Evaluation of Fibonacci Test Pattern Generator for Cost Effective IC Testing

Md. Tanveer Ahmed, Liakot Ali

Department of Information and Communication Technology
Institute of Information and Communication Technology
Bangladesh University of Engineering and Technology, Bangladesh

cs_tanveer@yahoo.com

Abstract- Integrated Circuits (ICs) are the key components in all modern electronic equipments. With the increase in complexities of ICs, it is a challenging issue to test ICs at low cost with reliable performance. In this paper we investigate the performance of IC. It is shown that Fibonacci Linear Feedback Shift Register (FLFSR) performs better in testing of IC. This paper presents a new architecture for test pattern generator that produces the highest fault coverage (FC) with minimum number of pseudo random test vectors. This paper focuses on the design and implementation of a 64-bit Fibonacci test pattern generator capable of generating sufficient long test pattern and conducting fault simulation experiments on International Symposium on Circuits and Systems (ISCAS) benchmark circuits. Test pattern generator is very important in VLSI Testing. By changing the seed and feedback connection, a set of test vectors was generated for different benchmark circuits. The objective was to produce Test Pattern with good randomness; then fault coverage will be better. Fault simulation was done using FSIM fault simulator.

Keywords - Linear Feedback Shift Register (LFSR); Fibonacci Linear Feedback Shift Register (FLFSR); Galois Linear Feedback Shift Register (GLFSR); Fault Coverage (FC); Test Vector (TV)

I. INTRODUCTION

With the dramatic improvement of semiconductor technology, the design complexities and packing densities of ICs have exceedingly increased. In IC manufacturing, various physical defects may occur during numerous physical, chemical and thermal processes. With the increase of the complexities of VLSI circuit, testing problem has become more acute [1]. To achieve the IC testing at low cost with reliable performance, researchers have proposed different testing approaches [2-5], among which mixed mode technique outperforms all the other proposed techniques. In mixed mode technique, Circuit Under Test (CUT) is first subjected under pseudo-random testing mode and then at an optimum point of fault coverage it is switched to deterministic test mode. In mixed mode technique, pseudo-random test pattern generation and proper switching point from pseudo-random test mode to deterministic test mode are very important to make the testing process cost-effective. Recently, Fibonacci pseudo-random test pattern generator has been proved efficient in many cryptographic applications because of its better randomness [6]. Usually, Linear Feedback Shift Register (LFSR) and Cellular Automata Register (CAR) are popular in pseudo-random test pattern generation. Due to the limitations of integration and fabrication technology, previous researchers used 32-bit LFSR or CAR in designing IC tester or test processor chip. However, now integration technology has tremendously improved. In this project, a 64-bit Fibonacci test pattern generator, which is a modified version of LFSR, has been evaluated in VLSI testing. Fibonacci test pattern generator has already been used in many cryptographic applications and proved very much efficient. So, there are scopes of research to evaluate its effectiveness in VLSI testing.

Fig. 1 Fault Coverage versus Random test vectors [16]
A typical fault detection curve during fault simulation is shown in Fig. 1. When simulation begins, a large percentage of faults are detected in a short amount of time. However, as time goes on, the rate at which faults are detected decreases because the test patterns applied detect many faults that have already been detected. If these detected faults are not dropped, extra time is spent on resimulating these faults but the fault coverage remains the same.

II. LINEAR FEEDBACK SHIFT REGISTER (LFSR)

A. Basic Description

LFSRs in the simplest definition are used as pseudorandom number generators. An LFSR with a well-chosen feedback function can produce a sequence of bits that appears random and has a very long cycle [9]. The initial value of the LFSR is called the seed. The bit positions in the LFSR state, which influences the input, are called taps. These are chosen based on the primitive polynomial. The arrangement of taps for feedback in an LFSR can be expressed in finite field arithmetic as a polynomial mod 2. This means that the coefficients of the polynomial must be 1’s or 0’s. This is called the feedback polynomial or the characteristic polynomial. For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is \(1 + x^{11} + x^{13} + x^{14} + x^{16}\). When properly configured for maximum length sequences, each state will be reached only once until every state has been reached. Once every state has been reached, the sequence will be repeated.

