	1	1	1	1	6	6
LFSR	4	4	10	10	10	10
Total Seeds	8	6	15	10	10	10
Eff Seeds	4	3	8	3	4	5
Path Targ	8	8	10	10	10	10
Path Det	6	8	6	10	6	10
S-a-f Targ	5	5	5	5	5	5
S-a-f Det	4	5	3	5	3	5
Power(mW)	125.43	117.14	147.54	142.08	168.91	159.61

Fig.9 shows the simulation result of test pattern generation for ISCAS'89 benchmark circuit using bit-swapping LFSR. Bit-swapping LFSR reduces the transition which occurs in test pattern by using multiplexer. The fault coverage is improved by using observation point insertion which controls the internal path for propagating faulty values.

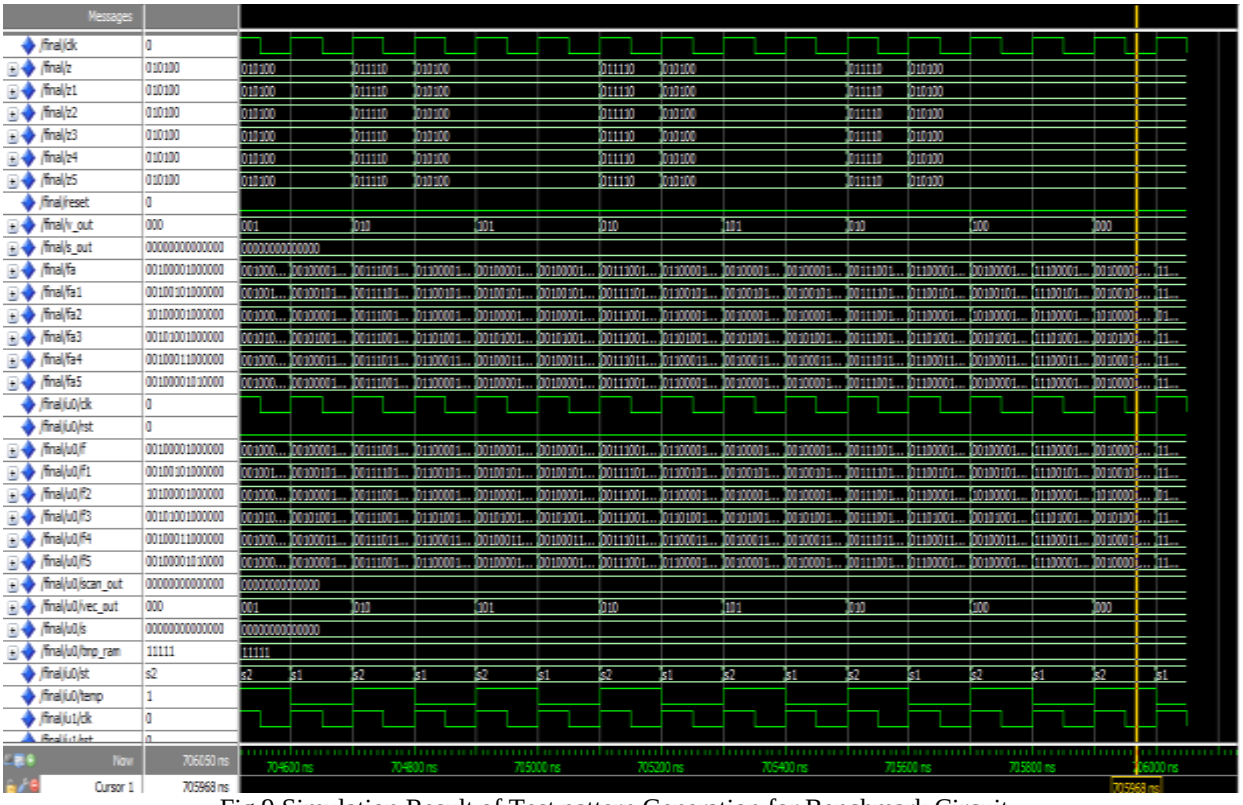

Fig.9 Simulation Result of Test pattern Generation for Benchmark Circuit.

