

ANALYSIS OF AN EFFICIENT FAULT TOLERANT LINEAR FEEDBACK SHIFT REGISTER FOR VLSI APPLICATIONS

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering degree in

Electronics and Communication Engineering

by

VAISHALI .M (38130229)

GNANA AKALYA .M (38130069)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALA, CHENNAI – 600 119

www.sathyabama.ac.in



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **VAISHALI .M (38130229)** and **GNANA AKALYA .M (38130069)** who carried out the project entitled “**ANALYSIS OF AN EFFICIENT FAULT TOLERANT LINEAR FEEDBACK SHIFT REGISTER FOR VLSI APPLICATIONS**” under our supervision from November 2021 to May 2022.

Internal Guide

Mrs. L .MAGTHELIN THERASE M.E., (Ph.D.)

Head of the department

Dr.T.RAVI. M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **VAISHALI .M (38130229)** and **GNANA AKALYA .M (38130069)** hereby declare that the Project Report entitled “**ANALYSIS OF AN EFFICIENT FAULT TOLERANT LINEAR FEEDBACK SHIFT REGISTER FOR VLSI APPLICATIONS**” done by us under the guidance of **Mrs.L.MAGTHELIN THERASE, M.E., (Ph.D.)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE:

1.

2.

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA. M.E., Ph.D., Dean, School of Electrical and Electronics** and **Dr. T. RAVI. M.E., Ph.D., Head of the Department, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our Project Guide **Mrs. L. MAGTHELIN THERASE, M.E., (Ph.D.)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

VLSI is the process of integrating thousands of transistors into a single chip. VLSI design is mainly used to minimize the interconnecting fabrics area. The linear feedback shift register (LFSR) is one of the most widely used sequential logics, which produces the linear function of its previous state. The conventional LFSR consists of shift registers and XORs. Linear Feedback Shift Register (LFSR) is the main component of the embedded systems that are based on pseudorandom bit sequences such as pseudo-random number and pseudo-noise sequence generators, fast digital counters, and Built-In-Self Tests. Hence, fault tolerant design of LFSR is essential for the applications with high reliability demands. Traditional fault tolerant LFSRs include large number of Single-Point-of-Failures (SPoFs) in which any fault results in the whole system failure. To overcome that single point failures a new fault tolerant architecture is proposed for LFSR named as Fault Tolerant-Linear Feedback Shift Register (FT-LFSR) in which the number of SPoFs are significantly reduced. To this end, a modified version of Triple Modular Redundancy (TMR) empowered with some extra controlling units for identifying the operational module is used in this project. Triple Modular Redundancy (TMR) is a well-known and common fault tolerant technique based on spatial redundancy. In TMR, the design is triplicated to perform a process in parallel and their result is processed by a majority-voting system to produce a single output. Hence the number of single point failures are reduced in this proposed model. Experimental results in Xilinx show that, the FT-LFSR is resilient to all single transient and permanent faults except in its limited SPoFs and many patterns of multiple faults.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PG NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF TABLES	xiii
1	INTRODUCTION	1
	1.1 MOTIVATION	2
	1.2 LINEAR FEEDBACK SHIFT REGISTER	3
	1.2.1 Many-to-one implementation	5
	1.2.2 One-to-many implementation	8
	1.3 FIBONACCI LINEAR FEEDBACK SHIFT REGISTER	10
	1.3.1 4-bit Fibonacci Linear Feedback Shift Register	10
	1.3.2 16-bit Fibonacci Linear Feedback Shift Register	11
	1.4 GALOIS LINEAR FEEDBACK SHIFT REGISTER	14
	1.4.1 16-bit Galois Linear Feedback Shift Register	14
	1.4.2 Non-binary Galois Linear Feedback Shift Register	17
	1.5 MAXIMAL LENGTH LINEAR FEEDBACK SHIFT REGISTER	17
	1.6 XOR SHIFTS LINEAR FEEDBACK SHIFT REGISTER	18
	1.7 MATRIX FORMS OF LFSR	20
	1.8 OUTPUT STREAM PROPERTIES OF LFSR	21
	1.9 POLYNOMIALS FOR MAXIMAL LFSRs	21

1.10	FACTORS AFFECTING COST OF CHIP AND ABOUT TESTING FOR POWER DISSIPATION	23
1.11	EFFECTS OF LARGE POWER DISSIPATION	24
1.12	SOC FOR POWER CONSUMPTION	25
1.13	OBJECTIVES OF THE PROJECT	25
1.14	ORGANIZATION OF THE REPORT	26
2	LITERATURE SURVEY	27
2.1	LITERATURE SURVEY	27
2.2	LITERATURE SUMMARY	34
3	EXISTING SYSTEM	35
3.1	FAULT TOLERANT LINEAR FEEDBACK SHIFT REGISTER	35
3.2	EXISTING FT-LFSR	36
3.3	EXISTING PROCEDURES	37
3.4	RELIABILITY-AREA-FACTOR (RAF) METRIC	38
4	PROPOSED METHODOLOGY	40
4.1	PROPOSED FT-LFSR	40
4.2	EXTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR	42
4.3	INTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR	43
4.3.1	XOR gate	44
4.3.2	XNOR gate	46
4.3.3	Multiplexer	47
4.3.4	Comparator	49

LIST OF FIGURES

FIGURE NO.	TITLE	PG NO.
1.1	LINEAR FEEDBACK SHIFT REGISTER	3
1.2	5-BIT LINEAR FEEDBACK SHIFT REGISTER	4
1.3	LFSR WITH XOR FEEDBACK PATH	5
1.4	COMPARISON OF ALTERNATIVE TAP SELECTIONS	6
1.5	COMPARISON OF XOR VERSUS XNOR FEEDBACK PATH	7
1.6	TAPS FOR MAXIMAL LENGTH LFSRs WITH 2 TO 32 BITS	8
1.7	ONE-TO-MANY VERSUS MANY-TO-ONE IMPLEMENTATIONS	9
1.8	FIBONACCI LINEAR FEEDBACK SHIFT REGISTER	10
1.9	4-BIT LINEAR FEEDBACK SHIFT REGISTER WITH ITS STATE DIAGRAM	11
1.10	16-BIT LINEAR FEEDBACK SHIFT REGISTER WITH ITS STATE DIAGRAM	12
1.11	GALOIS LINEAR FEEDBACK SHIFT REGISTER	14
1.12	16-BIT GALOIS LINEAR FEEDBACK SHIFT REGISTER	15
3.1	LFSR STRUCTURE (a)FIBONACCI IMPLEMENTATION (b) GALOIS IMPLEMENTATION	37
4.1	BLOCK DIAGRAM OF PROPOSED FT-LFSR	40

4.2	EXTERNAL BLOCK DIAGRAM OF PROPOSED FT-LFSR DESIGNED IN XILINX	42
4.3	INTERNAL BLOCK DIAGRAM OF PROPOSED FT-LFSR DESIGNED IN XILINX	43
4.4	LOGIC DESIGN OF 2-INPUT XOR GATE	45
4.5	TRUTH TABLE OF 2-INPUT XOR GATE	45
4.6	LOGIC DESIGN OF 2-INPUT XNOR GATE	46
4.7	TRUTH TABLE OF 2-INPUT XNOR GATE	47
4.8	BLOCK DIAGRAM OF 4x1 MULTIPLEXER	48
4.9	TRUTH TABLE OF 4x1 MULTIPLEXER	48
4.10	LOGIC DESIGN OF 4x1 MULTIPLEXER	48
4.11	BLOCK DIAGRAM OF 2-BIT MAGNITUDE COMPARATOR	49
4.12	TRUTH TABLE OF 2-BIT MAGNITUDE COMPARATOR	50
4.13	BLOCK DIAGRAM OF D-FLIP FLOP	51
4.14	TRUTH TABLE OF D-FLIP FLOP	51
5.1	VERILOG SOURCE CODE OF EXISTING LFSR	54
5.2	SIMULATION OF EXISTING LFSR, WHEN RST=1	55
5.3	SIMULATION OF EXISTING LFSR, WHEN RST=0	55

5.4	INTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR	56
5.5	VERILOG SOURCE CODE OF 2-INPUT XOR GATE	57
5.6	SIMULATION OF 2-INPUT XOR GATE	57
5.7	VERILOG SOURCE CODE OF 2-INPUT XNOR GATE	58
5.8	SIMULATION OF 2-INPUT XNOR GATE	58
5.9	VERILOG SOURCE CODE OF 4x1 MULTIPLEXER	59
5.10	SIMULATION OF 4x1 MULTIPLEXER	59
5.11	VERILOG SOURCE CODE OF 2-BIT MAGNITUDE COMPARATOR	60
5.12	VERILOG SOURCE CODE OF 2-BIT MAGNITUDE COMPARATOR (CONT)	60
5.13	SIMULATION OF 2-BIT MAGNITUDE COMPARATOR	61
5.14	VERILOG SOURCE CODE OF D-FLIP FLOP	61
5.15	SIMULATION OF D-FLIP FLOP	62
5.16	EXTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR	62
5.17	VERILOG SOURCE CODE OF THE PROPOSED FT-LFSR	63
5.18	VERILOG SOURCE CODE OF THE PROPOSED FT-LFSR (CONT-1)	63

5.19	VERILOG SOURCE CODE OF THE PROPOSED FT-LFSR (CONT-2)	64
5.20	VERILOG SOURCE CODE OF THE PROPOSED FT-LFSR (CONT-3)	64
5.21	VERILOG SOURCE CODE OF THE PROPOSED FT-LFSR (CONT-4)	65
5.22	SIMULATION OF THE PROPOSED FT-LFSR, WHEN SEED DATA=1000	65
5.23	SIMULATION OF THE PROPOSED FT-LFSR, WHEN SEED DATA=1011	66
5.24	SIMULATION OF THE PROPOSED FT-LFSR, WHEN SEED DATA=1001	66
5.25	POWER ANALYSIS OF THE PROPOSED FT-LFSR	67
5.26	SETUP TO CLOCK TIMING OF THE PROPOSED FT-LFSR	68
5.27	CLOCK TO PAD TIMING OF THE PROPOSED FT-LFSR	69
5.28	CLOCK TO SETUP TIMING OF THE PROPOSED FT-LFSR	69
5.29	PAD TO PAD DELAY OF THE PROPOSED FT-LFSR	69

LIST OF TABLES

TABLE NO.	TITLE	PG NO.
1.1	PATTERN-GENERATOR SEED VECTORS	18
1.2	POLYNOMIALS FOR MAXIMAL LFSRs	22
5.1	CLOCK PERIOD OF THE PROPOSED AND THE EXISTING LFSR	67
5.2	AREA ANALYSIS OF THE PROPOSED FT-LFSR	68
5.3	COMPARISON OF THE EXISTING AND THE PROPOSED DESIGN	70

CHAPTER 1

INTRODUCTION

Very large-scale integration was made possible with the wide adoption of the MOS transistor, originally invented by Mohamed M. Atalla and Dawon Kahng at Bell Labs in 1959. Atalla first proposed the concept of the MOS integrated circuit chip in 1960, followed by Kahng in 1961, both noting that the MOS transistor's ease of fabrication made it useful for integrated circuits. General Microelectronics introduced the first commercial MOS integrated circuit in 1964. In the early 1970s, MOS integrated circuit technology allowed the integration of more than 10,000 transistors in a single chip. This paved the way for VLSI in the 1970s and 1980s, with tens of thousands of MOS transistors on a single chip (later hundreds of thousands, then millions, and now billions).