The applications of LFSR include pseudorandom number generator, random pattern generator and analyzer, encryption/decryption and direct sequence spread spectrum for digital signal processing. There are two major implementations of LFSR, namely the Fibonacci LFSR (FLFSR) and Galois LFSR (GLFSR). Figs. 2.1 and 2.2 show these two types of LFSR each with characteristics polynomial \(P(x) = 1 + c_1x + c_2x^2 + \ldots + c_nx^n\). If a connection exists, then \(c_i = 1\), otherwise \(c_i = 0\). The Fibonacci implementation has logic in the feedback path, whereas the Galois implementation has an output that is fed back to selected points in the feed forward path.

B. Galois Linear Feedback Shift Register (GLFSR)

As shown in Fig. 2.1, the data flow is from left to right and the feedback path is from right to left. The polynomial increments from left to right with the \(x^0\) term (the “1” in the polynomial) as the first term. This is referred to as a Tap polynomial, as it indicates which taps are to be fed back from the shift register. Since the XOR gate is in the shift register path, the Galois implementation is also known as an in-line or modular type (M-type) LFSR.

![Fig. 2.1 Structure of Galois LFSR](image)

![Fig. 2.2 Structure of Fibonacci LFSR](image)
C. Fibonacci Linear Feedback Shift Register (FLFSR)

In Fig. 2.2, the data flow is from left to right and the feedback path is from right to left, similar to the Galois implementation. However, in Fibonacci implementation polynomial decrements from left to right with \( X^0 \) as the last term in the polynomial. This polynomial is referred to as a Reciprocal Tap polynomial and the feedback taps are incrementally annotated from right to left along the shift register. Since the XOR gate is in the feedback path, the Fibonacci implementation is also known as an out-of-line or simple type (S-type) LFSR. In this study, this type of LFSR was focused on for the experiments.

III. DESIGN AND IMPLEMENTATION

A. Test Pattern generation using FLFSR

First a 64-bit Fibonacci test pattern generator was designed using C programming language. It is user programmable in terms of tap position and seed. Since it is 64 bit, it is capable of generating sufficient long test pattern. Then, test pattern was generated for different benchmark circuits with respect to different seed values and different tap positions of the proposed pseudo-random pattern generator and was saved in different files. Then, fault simulation experiments were conducted using FSIM [8] fault simulator on the different benchmark circuits using the generated test pattern and respective fault coverage was recorded. Finally, the fault simulation results were compared with those of other researchers.

B. Feedback Logic

In order for an LFSR to iterate through its largest possible sequence of values, it must use a polynomial that will produce such a sequence. The tap positions shown in Fig. 3.1 produce maximum sequence lengths for the proposed 64-bit FLFSR [9]. The LFSR feedback function performs modulo-2 summation. These summations can be performed with either XOR or XNOR gates in the logic. The design uses the Fibonacci approach to implement test pattern generator. Figure 3.1 shows the feedback logic using XOR for the proposed 64-bit FLFSR with 4 taps. The outputs of the stages of Q64, Q63, Q61 and Q60 were XORed as shown in Fig. 3.1 and the output FB_Out was fed back to the first stage of the FLFSR.

C. Illustration of the FLFSR working

For simplicity, a simple 4-bit FLFSR is considered here. The output of stage 3 and 4 are XORed and fed back to the first stage of the FLFSR. The primitive polynomial for degree 4 is \( 1 + x^3 + x^4 \). This demonstrates the proposed test pattern generator. With the application of each clock, the value of each stage of the FLFSR is changed as follows. Assuming the initial value of the FLFSR is 0101. From Figure 3.2 (b), it is seen that the state of the FLFSR repeats after \( 2^4-1=15 \) clock cycles.
Figure 3.2 (a) Block Diagram, (b) Truth table, (c) State diagram of 4-bit FLFSR with characteristics polynomial $1 + x^3 + x^4$

Note that in Fig. 3.2(a), the outputs of the 3rd and 4th stage of the LFSR are XORed and fed back to the first stage. Since the LFSR has four stages, the truth table in Fig. 3.2(b) shows that it has 15 different states. After the 15th clock cycle the LFSR repeats its states. The 15 distinct states of the LFSR are depicted with the state diagram in Fig. 3.2(c).

IV. FAULT SIMULATION RESULTS

Pseudo-random testing is a cost-effective means of testing VLSI circuits. Using Fibonacci pseudo-random test patterns, it is possible to achieve a maximum percentage of fault coverage by only applying fewer number of test vectors. This fact was verified in this Paper. The seed of an FLFSR is defined as the initial value of the stages of the FLFSR before starting to generate the test vectors. Forty different seeds were used to generate PRV sequences. The PRV sequences were applied to the benchmark circuits, and fault coverage versus number of PRV was measured with respect to every seed. For seed of the FLFSR in the experiment, one of the stages of the FLFSR was set to ‘1’ and others to ‘0’, and in this paper, ‘1’ is mentioned as seed for simplicity. A lot of variations were done to improve the fault coverage.