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 7, Issue 4, April 2019

VI. CONCLUSION

BS-LFSR-based test generation targeting path delay faults and stuck-at fault are proposed. Bit-Swapping LFSR results in low transitions compared to conventional LFSR. Fault model to detect long path delay and stuck-at-fault are analyzed. The proposed method also helps in identifying the undetectable faults. Initially random seeds for the BS-LFSR are modified into seeds that produce even if the available tests cannot be compressed into seeds. Further the undetectable faults are analyzed using test point insertion, improving the fault coverage. Evaluation results for benchmark circuits to detect path delay faults and stuck-at fault shows that the proposed BS-LFSR result in low power consumption compared to conventional design.

REFERENCES

[1]. G. L. Smith, "Model for Delay Faults Based Upon Paths", in Proc. Intl. Test Conf., 1985, pp. 342-349.
[2]. W.-N. Li, S. M. Reddy and S. K. Sahni, "On Path Selection in Combinational Logic Circuits", IEEE Trans. on Computer-Aided Design, Jan. 1989, pp. 56-63.
[3]. M. H. Schultz, K. Fuchs and F. Fink, "Advanced Automatic Test Pattern Generation Techniques for Path Delay Faults", in Proc. Intl. Symp. On Fault-Tolerant Computing, 1989, pp. 44-51.
[4]. P. Varma, "On Test Generation for Path Delay Faults in ASICs", in Proc. VLSI Test Symp., 1992, pp. 19-24.
[5]. L.-C. Chen, S. K. Gupta and M. A. Breuer, "High Quality Robust Tests for Path Delay Faults", in Proc. VLSI Test Symp., 1997, pp. 88-93.
[6]. S. Tani, M. Teramoto, T. Fukazawa and K. Matsuhiro, "Efficient Path Selection for Delay Testing based on Partial Path Evaluation", in Proc. VLSI Test Symp., 1998, pp. 188-193.
[7]. A. Murakami, S. Kajihara, T. Sasao, I. Pomeranz and S. M. Reddy, "Selection of Potentially Testable Path Delay Faults for Test Generation", in Proc. Intl. Test Conf., 2000, pp. 376-384.
[8]. J.-J. Liou, A.Krstic, Y.-M.Jiang & K.-T. Cheng, "Path Selection & Pattern Generation for Dynamic Timing Analysis Considering Power Supply Noise Effects", in Proc. Intl. Conf. on Computer-Aided Design, 2000, pp. 493-496.
[9]. J. Jiang, M. Sauer, A. Czutro, B. Becker and I. Polian, "On the Optimality of K Longest Path Generation Algorithm under Memory Constraints", in Proc. Design, Autom. & Test in Europe Conf., 2012, pp. 418-423.
[10]. J. Chung, J. Xiong, V. Zolotov and J. A. Abraham, "Testability-Driven Statistical Path Selection", IEEE Trans. on Computer-Aided Design, Aug. 2012, Vol. 31, No. 8, pp. 1275-1287.
[11]. J. Chung and J. A. Abraham, "Concurrent Path Selection Algorithm in Statistical Timing Analysis", IEEE Trans. on VLSI Systems, Sept. 2013, Vol. 21, No. 9, pp. 1715-1726.
[12]. P. Das and S. K. Gupta, "Extending Pre-silicon Delay Models for Postsilicon Tasks: Validation, Diagnosis, Delay Testing, and Speed Binning", in Proc. VLSI Test Symp., 2013, pp. 1-6.
[13]. B. Yao, A. Sinha and I. Pomeranz, "Path Selection Based on Static Timing Analysis Considering Input Necessary Assignments", in Proc. VLSI Test Symp., 2013, pp. 1-6.
[14]. C.-Y. Chang, K.-Y. Liao, S.-C. Hsu, J. C.-M. Li and J.-C. Rau, "Compact Test Pattern Selection for Small Delay Defect", IEEE Trans. on Computer-Aided Design, June 2013, Vol. 32, No. 6, pp. 971-975.