The main challenging areas in VLSI are performance, cost, testing, area, reliability and power. The demand for portable computing devices and communications system are increasing rapidly. These applications require low power dissipation for VLSI circuits. The power dissipation during test mode is 200% more than in normal mode. Hence it is important aspect to optimize power during testing. Power optimization is one of the main challenges.

The first semiconductor chips held two transistors each. Subsequent advances added more transistors, and as a consequence, more individual functions or systems were integrated over time. The first integrated circuits held only a few devices, perhaps as many as ten diodes, transistors, resistors and capacitors, making it possible to fabricate one or more logic gates on a single device. Now known retrospectively as small-scale integration (SSI), improvements in technique led to devices with hundreds of logic gates, known as medium-scale integration (MSI). Further improvements led to large-scale integration (LSI), i.e. systems with at least a thousand logic gates.

Current technology has moved far past this mark and today's microprocessors have many millions of gates and billions of individual transistors. At one time, there was an effort to name and calibrate various levels of large-scale integration above VLSI. Terms like ultra-large-scale integration (ULSI) were used.

But the huge number of gates and transistors available on common devices has rendered such fine distinctions moot. Terms suggesting greater than VLSI levels of integration are no longer in widespread use.

In 2008, billion-transistor processors became commercially available. This became more commonplace as semiconductor fabrication advanced from the then-current generation of 65 nm processes. Current designs, unlike the earliest devices, use extensive design automation and automated logic synthesis to lay out the transistors, enabling higher levels of complexity in the resulting logic functionality. Certain high-performance logic blocks like the SRAM (static random-access memory) cell, are still designed by hand to ensure the highest efficiency.

1.1 MOTIVATION

Linear Feedback Shift Register plays a very important role in many digital systems such as calculators, microprocessor, digital filters, digital signal processor. They are also used in implementation of various applications such as Built-In-Self-Test circuits, Pseudorandom Number Generators, Pattern Generators, Counters, Error Detection and correction, Computer graphics, cryptography etc. The performance of any digital system is usually measured in terms of the used Shift Registers. Today a combination of external Automated Test Equipment (ATE) and internal BIST (Built-In-Self-Test) techniques are used to ensure the highest possible fault coverage of the device at the lowest possible costs.

During testing large power is dissipated than in the normal mode. This is due to lack of correlation between the successive test patterns generated by ATPG (for external testing) or LFSR (for BIST). Motivation for the proposed methodology arises from two observations. First is that the literature available focuses upon ordering the test vectors after obtaining the test vector sequences from the LFSR. And the other is LFSR can itself be power efficient if modified for the purpose. The aim of this work is to design an LFSR which in itself is power efficient and the single point failures arising from it also been reduced significantly.

1.2 LINEAR FEEDBACK SHIFT REGISTER

A linear feedback shift register (LSFR) is a shift register that takes a linear function of a previous state as an input. Most commonly, this function is a Boolean exclusive OR (XOR). The bits that affect the state in the other bits are known as taps. Its a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

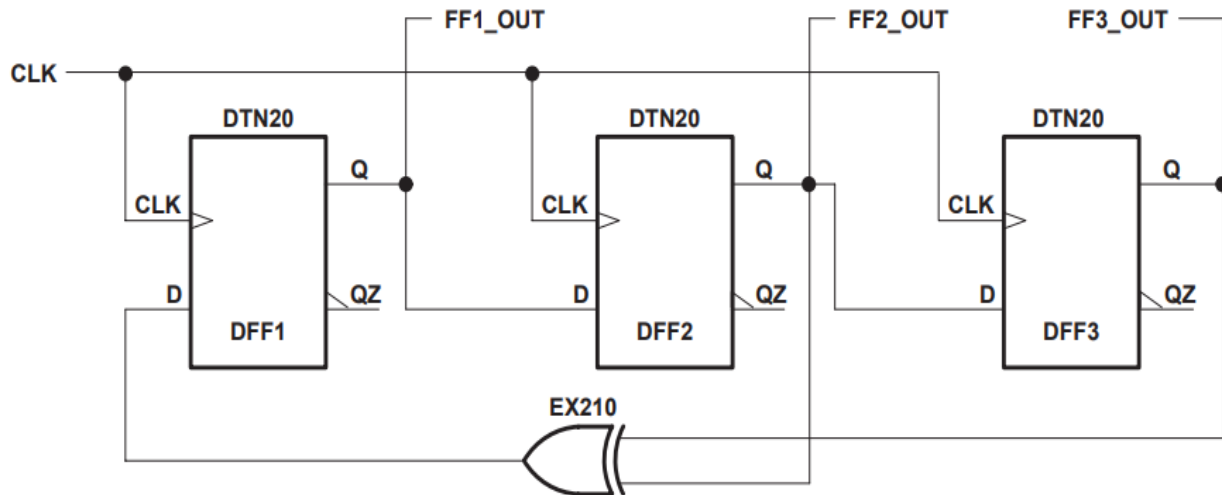


Fig:1.1 : Linear Feedback Shift Register

A linear feedback shift register takes a linear function, typically an exclusive OR, as an input. An LSFR, like other shift registers, is a cascade of flip-flop circuits. The bits that change state for the others in the cascade are called taps. Two of the major schemes for connecting taps are Fibonacci and Galois. In the Fibonacci configuration, the taps are cascaded and fed into the leftmost bit. In a Galois configuration, named after the French mathematician Évariste Galois, each tap is XOR'd to the output stream. A linear feedback shift register (LFSR) 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.

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream 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.

A n-bit LFSR is a n-bit length shift register with feedback to its input. The feedback is formed by XORing or XNORing the outputs of selected stages of the shift register - referred to as 'taps' - and then inputting this to the least significant bit (stage 0). Each stage has a common clock. The 'linear' part of the term 'LFSR' derives from the fact that XOR and XNOR are linear functions. An example of a 5-bit LFSR is shown below:

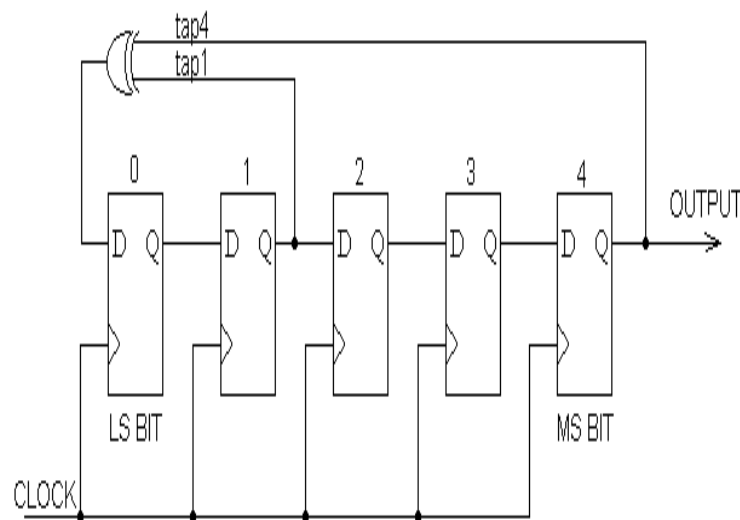


Fig:1.2 : 5-Bit Linear Feedback Shift Register

This has taps at stages 1 and 4 with XOR feedback. Here, the LS bit of the shift register is, by convention, shown at the left hand side of the shift register, with output being taken from the MS bit at the right hand side. It will produce a pseudorandom sequence of length 2^{n-1} states (where n is the number of stages) if the LFSR is of maximal length. The sequence will then repeat from the initial state for as long as the LFSR is clocked. LFSRs can be implemented in hardware, and this makes them useful in applications that require very fast generation of a pseudo-random sequence, such as direct-sequence spread spectrum radio.

1.2.1 Many-to-one implementation

LFSRs are simple to construct and are useful for a wide variety of applications, but are often sadly neglected by designers. One of the more common forms of LFSR is formed from a simple shift register with feedback from two or more points, or *taps*, in the register chain.

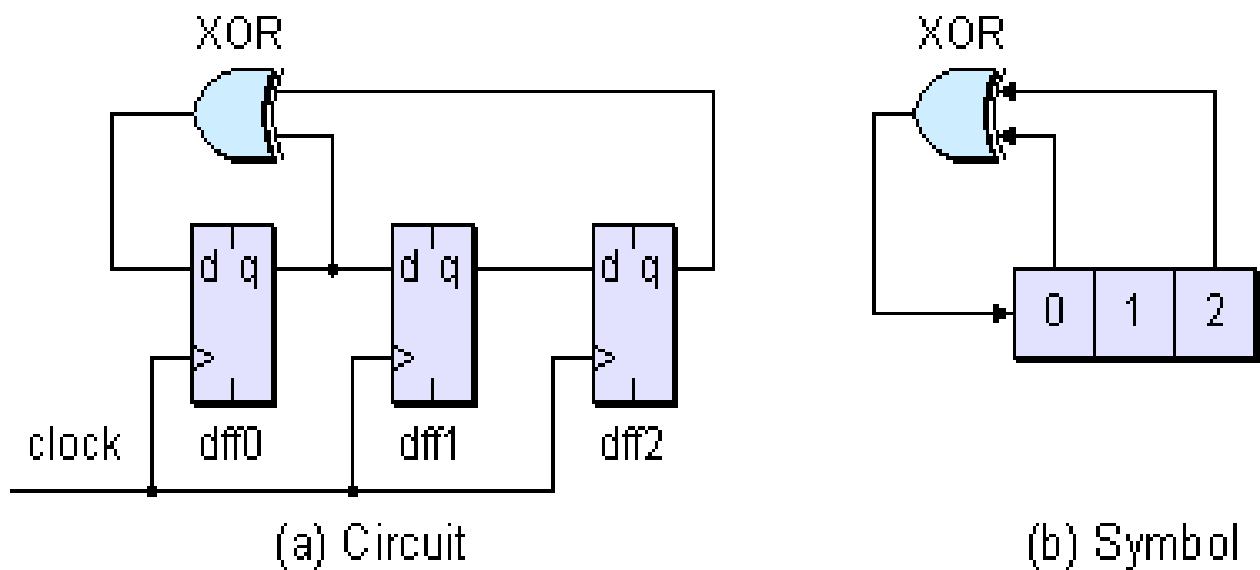


Fig:1.3 : LFSR with XOR Feedback path

The taps in this example are at bit 0 and bit 2, and can be referenced as [0,2]. All of the register elements share a common clock input, which is omitted from the symbol for reasons of clarity. The data input to the LFSR is generated by XOR-ing or XNOR-ing the tap bits; the remaining bits function as a standard shift register.

The sequence of values generated by an LFSR is determined by its feedback function (XOR versus XNOR) and tap selection. For example, consider two 3-bit XOR based LFSRs with different tap selections.