Fig. 4.1 Fault simulation result of circuit c432.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)

Fig. 4.2 Fault simulation result of circuit c499.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)

Fig. 4.3 Fault simulation result of circuit c2670.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)

Fig. 4.4 Fault simulation result of circuit c3540.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)

Fig. 4.1 Fault simulation result of circuit c432.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)

Fig. 4.4 Fault simulation result of circuit c3540.bench (for feedback polynomial $1+x^60+x^61+x^62+x^63$)
Figure 4.2 Fault simulation result of circuit c499.bench (for feedback polynomial 1+x^{60}+x^{61}+x^{63}+x^{64})

Figure 4.3 Fault simulation result of circuit c2670.bench (for feedback polynomial 1+x^{60}+x^{61}+x^{63}+x^{64})

Figure 4.4 Fault simulation result of circuit c3540.bench (for feedback polynomial 1+x+x^3+x^4+x^{64})

Forty different seeds were used to generate PRV sequences. The PRV sequences were applied to the benchmark circuits and fault coverage (%) versus number of PRVs was measured with respect to every seed. Figure 4.5 shows the fault detection profile of the PRV sequences for the benchmark circuit c432.bench. Note that the x-axis of Fig. 4.5 contains 40 slots. Each slot contains 5 test vectors to present the curve simply. So, for the 40 slots, there are 200 test vectors.

Figure 4.5 Fault detection profile of PRV for the benchmark circuit c432.bench

- 32 -
Figure 4.5 shows that the first few PRVs detected the maximum faults of the circuit c432.bench. Then the slope of the fault detection profile of the PRV rapidly decreased with the increase of the number of test vectors. More than 80 percent faults were detected using only 50 test vectors. These faults were ETD faults. After the detection of the ETD faults, much higher number of test vectors was needed to detect the remaining faults. These remaining faults were HTD faults and random resistant faults. These faults caused potential difficulties in achieving acceptable fault coverage in the pseudo-random testing of IC. Fault detection profiles of the PRV sequences for the rest of the benchmark circuits were similar to that as shown in Fig. 4.5. The figure clearly indicates that with the increase of the number of test vectors, increase of fault coverage sharply decreases and approximates zero. When the increment of fault coverage was very low or almost zero, the mode of test was switched from pseudo-random test to deterministic test. For example, in the simulation result as shown in Figure 4.1, when the number of PRV is 200 for circuit c432.bench, then it is appropriate to switch from the pseudo-random test mode to deterministic mode. Fault simulation results for the rest of the benchmark circuits followed the similar profile.

To analyze the effect of reseeding and polynomial programmability on achieving full fault coverage, experiments were performed on different ISCAS85 benchmark circuits. Table 1 presents two samples of 64-degree primitive polynomial such as \(1+x^{60}+x^{51}+x^{63}+x^{64}\) and \(1+x+x^{3}+x^{4}+x^{64}\) were chosen. For any of the primitive polynomials, the position of the seed was changed from first stage to last stage of the FLFSR and with respect to every bit position, a test vector file was generated.

<table>
<thead>
<tr>
<th>ISCAS85 Benchmark Circuits</th>
<th>*NTV</th>
<th>Polynomial1 ((1+x+x^{3}+x^{4}+x^{64}))</th>
<th>Polynomial2 ((1+x^{60}+x^{51}+x^{63}+x^{64}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>224</td>
<td>98.28</td>
<td>98.47</td>
</tr>
<tr>
<td>c499</td>
<td>352</td>
<td>97.89</td>
<td>98.15</td>
</tr>
<tr>
<td>c880</td>
<td>120</td>
<td>90.13</td>
<td>92.15</td>
</tr>
<tr>
<td>c1355</td>
<td>450</td>
<td>95.30</td>
<td>96.76</td>
</tr>
<tr>
<td>c1908</td>
<td>930</td>
<td>94.84</td>
<td>96.49</td>
</tr>
<tr>
<td>c2670</td>
<td>250</td>
<td>82.71</td>
<td>83.55</td>
</tr>
<tr>
<td>c3540</td>
<td>540</td>
<td>1(\text{91.80})</td>
<td>91.57</td>
</tr>
<tr>
<td>c5315</td>
<td>600</td>
<td>98.34</td>
<td>98.26</td>
</tr>
<tr>
<td>c6288</td>
<td>60</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*FC= Fault Coverage  
*NTV= No. of Test Vectors