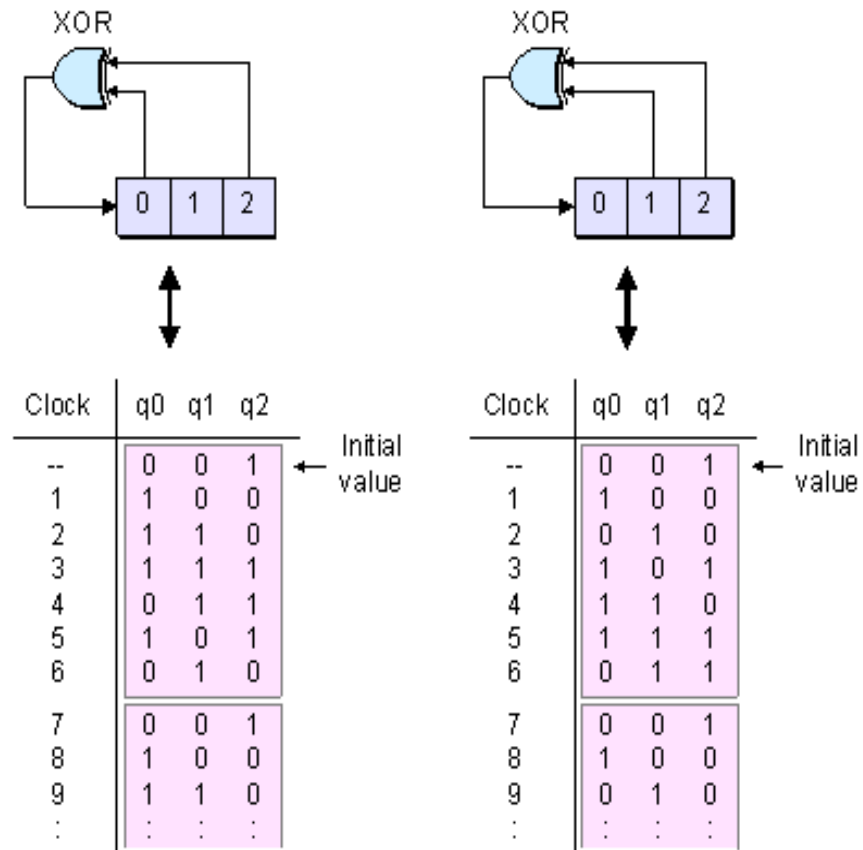


Fig:1.4 : Comparison of alternative tap selections

Both LFSRs start with the same initial value but, due to the different taps, their sequences rapidly diverge as clock pulses are applied. In some cases an LFSR will end up cycling round a loop comprising a limited number of values. However, both of the LFSRs are said to be of *maximal length* because they sequence through every possible value (excluding all of the bits being 0) before returning to their initial values.

A binary field with 'n' bits can assume 2^n unique values, but a maximal-length LFSR with 'n' register bits will only sequence through $(2^n - 1)$ values. This is because LFSRs with XOR feedback paths will not sequence through the value where all the bits are 0, while their XNOR equivalents will not sequence through the value where all the bits are 1.

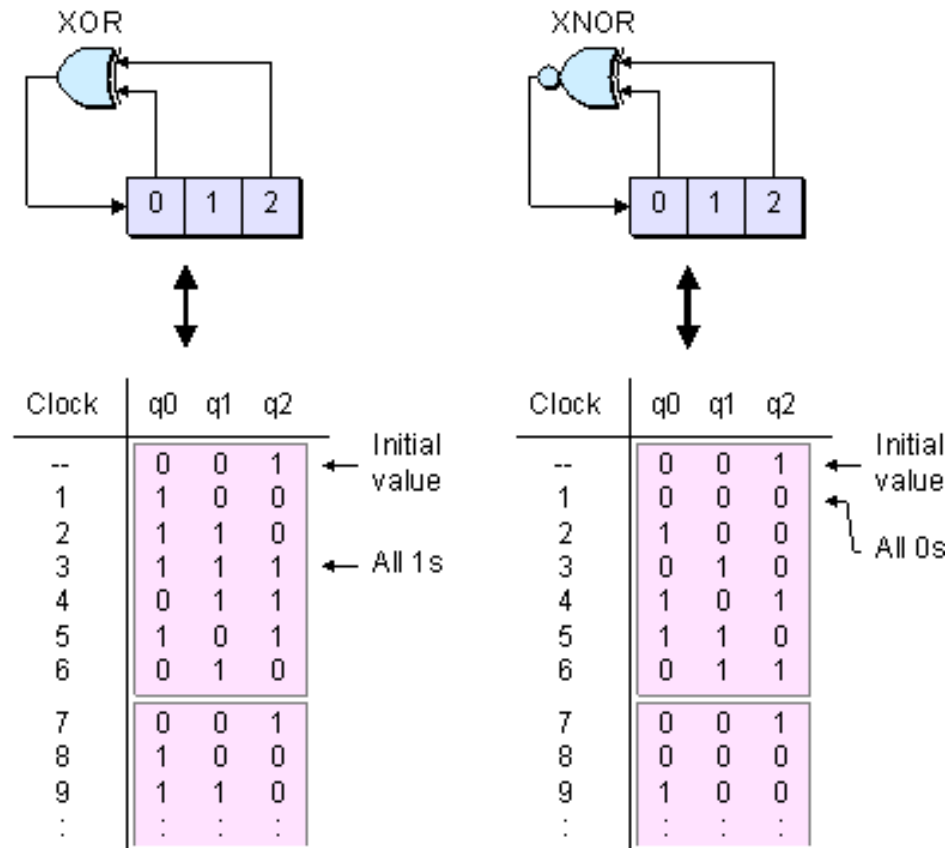


Fig:1.5 : Comparison of XOR versus XNOR feedback paths.

Each LFSR supports a number of tap combinations that will generate maximal-length sequences. The problem is weeding out the ones that do from the ones that don't, because badly chosen taps can result in the register entering a loop comprising only a limited number of states.

The taps are identical for both XOR-based and XNOR-based LFSRs, although the resulting sequence will of course differ. As was previously noted, alternative tap combinations may also yield maximum-length LFSRs, although the resulting sequences will vary.

For example, in the case of a 10-bit LFSR, there are two 2-tap combinations that result in a maximal-length sequence: [2,9] and [6,9]. There are also twenty 4-tap combinations, twenty-eight 6-tap combinations, and ten 8-tap combinations that satisfy the maximal-length criteria.

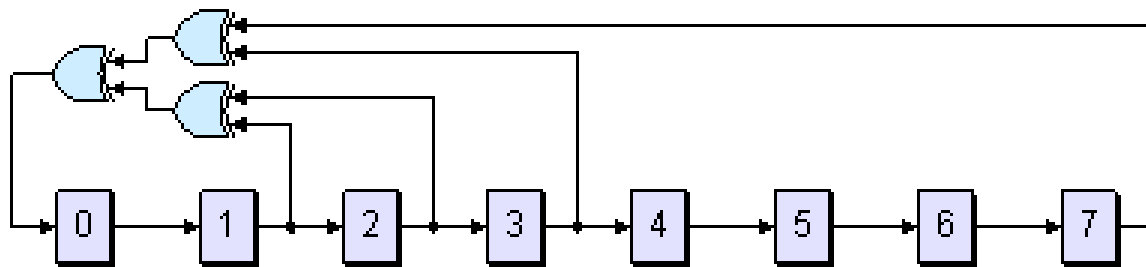
# of Bits	Length of Loop	Taps
2	3 *	[0,1]
3	7 *	[0,2]
4	15	[0,3]
5	31 *	[1,4]
6	63	[0,5]
7	127 *	[0,6]
8	255	[1,2,3,7]
9	511	[3,8]
10	1,023	[2,9]
11	2,047	[1,10]
12	4,095	[0,3,5,11]
13	8,191 *	[0,2,3,12]
14	16,383	[0,2,4,13]
15	32,767	[0,14]
16	65,535	[1,2,4,15]
17	131,071 *	[2,16]
18	262,143	[6,17]
19	524,287 *	[0,1,4,18]
20	1,048,575	[2,19]
21	2,097,151	[1,20]
22	4,194,303	[0,21]
23	8,388,607	[4,22]
24	16,777,215	[0,2,3,23]
25	33,554,431	[2,24]
26	67,108,863	[0,1,5,25]
27	134,217,727	[0,1,4,26]
28	268,435,455	[2,27]
29	536,870,911	[1,28]
30	1,073,741,823	[0,3,5,29]
31	2,147,483,647 *	[2,30]
32	4,294,967,295	[1,5,6,31]

Fig:1.6 : Taps for maximal length LFSRs with 2 to 32 bits.

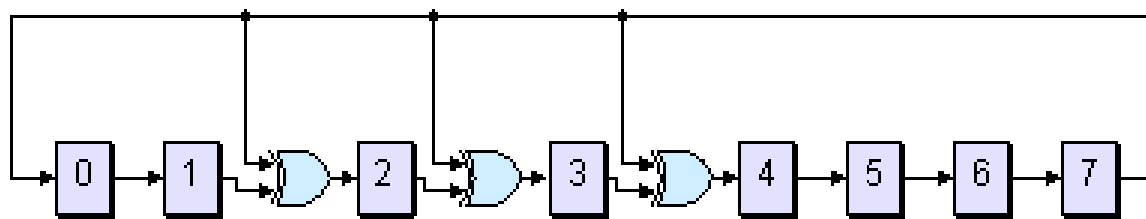
1.2.2 One-to-many versus Many-to-one implementation

Consider the case of an 8-bit LFSR, for which the minimum number of taps that will generate a maximal-length sequence is four. In the real world, XOR gates only have two inputs, so a four-input XOR function has to be created using three XOR gates arranged as two levels of logic. Even in those cases where an LFSR does support a minimum of two taps, you may actually wish to use a greater number of taps such as eight (which would result in three levels of XOR logic).

The problem is that increasing the levels of logic in the combinational feedback path can negatively impact the maximum clocking frequency of the function. One solution is to transpose the *many-to-one implementations* discussed above into their *one-to-many counterparts*.



(a) Many-to-one implementation



(b) One-to-many implementation

Fig:1.7 : One-to-many versus many-to-one implementations

The traditional many-to-one implementation for the eight-bit LFSR has taps at [7,3,2,1]. To convert this into its one-to-many counterpart, the most-significant tap (which is always the most significant bit) is fed back directly into the least significant bit, and is also individually XORED with the other original taps (bits [3,2,1] in this example).

Note that although both styles result in maximal-length LFSRs, the actual sequences of values will differ between them. But the main point is that using the one-to-many style means that there is never more than one level of combinational logic in the feedback path, irrespective of the number of taps being employed.

1.3 FIBONACCI LINEAR FEEDBACK SHIFT REGISTER

LFSR is a state machine composed of a shift register whose input bit is a linear function of its previous state. In the Fibonacci implementation, the outputs from some of the registers are XORED with each other and fed back to the input of the shift register. There are two types of Fibonacci LFSR: 4-Bit Fibonacci LFSR and 16-Bit Fibonacci LFSR.

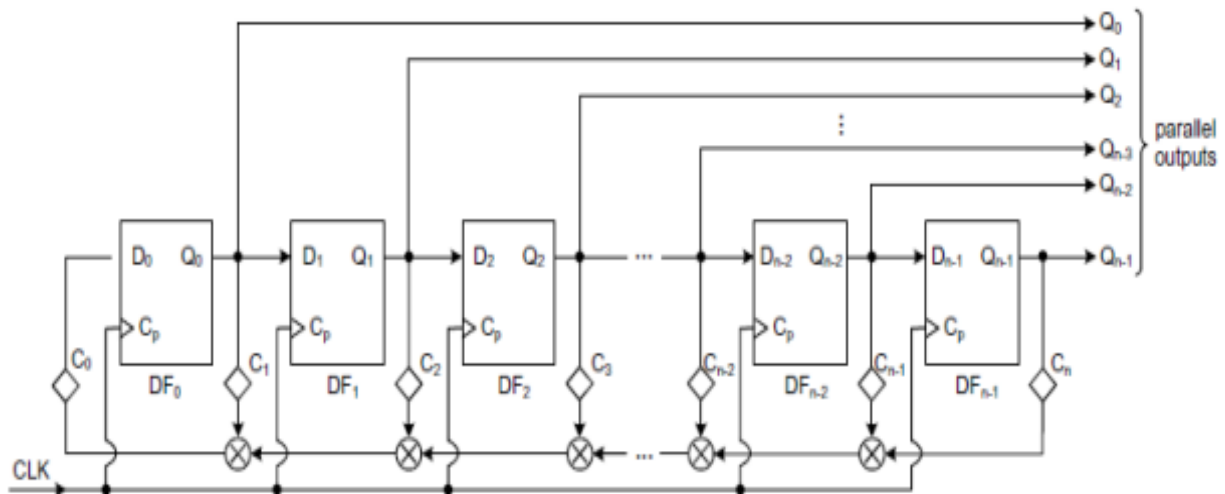


Fig:1.8 : Fibonacci Linear Feedback Shift Register

1.3.1 4-Bit Fibonacci Linear Feedback Shift Register

A 4-bit LFSR with its state diagram. The XOR gate provides feedback to the register that shifts bits from left to right. The maximal sequence consists of every possible state except the "0000" state. A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is XOR.

Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, are closely related to those of an LFSR.

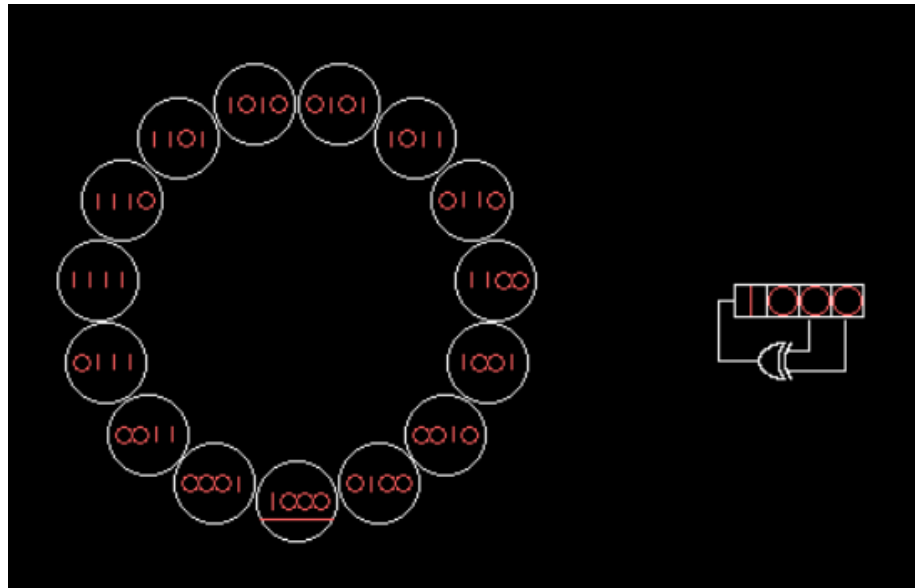


Fig:1.9 : 4-Bit Linear Feedback Shift Register with its state diagram

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream 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. The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, are closely related to those of an LFSR.

1.3.2 16-Bit Fibonacci Linear Feedback Shift Register

A 16-bit LFSR. The feedback tap numbers in white correspond to a primitive polynomial in the table so the register cycles through the maximum number of 65535 states excluding the all-zeroes state. The state shown, 0xACE1 (hexadecimal) will be followed by 0x5670. As well as Fibonacci, this LFSR configuration is also known as standard, many-to-one or external XOR gates. LFSR has an alternative configuration.

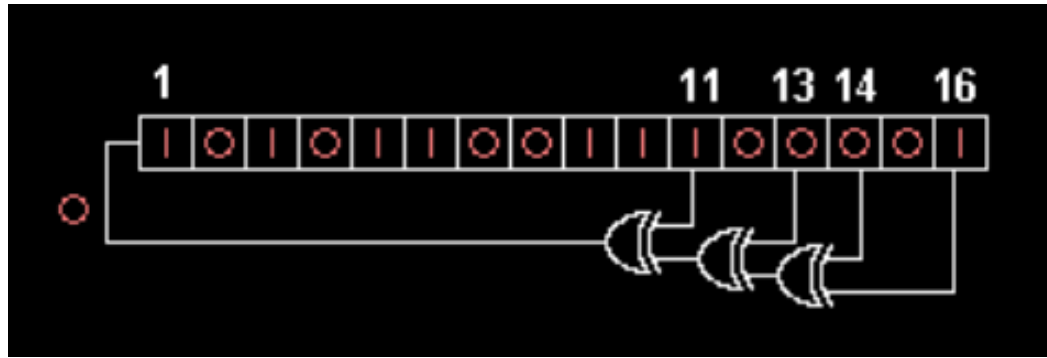


Fig:1.10 : 16-Bit Linear Feedback Shift Register with its state diagram

The bit positions that affect the next state are called the taps. In the above diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.

- The bits in the LFSR state which influence the input are called taps (white in the diagram).
- A maximum-length LFSR produces an m-sequence (i.e. it cycles through all possible $2^n - 1$ states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.
- As an alternative to the XOR based feedback in an LFSR, one can also use XNOR. This function is an affine map, not strictly a linear map, but it results in an equivalent polynomial counter whose state of this counter is the complement of the state of an LFSR. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state.

The sequence of numbers generated by an LFSR or its XNOR counterpart can be considered a binary numeral system just as valid as Gray code or the natural binary code. 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 characteristic polynomial.

For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is,

$$x^{16}+x^{14}+x^{13}+x^{11}+1 \quad (1.1)$$

The 'one' in the polynomial does not correspond to a tap — it corresponds to the input to the first bit (i.e. x^0 , which is equivalent to 1). The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and output tap respectively.

- The LFSR will only be maximum-length if the number of taps is even; just 2 or 4 taps can suffice even for extremely long sequences.
- The set of taps — taken all together, not pairwise (i.e. as pairs of elements) — must be relatively prime. In other words, there must be no common divisor to all taps.
- Once one maximum-length tap sequence has been found, another automatically follows. If the tap sequence, in an n -bit LFSR, is $[n, A, B, C, 0]$, where the 0 corresponds to the $x^0 = 1$ term, then the corresponding 'mirror' sequence is $[n, n - C, n - B, n - A, 0]$. So the tap sequence $[32, 7, 3, 2, 0]$ has as its counterpart $[32, 30, 29, 25, 0]$. Both give a maximum-length sequence.

Example code for 16-Bit LFSR,

```
#include <stdint.h>
uint16_t lfsr = 0xACE1u;
unsigned bit;
unsigned period = 0;
do {
/* taps: 16 14 13 11; characteristic polynomial:  $x^{16} + x^{14} + x^{13} + x^{11} + 1$  */
bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ) & 1;
lfsr = (lfsr >> 1) | (bit << 15);
++period;
} while(lfsr != 0xACE1u);
```

The above code assumes the most significant bit of (lfsr) is bit 1, and the least significant bit is bit 16.

1.4 GALOIS LINEAR FEEDBACK SHIFT REGISTER

Galois LFSR is named after the French mathematician Évariste Galois, an LFSR in Galois configuration, which is also known as modular, internal XORs as well as one-to-many LFSR, is an alternate structure that can generate the same output stream as a conventional LFSR. In the Galois configuration, when the system is clocked, bits that are not taps are shifted one position to the right unchanged. The taps are XOR'd with the output bit before they are stored in the next position. The new output bit is the next input bit. The effect of this is that when the output bit is zero all the bits in the register shift to the right unchanged, and the input bit becomes zero. When the output bit is one, the bits in the tap positions all flip (if they are 0, they become 1, and if they are 1, they become 0), then the entire register is shifted to right and the input bit becomes 1.

The taps, on the other hand, are XOR'd with the output bit before they are stored in the next position. The new output bit is the next input bit. The effect of this is that when the output bit is zero all the bits in the register shift to the right unchanged, and the input bit becomes zero. When the output bit is one, the bits in the tap positions all flip (if they are 0, they become 1, and if they are 1, they become 0), then the entire register is shifted to right and the input bit becomes 1.

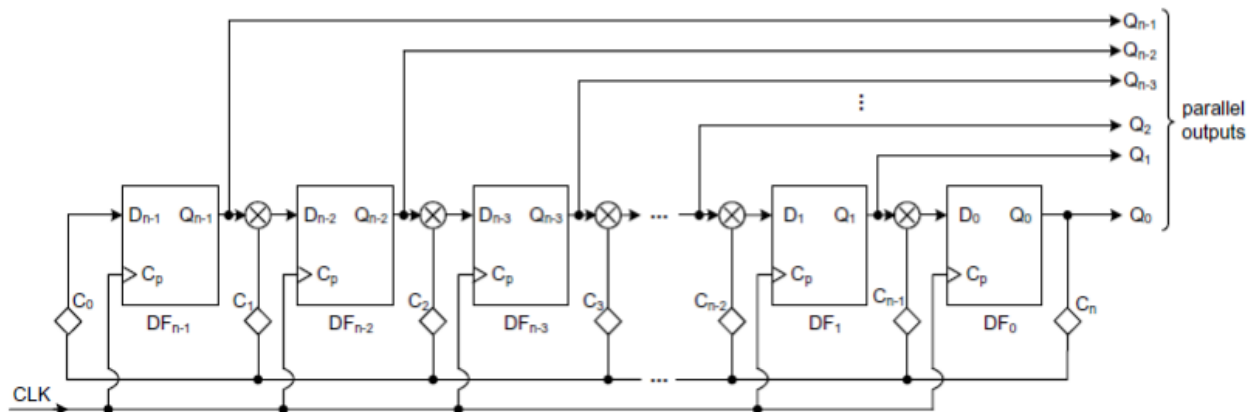


Fig:1.11 : Galois Linear Feedback Shift Register

1.4.1 16-Bit Galois Linear Feedback Shift Register

A 16-bit Galois LFSR. The register numbers in white correspond to the same primitive polynomial as the Fibonacci example but are counted in reverse to the shifting direction. This register also cycles through the maximal number of 65535 states excluding the all-zeroes state. The state ACE1 hex will be followed by E270 hex.

To generate the same output stream, the order of the taps is the counterpart (see above) of the order for the conventional LFSR, otherwise the stream will be in reverse. Note that the internal state of the LFSR is not necessarily the same. The Galois register has the same output stream as the Fibonacci register in the first section.

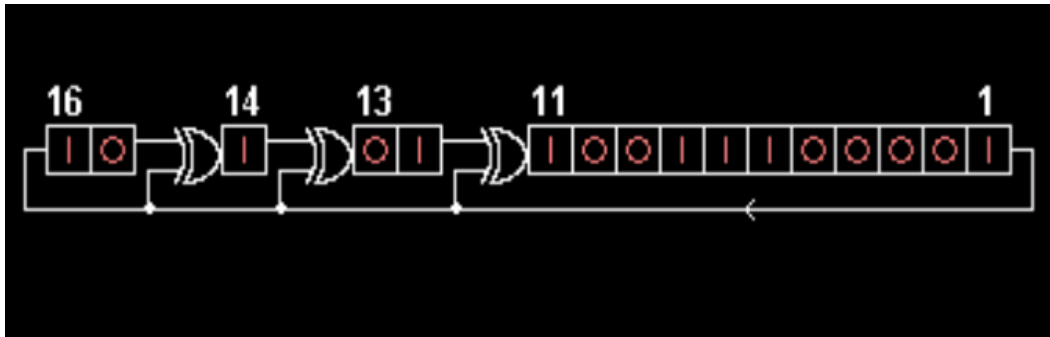


Fig:1.12 : 16- Bit Galois Linear Feedback Shift Register

- Galois LFSRs do not concatenate every tap to produce the new input (the XOR'ing is done within the LFSR and no XOR gates are run in serial, therefore the propagation times are reduced to that of one XOR rather than a whole chain), thus it is possible for each tap to be computed in parallel, increasing the speed of execution.
- In a software implementation of an LFSR, the Galois form is more efficient as the XOR operations can be implemented a word at a time: only the output bit must be examined individually.