The screenshot of fault simulation result on the ISCAS85 benchmark circuits using the PRV sequences generated by the proposed FLFSR is given in Fig. 4.6.
Figure 4.6 shows the screenshot of fault simulation result of circuit c432.bench. First it shows the numbers of inputs, outputs, gates and the level of the circuit. The name of the test vector file, output1.test, is also shown. The figure shows that the percentage of fault coverage was 98.282 using 200 test vectors. Among the 524 faults, 515 faults were detected and the remaining 9 faults were undetected. It also shows the amount of memory used and the total CPU time required.

Fault simulation experiments were conducted using FSIM digital fault simulator [8] on ISCAS85 benchmark circuits. Summary of the fault simulation results of the ISCAS85 benchmark circuits using the proposed 64-bit FLFSR is presented in Table 2. The table shows the total number of test vector including deterministic vectors required to achieve complete fault coverage for ISCAS benchmark circuit. It shows that 100% fault coverage can be achieved using the proposed technique.

<table>
<thead>
<tr>
<th>ISCAS85 Benchmark Circuits</th>
<th>Total Number of Faults Inserted</th>
<th>Number of Test Vectors</th>
<th>Total Number of Test Vectors</th>
<th>% Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Random</td>
<td>Deterministic</td>
<td></td>
</tr>
<tr>
<td>c432</td>
<td>802</td>
<td>200</td>
<td>9</td>
<td>209</td>
</tr>
<tr>
<td>c499</td>
<td>1306</td>
<td>190</td>
<td>25</td>
<td>215</td>
</tr>
<tr>
<td>c880</td>
<td>1428</td>
<td>120</td>
<td>61</td>
<td>181</td>
</tr>
<tr>
<td>c1355</td>
<td>1970</td>
<td>180</td>
<td>126</td>
<td>306</td>
</tr>
<tr>
<td>c1908</td>
<td>1282</td>
<td>880</td>
<td>69</td>
<td>949</td>
</tr>
<tr>
<td>c2670</td>
<td>2588</td>
<td>250</td>
<td>452</td>
<td>702</td>
</tr>
<tr>
<td>c3540</td>
<td>2988</td>
<td>540</td>
<td>281</td>
<td>821</td>
</tr>
<tr>
<td>c5315</td>
<td>5640</td>
<td>560</td>
<td>91</td>
<td>651</td>
</tr>
<tr>
<td>c6288</td>
<td>9804</td>
<td>60</td>
<td>41</td>
<td>101</td>
</tr>
</tbody>
</table>

V. COMPARISON

Table 3 presents the summary of the above fault simulation results by FLFSR generated using FSIM. It also compares the results with those obtained from weighted random method used by other researchers.

<table>
<thead>
<tr>
<th>ISCAS85 Benchmark Circuits</th>
<th>*NTV1</th>
<th>*NTV2</th>
<th>*NTV3</th>
<th>*NTV4</th>
<th>*NTV5</th>
<th>*NTV6</th>
<th>*NTV7</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>209</td>
<td>214</td>
<td>224</td>
<td>352</td>
<td>320</td>
<td>512</td>
<td>1024</td>
</tr>
<tr>
<td>c499</td>
<td>215</td>
<td>225</td>
<td>512</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c880</td>
<td>181</td>
<td>248</td>
<td>160</td>
<td>4544</td>
<td>416</td>
<td>260</td>
<td>1280</td>
</tr>
<tr>
<td>c1355</td>
<td>306</td>
<td>314</td>
<td>512</td>
<td>1248</td>
<td>1664</td>
<td>2244</td>
<td>2098</td>
</tr>
<tr>
<td>c1908</td>
<td>949</td>
<td>969</td>
<td>992</td>
<td>4608</td>
<td>2496</td>
<td>2308</td>
<td>5376</td>
</tr>
<tr>
<td>c2670</td>
<td>702</td>
<td>724</td>
<td>288</td>
<td>-</td>
<td>6240</td>
<td>10766</td>
<td>5888</td>
</tr>
<tr>
<td>c3540</td>
<td>821</td>
<td>271</td>
<td>640</td>
<td>1065</td>
<td>9504</td>
<td>12220</td>
<td>3840</td>
</tr>
<tr>
<td>c5315</td>
<td>651</td>
<td>388</td>
<td>640</td>
<td>1632</td>
<td>1950</td>
<td>1316</td>
<td>2048</td>
</tr>
<tr>
<td>c6288</td>
<td>101</td>
<td>234</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*NTV1 = Number of test vectors required using 64-bit FLFSR based mixed-mode technique in the present work
*NTV2 = Number of test vectors required using 32-bit GLFSR based mixed-mode technique [12]
*NTV3 = Number of test vectors required using 32-bit LFSR based mixed-mode technique [1]
*NTV4 = Number of test vectors using weighted random technique [17]
*NTV5 = Number of test vectors using weighted random technique [18]
*NTV6 = Number of test vectors using weighted random technique [19]
*NTV7 = Number of test vectors using weighted random technique [20]