Below is a code example of a 32-bit maximal period Galois LFSR,

```
#include <stdint.h>

uint32_t lfsr = 1;

unsigned period = 0;

do {
/* taps: 32 31 29 1; characteristic polynomial:  $x^{32} + x^{31} + x^{29} + x + 1$  */
lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xD0000001u);
++period;
} while(lfsr != 1u);
```

And here is the code for the 16 bit example,

```
#include <stdint.h>
uint16_t lfsr = 0xACE1u;
5 unsigned period = 0;
do {
/* taps: 16 14 13 11; characteristic polynomial:  $x^{16} + x^{14} + x^{13} + x^{11} + 1$  */
lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xB400u);
++period;
} while(lfsr != 0xACE1u);
```

These code examples create a toggle mask to apply to the shifted value using the XOR operator. The mask is created by first removing all but the least significant bit (the output bit) of the current value. This value is then negated (two's complement negation), which creates a value of either all 0s or all 1s, if the output bit is 0 or 1, respectively. By ANDing the result with the tapvalue (e.g., 0xB400 in the second example) before applying it as the toggle mask, it acts functionally as a conditional to either apply or not apply the toggle mask based on the output bit. A more explicit but significantly less efficient code example is shown below,

```
#include <stdint.h>
uint16_t lfsr = 0xACE1u;
unsigned period = 0;
do {
    unsigned lsb = lfsr & 1;    /* Get lsb (i.e., the output bit). */
    lfsr >>= 1;                /* Shift register */
    if (lsb == 1)               /* Only apply toggle mask if output bit is 1. */
        lfsr ^= 0xB400u;       /* Apply toggle mask, value has 1 at bits corresponding *
                                to taps, 0 elsewhere. */
    ++period;
} while(lfsr != 0xACE1u);
```

1.4.2 Non-Binary Galois Linear Feedback Shift Register

Binary Galois LFSRs like the ones shown above can be generalized to any q -ary alphabet $\{0, 1, \dots, q - 1\}$ (e.g., for binary, q is equal to two, and the alphabet is simply $\{0, 1\}$). In this case, the exclusive-or component is generalized to addition modulo- q (note that XOR is addition modulo 2), and the feedback bit (output bit) is multiplied (modulo- q) by a q -ary value which is constant for each specific tap point.

Note that this is also a generalization of the binary case, where the feedback is multiplied by either 0 (no feedback, i.e., no tap) or 1 (feedback is present). Given an appropriate tap configuration, such LFSRs can be used to generate Galois fields for arbitrary values of q .

1.5 MAXIMAL LENGTH LINEAR FEEDBACK SHIFT REGISTER

A maximal-length LFSR produces the maximum number of PRPG patterns possible and has a pattern count equal to $2^n - 1$, where n is the number of register elements in the LFSR. It produces patterns that have an approximately equal number of 1s and 0s and have an equal number of runs of 1s and 0s.

Because there is no way to predict mathematically if an LFSR will be maximal length, Peterson and Weldon² have compiled tables of maximal-length LFSRs to which designers may refer. Table 1.1 shows the patterns produced by the LFSR, assuming that a pattern of 111 was used as a seed.

In a practical ASIC design, the user would create an LFSR that is much bigger than three bits to get a large number of pseudorandom patterns before the patterns repeated. However, there are some practical restrictions to the length of the LFSR. A 32-bit maximal-length LFSR would create over 4 billion patterns that, at a 16-MHz clock rate, would take almost 5 minutes to generate the whole pattern set.

Table 1.1 Pattern-Generator Seed Values

CLOCK PULSE	FF1_OUT	FF2_OUT	FF3_OUT	COMMENTS
1	1	1	1	Seed Value
2	0	1	1	
3	0	0	1	
4	1	0	0	
5	0	1	0	
6	1	0	1	
7	1	1	0	
8	1	1	1	Starts repeat

Maximal length Linear feedback shift registers make extremely good pseudorandom pattern generators. When the outputs of the flip-flops are loaded with a seed value (anything except all 0s, which would cause the LFSR to produce all 0 patterns) and when the LFSR is clocked, it will generate a pseudorandom pattern of 1s and 0s.

1.6 XOR SHIFTS LINEAR FEEDBACK SHIFT REGISTER

Xorshift random number generators, also called shift-register generators, are a class of pseudorandom number generators that were discovered by George Marsaglia. They are a subset of linear-feedback shift registers (LFSRs) which allow a particularly efficient implementation in software without using excessively sparse polynomials. They generate the next number in their sequence by repeatedly taking the exclusive or of a number with a bit-shifted version of itself. This makes them execute extremely efficiently on modern computer architectures, but does not benefit efficiency in a hardware implementation. Like all LFSRs, the parameters have to be chosen very carefully in order to achieve a long period.

For execution in software, xorshift generators are among the fastest non-cryptographically-secure random number generators, requiring very small code and state. However, they do not pass every statistical test without further refinement. This weakness is well-known and is amended (as pointed out by Marsaglia in the original paper) by combining them with a non-linear function, resulting e.g. in a xorshift+ or xorshift* generator. A native C implementation of a xorshift+ generator that passes all tests from the BigCrush suite (with an order of magnitude fewer failures than Mersenne Twister or WELL) typically takes fewer than 10 clock cycles on x86 to generate a random number, thanks to instruction pipelining.

The scramblers known as + and * still leave weakness in the low bits, so they're intended for floating point use, as conversion of a random number to floating point discards the low bits. For general purpose, the scrambler ** (pronounced 'starstar') makes the LFSR generators pass in all bits. Because plain xorshift generators (without a non-linear step) fail some statistical tests, they have been accused of being unreliable.

As shown by George Marsaglia and further analysed by Richard P. Brent, linear feedback shift registers can be implemented using XOR and Shift operations. This approach lends itself to fast execution in software because these operations typically map efficiently into modern processor instructions.

Below is a code example for a 16-bit maximal-period Xorshift LFSR using the 7,9,13 triplet from John Metcalf:

```
#include<stdint.h>
unsigned lfsr_xorshift(void)
{
    unit16_t start_state = 0xACE1u;
    unit16_t lfsr = start_state;
    unsigned period = 0;
    do
    {
        lfsr ^= lfsr >> 7;
        lfsr ^= lfsr << 9;
        lfsr ^= lfsr >> 13;
```

```

++period;
}
while (lfsr != start_state);
return period;
}

```

1.7 MATRIX FORMS OF LFSR

Binary LFSRs of both Fibonacci and Galois configurations can be expressed as linear functions using matrices. Using the companion matrix of the characteristic polynomial of the LFSR and denoting the seed as a column vector $(a_0, a_1, \dots, a_{n-1})^T$, the state of the register in Fibonacci configuration after k steps is given by,

$$\begin{pmatrix} a_k \\ a_{k+1} \\ a_{k+2} \\ \vdots \\ a_{k+n-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \\ c_0 & c_1 & \dots & \dots & c_{n-1} \end{pmatrix} \begin{pmatrix} a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ a_{k+n-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \\ c_0 & c_1 & \dots & \dots & c_{n-1} \end{pmatrix}^k \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Matrix for the corresponding Galois form is :

$$\begin{pmatrix} c_0 & 1 & 0 & \dots & 0 \\ c_1 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c_{n-2} & 0 & \dots & 0 & 1 \\ c_{n-1} & 0 & \dots & \dots & 0 \end{pmatrix}$$

For a suitable initialisation,

$$a'_i = \sum_{j=0}^i a_{i-j} c_{n-j}, \quad 0 \leq i < n$$

the top coefficient of the column vector :

$$\begin{pmatrix} c_0 & 1 & 0 & \cdots & 0 \\ c_1 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c_{n-2} & 0 & \cdots & 0 & 1 \\ c_{n-1} & 0 & \cdots & \cdots & 0 \end{pmatrix}^k \begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ \vdots \\ a'_{n-1} \end{pmatrix}$$

gives the term a_k of the original sequence.

These forms generalize naturally to arbitrary fields.

1.8 OUTPUT STREAM PROPERTIES OF LFSR

- 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 half of these runs will be one bit long, a quarter 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 distribution almost equals the statistical expectation value for a truly random sequence. However, the probability of finding exactly this distribution in a sample of a truly random sequence is rather low.
- LFSR output streams are deterministic. If you know the present state, you can predict the next state. This is not possible with truly random events.
- The output stream is reversible; an LFSR with mirrored taps will cycle through the output sequence in reverse order.

1.9 POLYNOMIALS FOR MAXIMAL LFSRs

The following table lists examples of maximal-length feedback polynomials (primitive polynomials) for shift-register lengths up to 24. The formalism for maximum-length LFSRs was developed by Solomon W. Golomb in his 1967 book. The number of different primitive polynomials grows exponentially with shift-register length and can be calculated exactly using Euler's totient function.

Table 1.2 Polynomials for maximal LFSRs

Bits (n)	Feedback polynomial	Taps	Taps (hex)	Period ($2^n - 1$)
2	$x^2 + x + 1$	11	0x3	3
3	$x^3 + x^2 + 1$	110	0x6	7
4	$x^4 + x^3 + 1$	1100	0xC	15
5	$x^5 + x^3 + 1$	10100	0x14	31
6	$x^6 + x^5 + 1$	110000	0x30	63
7	$x^7 + x^6 + 1$	1100000	0x60	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	10111000	0xB8	255
9	$x^9 + x^5 + 1$	100010000	0x110	511
10	$x^{10} + x^7 + 1$	1001000000	0x240	1,023
11	$x^{11} + x^9 + 1$	10100000000	0x500	2,047
12	$x^{12} + x^{11} + x^{10} + x^4 + 1$	111000001000	0xE08	4,095
13	$x^{13} + x^{12} + x^{11} + x^8 + 1$	1110010000000	0x1C80	8,191
14	$x^{14} + x^{13} + x^{12} + x^2 + 1$	11100000000010	0x3802	16,383
15	$x^{15} + x^{14} + 1$	110000000000000	0x6000	32,767
16	$x^{16} + x^{15} + x^{13} + x^4 + 1$	1101000000001000	0xD008	65,535
17	$x^{17} + x^{14} + 1$	10010000000000000	0x12000	131,071
18	$x^{18} + x^{11} + 1$	100000010000000000	0x20400	262,143
19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	1110010000000000000	0x72000	524,287
20	$x^{20} + x^{17} + 1$	10010000000000000000	0x90000	1,048,575
21	$x^{21} + x^{19} + 1$	101000000000000000000	0x140000	2,097,151
22	$x^{22} + x^{21} + 1$	1100000000000000000000	0x300000	4,194,303
23	$x^{23} + x^{18} + 1$	10000100000000000000000	0x420000	8,388,607
24	$x^{24} + x^{23} + x^{22} + x^{17} + 1$	111000010000000000000000	0xE10000	16,777,215

Note that more than one maximal-length polynomial may exist for any given shift-register length.

1.10 FACTORS AFFECTING COST OF CHIP AND ABOUT TESTING FOR POWER DISSIPATION

There are various factors that affect the cost of chip like packaging, application, testing etc. In VLSI, according to thumb rule 50% of the total integrated circuits cost is due to testing. During testing two key challenges are:

- Cost of testing that can't be scaled.
- Engineering effort for generating test vectors increases as complexity of circuit increases.

One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are things the tester tries to do with the product, and the product answers with its behavior in reaction to the probing of the tester. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its spaces.

There are main two sources of power dissipation in digital circuits; these are static and dynamic power dissipation. Static power dissipation is mainly due to leakage current and its contribution to total power dissipation is very small. Dynamic power dissipation is due to switching i.e. the power consumed due to short circuit current flow and charging of load capacitances. Power dissipation during the testing is one of most important issue. There are several reasons for this power increased in test mode.

To test large circuit, circuits are partitioned to save the test time but this parallel testing results in excessive energy and power dissipation. Due to the lack of at-speed equipment availability, delay is introduced in the circuit during testing. This causes power dissipation. In the successive functional input vectors applied to a given circuit in normal mode have a significant correlation, while the correlation between consecutive test patterns can be very low. This can cause large switching activity in the circuit during test than that during its normal operation. Power dissipation in CMOS circuits is proportional to switching activity, this excessive switching activity during test may be responsible for cost, reliability, performance verification, autonomy and technology related problems.

Generally total power consumption of the device is a sum of the power consumed by the core and I/O's. Appropriate package attributes such as the material and thermal properties need to be selected to ensure maximum heat dissipation of the die through the package.

IC Architects normally have a specific power consumption budget for the SOC based on the overall system level power budget. Functional operation of the device in a system usually consumes power either at or under the budgeted power for the IC. However the same device under manufacturing test especially with random pattern generator can consume more power than the budgeted power.

This increase in the power consumption of the IC in the test mode is well known in the industry to cause sudden un-repairable device failures resulting in significant manufacturing fall-out directly impacting the cost of the IC.

1.11 EFFECTS OF LARGE POWER DISSIPATION

The increased power may be responsible for cost, reliability, performance verification, autonomy and technology related problems. Low power dissipation during test application is thus becoming an equally important figure of merit in today's VLSI circuits design and is expected to become one of the major objectives in the near future.

High power and ground noise caused by high switching during testing are serious problem where the supply connects are poor. Thus excessive noise can change the logic state of the circuit lines leading good dies to fail the test and hence loss of yield. Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. A secondary benefit of testing is that it demonstrates that the software appears to be working as stated in the specifications. The data collected through testing can also provide an indication of the software's reliability and quality.

The test vectors applied to a circuit under test to the nominal operating frequency often cause more average and / or maximum power dissipation in normal mode. The reason is that the randomness of patterns generated by the LFSR, the correlation between the reduction and pseudo-random patterns in each model as well. This, in turn, can lead to more power dissipation and switching in test mode.

1.12 SOC FOR POWER CONSUMPTION

Estimating the IC power budget involves maintaining perspective of several technical and business factors such as process technology, cell design libraries, pin-out and package constraints, logic and IP macro clock domains, cost and time pressures, etc.

IC architects are required to select the optimum combination of these elements to be able to achieve not only a functional device but also cost competitive and meeting customer schedule. Total power consumption of the CMOS device is the sum of static power, dynamic power, leakage and short circuit¹¹. Static power can be defined as the power consumed by the CMOS gate under no switching activity. It is normally attributed to the leakage currents in the device.

Dynamic power on the other hand is a direct result of the switching activity of the gate and is generally most of the power consumed by the device. The more the gates toggle or change states under various load conditions, the more dynamic power is consumed. Short circuit power of the cell is caused by the temporary short circuit between the N and P transistors of the gate during logic transitions.

1.13 OBJECTIVES OF THE PROJECT

- To design a new fault tolerant architecture for LFSR (named as FT-LFSR) in which the number of SPoFs are significantly reduced.
- To fabricate a modified version of Triple Modular Redundancy (TMR) empowered with some extra controlling units for identifying the operational module.
- To introduce a novel metric called Reliability-Area-Factor (RAF) to evaluate the efficacy of the redundancy-based fault tolerant techniques such as FT-LFSR.
- To compare the fault tolerant linear feedback shift register in terms of power, delay and power delay product.
- To identify the feasibility of FT-LFSR for VLSI Applications.

1.14 ORGANIZATION OF THE REPORT

- **CHAPTER 1** discusses about VLSI, Motivation, LFSRs, Various types of LFSRs, XOR shifts, Matrix forms ,output streaming properties, polynomials of maximal LFSRs. It also discusses about the various factors affecting the cost of the chip, power dissipation and power consumption, objectives of the project and organization of the project work.
- **CHAPTER 2** delivers the detailed literature survey related to the project, literature summary and problem definition.
- **CHAPTER 3** describes the working and existing techniques of Fault tolerant Linear Feedback Shift Register.
- **CHAPTER 4** discusses the proposed Fault Tolerant Linear Feedback Shift register, simulation XILINX and MODELSIM.
- **CHAPTER 5** provides the simulation results and performance analysis of the existing and proposed project work. It does a full comparative analysis on how the proposed work is better off than the existing design.
- **CHAPTER 6** provides the conclusion of the project work. It also gives the schema of the future work that is how the project could be enhanced further. Finally, the references referred to lead the project are given in brief.

CHAPTER 2

LITERATURE SURVEY

This chapter discusses about the literature survey and literature summary. It also give idea about analysis of an efficient fault tolerant linear feedback shift registers for vlsi applications.

2.1 LITERATURE SURVEY

Area-Efficient Error Detection Structure for Linear Feedback Shift Registers
Hwasoo Shin, Soyeon Choi, Jiwoon Park, Byeong Yong Kong and Hoyoung Yoo
2020

To propose a new error detection LFSR, which can detect errors with a small hardware overhead by employing the parity check technique. The linear feedback shift register (LFSR) is one of the most widely used sequential logics, which produces the linear function of its previous state. The proposed error detection LFSR (ED-LFSR) estimates the parity of the next state by using the linear operation of the LFSRs. At the next clock cycle, the actual parity is generated and compared with the estimated one to detect the occurrence of errors. The parity function is applied to the conventional LFSR and later simplified using the linear property of the parity function. The structure is carefully designed to reduce hardware complexity by sharing a part of the circuitries. The experimental results indicate that the proposed ED-LFSR always maintains the smallest hardware complexity among all the error detection LFSR designs. Thus, the method can provide an area-efficient solution to incorporate error detection technique in conventional LFSRs.

The experimental results indicate that the proposed ED-LFSR requires an additional area of only 31.1% compared to that required by the conventional LFSR and it saves 39.1% and 31.9% of the resources compared to the corresponding utilization of the hardware and time redundancy methods. The experimental results indicate that the proposed ED-LFSR always maintains the smallest hardware complexity among all the error detection LFSR designs, regardless of the employed generator polynomials. The further work includes the complete error detection and the error-correction methods for LFSRs since the proposed method can support only the odd numbers of error detection due to the property of the parity check.

LFSR generation for high test coverage and low hardware overhead
Leonel Hernández Martínez , Saqib Khursheed , Sudhakar Mannapuram Reddy,
2019

To analyze the LFSR, using a automatic test PG (ATPG) test set by concatenating each test vector with the immediate next sequential pattern of the set, which generates a concatenated pattern. The proposed technique prioritises high fault coverage through a novel approach removing storage of patterns and improves the calculation process of LFSR. The proposed technique forms one concatenated test pattern for the input of the BM algorithm and this outputs the polynomial expression that fully maps the concatenated set. The bit generation by LFSR is based on the recurrence relation model, where each consecutive bit results from a linear combination between previous patterns and polynomial coefficients. The linear combination among coefficients of the polynomial representation of LFSR and each bit position of the target pattern is utilised by BM algorithm for the assignment of unknown bits from relaxed test patterns. The concatenated patterns are taken as the new goal, and the next step is the calculation of LFSR that will map the full bit pattern. In the procedure of calculating the LFSR, this is represented as polynomial; therefore, this proposition proceeds by finding the polynomial representation and then use the mathematical expression to generate the test pattern.

Results are generated using 32 benchmarks and 65 nm technology. The concatenated technique provides reductions on area overhead for 90.6% cases with a best case of 57 and 39% means. The remaining 9.4% of cases, non-concatenated technique provides the best reduction of 37 with 1.4% means, whilst achieving 100% test mapping in both cases. Results are provided on the technique for concatenated and non-concatenated ATPG set for LFSR. Each technique is formed by the necessary logic to perform the mapping of bit patterns. In the case of concatenated patterns, only the LFSR is required, whereas in the non-concatenated technique the LFSR and all seeds per benchmark are necessary. Safety-critical technology rests on optimised and effective testing techniques for every embedded system involved in the equipment. Pattern generator (PG) such as linear feedback shift register (LFSR) is used for fault detection and useful for reliability and online test.

Concurrent generation of pseudo random numbers with LFSR of Fibonacci and Galois type

Emina Milovanovic, Mile Stojcev, Igor Milovanovic, Tatjana Nikoli and Zoran Stamenkovic

2015

To propose a design which can drive several circuits under test (CUT) simultaneously and to describe the mathematical procedure for concurrent pseudo random number (PRN) generation. In this paper, a parallel PRNG with multiple outputs based on LFSR of Fibonacci and Galois type, which is used to drive several IP cores within a SoC (System on Chip) during single clock cycle simultaneously is presented. The crucial novelty of the design deals with involving a combination of pipelining and parallelism in LFSR operation. Based on the mathematical model for the polynomial (1), it shows how it is possible to generate k , $1 \leq k \leq n - 1$ parallel outputs with the multiple output parallel LFSR architecture. The basic idea is based on concurrent generation of $k = 4$ pseudo random sequences. As it can be seen, parallel sequences are staggered and interleaved so that each parallel LFSR scheme reproduces the sequential sequence with different order of PRNs. Note that the length of each generated parallel sequence is identical to the length of the original LFSR, but with a different pattern.

For the sake of verification they have implemented PLFSR F that generates two and three consecutive pseudo random sequences in parallel. For FPGA implementation they have used Xilinx development CAD tool ISE WebPack 13.1. Design verification was performed using test benches intended for excitation of PLFSR F and standard LFSR. FPGA devices from Virtex-6 LP series (circuit xc6vlx75tl-1lf484) were used. Implementation of parallel test pattern generator based on a linear feedback shift register (LFSR) with multiple outputs used as a building block in built-in-self-test (BIST) design within SoC. The proposed design can drive several circuits under test (CUT) simultaneously. The mathematical procedure for concurrent pseudo random number (PRN) generation is described. They have implemented LFSRs that generate two and three PRNs in FPGA and ASIC technology. The design was tested at the operating frequency of 400 MHz.

Fault-Tolerant Reconfigurable Low-Power Pseudorandom Number Generator
V. Petrović and Z. Stamenković, M. Stojčev, T. Nikolić, and G. Jovanović,
2013

To design a Fault-Tolerant Reconfigurable Low-power pseudoRandom number Generator (FT_RLRG) which integrates both Fibonacci and Galois LFSRs into a single hardware core. To design a linear feedback shift register (LFSR) which is the heart of any embedded digital system that relies on pseudorandom bit sequences. LFSR belongs to the devices known as state machines. It, using feedback, modifies itself on each rising edge of the clock. LFSR can be created using the Galois and Fibonacci configurations of gates and registers, alternatively known as internal-XOR LFSR and external-XOR LFSR. In Fibonacci implementation, the outputs of some registers are XOR-ed with each other and fed back to the input of the shift register. In the Galois implementation, the gates are placed between the registers.