Fault simulation results of the benchmark circuits c499.bench and c6288.bench from other researchers are not available. The ‘− ‘ sign in Table 4 indicates the unavailability of the actual data. The comparison showed that the proposed Fibonacci test pattern generator in mixed mode approach was capable of producing 100% fault coverage using much smaller number of test vectors than other researchers.

- 34 -
VI. BEST SEED DETERMINATION

A. Coefficient of Variation (CV)

Coefficient of Variation (CV) is a relative measure in Statistics. This measure developed by Karl Pearson is the most commonly used measure of relative variation. It is used in such problems where comparing the variability of two or more than two series is needed. The series for which the coefficient of variation is greater is said to be more variable or conversely less consistent, less uniform, less stable, and less homogeneous. On the other hand, the series for which the coefficient of variation is less is said to be less variable or more consistent, more uniform, more stable, and more homogeneous. CV is obtained as follows:

\[ CV = \frac{\sigma}{\bar{X}} \times 100, \]

where \( \sigma \) = Standard Deviation and
\( \bar{X} \) = Arithmetic Mean.

To measure the CV of a PRV sequence for a particular seed for a benchmark circuit, MATLAB R2012b was used. To compare the values of CVs of PRV sequences generated for different seeds, a number of different seeds were selected randomly. The CVs of PRV sequences for different seeds were compared. The result is given in Table 4.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Seed 6</th>
<th>Seed 15</th>
<th>Seed 23</th>
<th>Seed 24</th>
<th>Seed 26</th>
<th>Seed 35</th>
<th>Seed 38</th>
<th>Seed 42</th>
<th>Seed 51</th>
<th>Seed 59</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>118.54</td>
<td>115.58</td>
<td>116.51</td>
<td>119.01</td>
<td>118.41</td>
<td>118.30</td>
<td>118.30</td>
<td>115.31</td>
<td>113.86</td>
<td>113.77</td>
</tr>
<tr>
<td>c499</td>
<td>109.51</td>
<td>112.39</td>
<td>111.25</td>
<td>111.70</td>
<td>108.25</td>
<td>110.23</td>
<td>109.35</td>
<td>110.13</td>
<td>111.47</td>
<td>110.88</td>
</tr>
</tbody>
</table>

From Table 5 it is observed clearly that the value of CV of the PRV sequence is the maximum for the seed 24 for benchmark circuit c432.bench. Therefore, it can be concluded that the PRV sequence generated for seed 24 is more random as compared to other seeds. As a result, seed 24 for circuit c432.bench was determined as the best seed. In a similar way, seed 7 for circuit c499.bench was the best seed. Results of CVs for the rest of the benchmark circuits followed the similar profile.

VII. CONCLUSIONS

This paper highlights the design and implementation of a 64-bit Fibonacci test pattern generator capable of generating sufficient long test patterns. Experiments were carried out on different seeds and primitive polynomials to achieve highest percentage of fault coverage. In the simulation results, it was shown that by changing the seeds and feedback polynomial, fault coverage was improved with lower number of test vectors than using single polynomial and single seed. In the comparison section, it was shown that the proposed Fibonacci test pattern generator in mixed mode approach was capable of producing the highest fault coverage using much lower number of test vectors than other researchers. Moreover, the best seed and optimum switching point were examined by conducting fault simulation experiments on ISCAS benchmark circuits. Determination of the best seed was again verified by calculating the coefficient of variation of the random sequences that have been applied to the benchmark circuits. Based on the result in this paper, initiative can be taken for designing low cost IC Tester.

ACKNOWLEDGMENT

The author is grateful to IICT, BUET for providing facilities for the research, and all the reviewers for their excellent comments.

REFERENCES