The advantages of this solution are superior performance in respect to speed of operation, silicon area and power dissipation, but the main disadvantages relate to inflexibility of the hardware to adopt testing update demands and high cost of custom ASIC fabrication. In order to achieve a reasonable tradeoff of speed and flexibility, for both design approaches, we propose here a reconfigurable hardware. During this, the two main features of LFSR, the polynomial degree (determined by the length n of the shift register) and characteristic polynomial type (defines the number of active polynomial terms), can be implemented as reconfigurable hardware.

Proposed FT_RLRG design was implemented as an ASIC in the IHP's 130 nm CMOS technology. The results show that the area overhead and power consumption are proportional to the polynomial degree n and small enough. The FT_RLRG test core seems to be a promising solution for test issues of advanced digital integrated circuits. The design of FT_RLRG is of practical interest in testing triple modular fault-tolerant systems in the presence of single event upsets (SEUs), especially in case when the system is SRAM based. Fault can be permanent, transient or intermittent. Permanent faults reflect irreversible physical changes. Intermittent faults occur because of unstable source of information or marginal hardware, while transient faults emerge due to temporary environmental conditions. Intermittent and permanent faults appear after several months of the circuit's operation and can be removed by replacing the faulty circuit.

A study on the effect of shifting on LFSR PRNG
Ankur Kumar Varshney , Sushil Kumar Sharma and Rajesh Singh
2012

To generate Random numbers which are useful for a variety of purposes as generating data encryption keys, simulating and modeling complex phenomena using LFSR PRNG. To program a series of logical steps that produce numbers that do not follow some definite sequence. 7 different tapping are chosen. These tapping are selected by taking certain thing into mind such as redundancy and randomness. These tapings are fed into tapping selector table. A 3bit PRNG is used to randomly select these tapings. The initial vector is chosen 001 because it will generate all possible number between 1 to 7. Case 0 is prohibited here because when the all zero condition arises it will generate on zero value onwards. The 3bit PN sequence generator gives a selection to the tapping selector to choose a particular tapping among 7 selections . This tapping selection is given to the feedback network.

The linear feedback shift register is one of the most useful techniques for generating pseudorandom numbers. In short, an LFSR takes a series of bits from a long shift register, XORs them together to come up with a resultant bit, shifts the register along one bit, and then sticks the new bit back into the beginning of the register. PRNG is widely used method improving the controllability of random number. The PRNG can be implemented through software or hardware. For hardware implementation, LFSR is the most popular. It advantages depend on simple implementation and high speed performance. However, it has poor security in terms violability. The output bit of an LFSR that is n bits long, the feedback tapping are kept changing which make the generated code quite complex. The output of a model can be apportioned to different sources of uncertainty in the model input. A related practice is uncertainty analysis which focuses rather on quantifying uncertainty in model output.

There are two kinds of the random number generator (RNG): truly random number generator (TRNG) and pseudorandom number generator (PRNG). The most common way to implement a random number generator is LFSR. Codes generated by the LFSR are actually pseudo random sequences (PN) because the sequence repeats itself after a certain number of cycles. Random numbers are a fundamental tool in many cryptographic applications like key generation, encryption, masking protocols etc.

Parallel LFSR Reseeding with selection register for mixed-mode BIST
Piyanart Kongtim and Taweesak Reungpeerakul
2010

To represent a new parallel LFSR reseeding with selection register for mixed-mode BIST in order to reduce the number of test data. The selection register has been added to the original parallel reseeding in order to further reduce test data. The state of signal is determined by BIST controller Extra 2-input AND gates are added for switching between Pseudo-Random Pattern Generator (PRPG) and Deterministic Test Pattern Generator (DTPG). One input of AND gates from BIST controller is a control signal, which is used to control between enable mode and disable mode seed injection. For enable mode, seeds are fed into LFSR through phase shifter to scan chains. For disable mode, data of scan cells are received from combination of LFSR through phase shifter. Extra XOR gates are added in LFSR for feeding seeds in DTPG mode. Position of tap seeds depends on the tap of phase shifter.

The experimental results of parallel reseeding were performed with ISCAS 89 benchmark circuits. The information details of ISCAS 89 benchmark circuits are monitored. The Structural dependency in output of LFSR is the fact that the patterns coming from each stage of the LFSR are identical to those of the previous stage but delayed by one clock cycle. The partial LFSR reseeding scheme uses output of LFSR in serial mode, which is not suitable with XOR network that shifts in parallel form.

Here a new parallel LFSR reseeding with selection register for mixed-mode BIST in order to reduce the number of test data. The dynamic seeds were injected in parallel form through the LFSR, phase shifter, and scan chains, respectively. The selection register was added to improve the efficiency of encoding test data. For seed computing, after the equations were formed, they were solved by Gauss-Jordan elimination. The experimental results for ISCAS 89 benchmark circuits indicate the significant improvement in terms of test data. There are several main advantages of proposed approach such as 100% test coverage, low test data, low test application time, high fault coverage as intended by the deterministic patterns, etc.

Efficient PGA LFSR Implementation Whitens Pseudorandom Numbers
Leonard Colavito and Dennis Silage
2009

To perform efficient PGA Linear Feedback Shift Register (LFSR) and to implement in Whitens Pseudorandom Numbers. For this digital communication system simulation in PGA hardware it requires ten pseudorandom number sources that are utilized in pairs to produce complex noise and multipath characteristics. The pseudorandom number generators (PNG) all utilize an identical 55-bit LFSR and a maximal length generating polynomial. The LFSR are initially seeded to provide 10 non-overlapping subsequences. Our simulation requires 16 of the 55 PNG register bits as an unsigned 16-bit integer value. The sequences are then transformed by look up tables to provide Gaussian distributions. This technique works adequately for simple simulation models. However, in our multipath simulation this basic technique introduced inaccuracies that precluded passing of our requisite validation test cases.

After replacing all of the LFSR number generators in multipath simulation with the skip-ahead PNG, all of requisite validation test cases were passed. Additionally, sample sequences from the PNG were also analyzed for autocorrelation and crosscorrelation performance, discernable patterns when sequence values are plotted in pairs and for uniform distribution. All of these analyses produced very acceptable results. Linear feedback shift registers (LFSR) are commonly utilized in digital communication system simulations as noise and data sources because they are easily implemented and require minimal resources. However, the LFSR exhibits a deficiency common to all multiplicative congruential pseudorandom number generators (PNG). This class of PNG exhibits a correlation between successive values that gives an undesired low-pass characteristic to the generated sequence. This characteristic can affect the simulation results when whiteness of the pseudorandom random sequence is assumed. Several techniques have been proposed to mitigate this deficiency. Here the skip-ahead technique can effectively whiten the pseudorandom number sequence produced by an LFSR. The skip-ahead technique produces values at the same rate as the basic LFSR and so does not degrade system performance.

2.2 LITERATURE SUMMARY

Linear feedback shift registers have many applications. Serial LFSR is used as test pattern generator in BIST. There are many applications of LFSR. In multibit LFSR multiple bits shift at every rising edge of clock pulse whereas in serial LFSR only one bit is shifted at every rising edge of clock. As multiple bit shifts in Multibit LFSR, this LFSR is more secure as compared to serial LFSR. There are two implementation styles of LFSRs, Galois implementation and Fibonacci implementation. In Fibonacci implementation style XOR gate is present in the feedback path, therefore this style is also known as an out-of-line, simple type (S-type), standard or external feedback LFSR. The difference in modular LFSR compared to standard LFSR is due to the positions of the XOR gates in the feedback function. Modular LFSR works faster than standard LFSR, because it has at most one XOR gate between adjacent flip-flops, while there can be several levels of XOR gates in the feedback of standard LFSR. Since the XOR gate is in the shift register path, the Galois implementation is also known as an in-line, internal feedback or modular type (M-type) LFSR. Applications of LFSR are in Cryptography, used as a counter, used in testing for pattern generation and used as BCH and CRC encoder.

CHAPTER 3

EXISTING SYSTEM

3.1 FAULT TOLERANT LINEAR FEEDBACK SHIFT REGISTER

Linear Feedback Shift Register (LFSR) is the main component of the embedded systems that are based on pseudorandom bit sequences such as pseudo-random number and pseudo-noise sequence generators, fast digital counters, and Built-In-Self Tests. In real-time and safety-critical embedded applications, incorrect functionality may cause catastrophic effects and thus, the fault-tolerance is known as a traditional design requirement. In recent years, the continuous reduction of the transistor dimensions and supply voltage levels have led to an increase in the vulnerability of Integrated Circuits (ICs) against various hardware faults. Therefore, fault tolerant design of LFSR considering hardware faults is very essential in safety-critical embedded applications. Linear Feedback Shift Registers (LFSR) are extensively used in variety of applications such as Built-In-Self-Test circuits or Pseudo Random Number Generators. Hence, fault tolerant design of LFSR is essential for the applications with high reliability demands.

Traditional fault tolerant LFSRs include large number of Single-Point-of-Failures (SPoFs) in which any fault results in the whole system failure. In this paper, a new fault tolerant architecture for LFSR (named as FT-LFSR) is proposed in which the number of SPoFs are significantly reduced compared to the previous ones. To this end, a modified version of Triple Modular Redundancy (TMR) empowered with some extra controlling units for identifying the operational module is used. In addition, a novel metric called Reliability-Area-Factor (RAF) is introduced to evaluate the efficacy of the redundancy-based fault tolerant techniques (such as FT-LFSR) in terms of number of SPoFs and the area overhead. Experimental results show that, the FT-LFSR is resilient to all single transient and permanent faults except in its limited SPoFs and many patterns of multiple faults. Parallel LFSRs performance analysed in terms of number of ones, number of XOR gates, Delay element and Critical path delay.

3.2 EXISTING FT-LFSR

Existing design of LFSR is a new fault tolerant architecture for LFSR (named as FT-LFSR) is proposed in which the number of SPoFs are significantly reduced compared to the previous ones. Linear Feedback Shift Register (LFSR) is the main component of the embedded systems that are based on pseudorandom bit sequences such as pseudo-random number and pseudo-noise sequence generators, fast digital counters, and Built-In-Self Tests. SPoF is defined as the design point that will result in the entire system failure if it fails. a new fault tolerant LFSR (named as FTLFSR) is proposed in which the number of SPoFs is significantly reduced. To introduce a novel metric called Reliability-Area-Factor (RAF) to evaluate the efficacy of the redundancy-based fault tolerant techniques such as FT-LFSR.

The introduction of ReliabilityArea-Factor (RAF) integrates the number of SPoFs and the area overhead of the techniques. Higher RAF means that the proposed fault tolerant architecture provides higher reliability in terms of less area overhead. More precisely, a fault tolerant architecture is better than the original one in terms of RAF if $1 < \text{RAF} < \infty$ and it is worse if $0 < \text{RAF} < 1$. The fault tolerant architecture is as the same as original architecture if $\text{RAF} = 1$. Higher RAF value of FT-LFSR and higher fault coverage of it shows that we can use RAF in hardware redundancy as a fair reliability metric in fault tolerant designs.

Their proposed design is simulated and implemented using Verilog on Xilinx Spartan 6. The fault injection-based experimental results show that, FT-LFSR tolerates all single transient and permanent faults (except the ones which occur in its limited SPoFs) and many patterns of multiple faults. The fault tolerance of FT-LFSR is investigated quantitatively and also through some experimental results on Spartan 6 FPGA. The results show the high fault tolerance of FT-LFSR against single and multiple faults in expense of reasonable area and delay overhead. A fault tolerant architecture for LFSRs called FT-LFSR is proposed which is applicable pseudorandom sequence generators such as BIST structures and pseudo random number generators. The main superiority of FT-LFSR over its counterparts is its less number of SPoFs. This leads to high resiliency to single (transient and permanent) faults and many patterns of multiple faults.

3.3 EXISTING PROCEDURES

LFSR is a state machine composed of a shift register whose input bit is a linear function of its previous state. There are two main forms of LFSR; i.e. Galois and Fibonacci configurations. In the Fibonacci implementation, the outputs from some of the registers are XORed with each other and fed back to the input of the shift register. In the Galois implementation, the gates are placed between the registers. The conventional Fibonacci and Galois LFSR implementations are shown below.

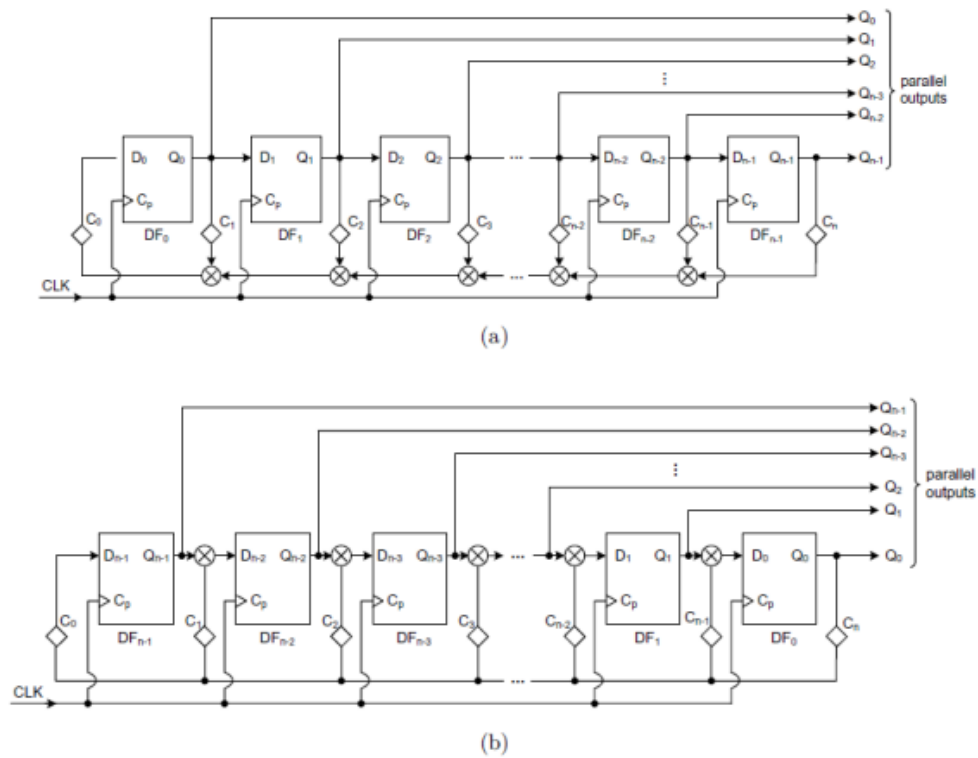


Fig:3.1 : LFSR structure. (a) Fibonacci implementation; (b) Galois implementation.

n denotes the length of the LFSR, i.e., the number of flip-flops, $DF_0, \dots, DF_n$, in the register. $c_0, c_1, \dots, c_n$ are the binary coefficients that correspond to $n + 1$ feedback terms. The numbering of the feedback for the Galois implementation is the reverse of that for the Fibonacci variant. The characteristic polynomial for the LFSR with n flip-flops has the following form:

$$c_0.1 + c_1.x + c_2.x^2 + \dots + c_{n-1}.x^{n-1} + c_n.x^n \quad (3.1)$$

when $c_i = 1$, It means that a connection exists and the respective feedback is set. On the contrary when $c_i = 0$ implies that no connection exists and the corresponding XOR gate can be replaced by a direct connection from its input to its output [8]. Both c_0 and c_n are always equal to one.

The initial value of the LFSR is called the seed. A LFSR with a well-chosen feedback function can produce a sequence of parallel words, which appears random in nature but has a finite number of possible states with a very long repeating cycle. The outputs that influence the inputs are named taps. The sequences generated by both the Fibonacci and Galois implementations go through all possible $2^n - 1$ states except the forbidden all-zero state, in which case it will never change. Only characteristic polynomials that are primitive (in the mathematical sense) cause the LFSR to go through all possible states and to repeat with a period of $2^n - 1$. The output of such a LFSR configuration is called a maximum-length sequence. In the sequel, we will focus on Fibonacci LFSRs. Such LFSRs have been proven to be equivalent to Galois LFSR with respect to the property of generating the same output vectors.

3.4 RELIABILITY-AREA-FACTOR (RAF) METRIC

In order to evaluate the efficacy of hardware redundancybased techniques, we propose a novel metric called ReliabilityArea-Factor (RAF) which integrates the number of SPoFs and the area overhead of the techniques.

RAF is computed as:

$$RAF = S * A \quad (3.2)$$

where S and A respectively show the ratio of number of SPoFs and the area in the original and fault tolerant architectures. Parameter S is calculated as:

$$S = \frac{\sum_i NSO_i}{\sum_j NSF_j} \quad (3.3)$$

where NSO_i shows the number of logic blocks in ith SPoF in the original architecture and NSF_j is the number of logic blocks in jth SPoF in the fault tolerant architecture. Higher parameter S is proportional to the higher reliability achieved by the fault tolerant architecture.

Parameter A is calculated as:

$$A = \frac{N_b}{N_f} \quad (3.4)$$

where N_b and N_f respectively represent the total number of logic blocks in the original and fault tolerant architectures. The overhead imposed by the fault tolerant approach leads to an increase in parameter A. Therefore, RAF integrates the reliability improvement and the imposed area overhead achieved by a fault tolerant technique. Higher RAF means that the proposed fault tolerant architecture provides higher reliability in terms of less area overhead. More precisely, a fault tolerant architecture is better than the original one in terms of RAF if $1 < \text{RAF} < \infty$ and it is worse if $0 < \text{RAF} < 1$. The fault tolerant architecture is as the same as original architecture if $\text{RAF} = 1$. Higher RAF value of FT-LFSR and higher fault coverage of it shows that we can use RAF in hardware redundancy as a fair reliability metric in fault tolerant designs

The results show the high fault tolerance of FT-LFSR against single and multiple faults in expense of reasonable area and delay overhead.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 PROPOSED FT-LFSR

Traditional fault tolerant LFSRs include large number of Single-Point-of-Failures (SPoFs) in which any fault results in the whole system failure. In this proposed technique, a new fault tolerant architecture for LFSR (named as FT-LFSR) is proposed in which the number of SPoFs are significantly reduced compared to the previous ones. To this end, a modified version of Triple Modular Redundancy (TMR) empowered with some extra controlling units for identifying the operational module is used. The proposed approach shows, concept of reducing the transitions in test pattern generated. The transition is reduced by increasing correlation between the successive bits. The simulation results shows how patterns are generated for the applied seed vector.

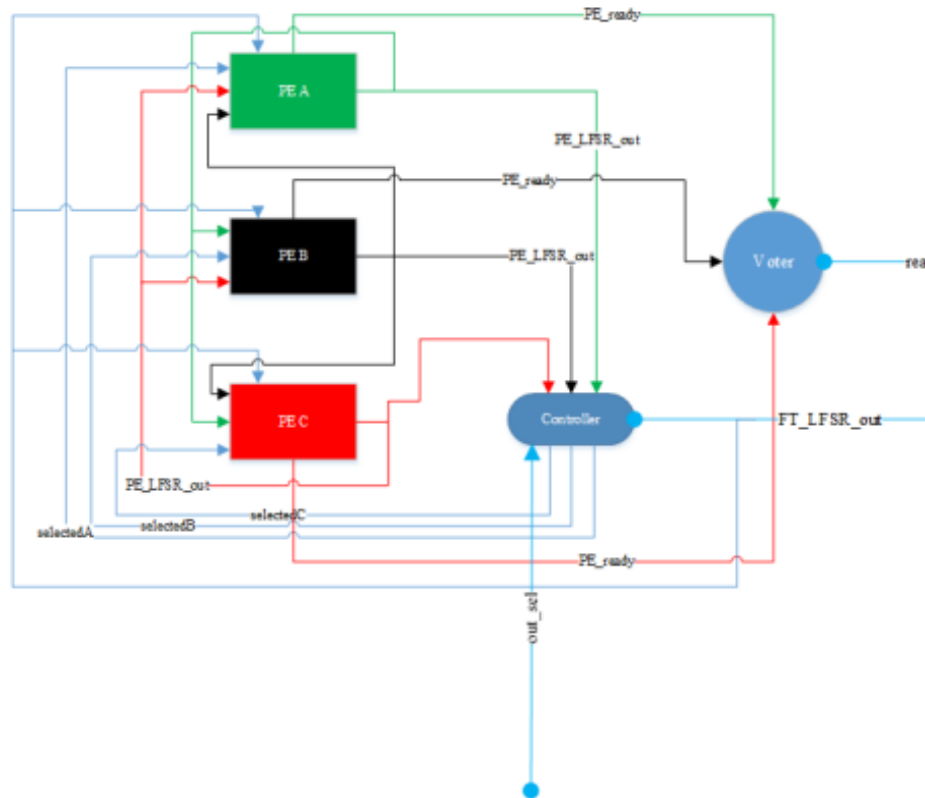


Fig:4.1 : Block diagram of proposed FT-LFSR

The proposed architecture FT-LFSR has three PEs and one voter based on TMR approach with Controller module which reduces the number of SPoFs. This architecture has two outputs and one input :

1. *out_sel.*

out_sel comes from a module which controls the FTLFSR (e.g. BIST) and determines which PE must be connected to FT LFSR out (selected PE).

2. *FT LFSR out.*

FT LFSR out is the N bit output of selected PE which is the same as a conventional LFSR's output.

3. *ready.*

ready shows that FT LFSR out is valid or not. if ready equals one, it means that FT LFSR out is valid and otherwise it means invalid.

To demonstrate how this FT-LFSR works from when input is given until two outputs are ready, the following steps are done respectively:

- PE corresponds to Processing Element. Each PE is a conventional LFSR.
- The second and third items clock and reset are also the inputs of FT-LFSR but they are not shown in the figure to not make it messy.

At first, out_sel which is a 2-bit input for Controller, is initialized. Then Controller decides which PE LFSR out must be connected to FT-LFSR out. Now FT-LFSR out is available. This output goes to PEs to compare. Controller signals which PE is selected PE by selected A, selected B and selected C. Each PE based on being selected or not determines to compare its output with whom. Selected PE compares its output with two other PE's outputs and other PEs compare their outputs just with FT-LFSR out (selected PE's output). After comparison, each PE produces PE ready that shows correctness of FT-LFSR out based on its point of view. If no mismatch in total N bits occurs then PE ready=1, otherwise PE ready = 0. Finally, Voter which is a majority voter, votes between PE readys of PE A, PE B and PE C and sets/resets the ready output. Now one period of FT-LFSR is finished.

4.2 EXTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR

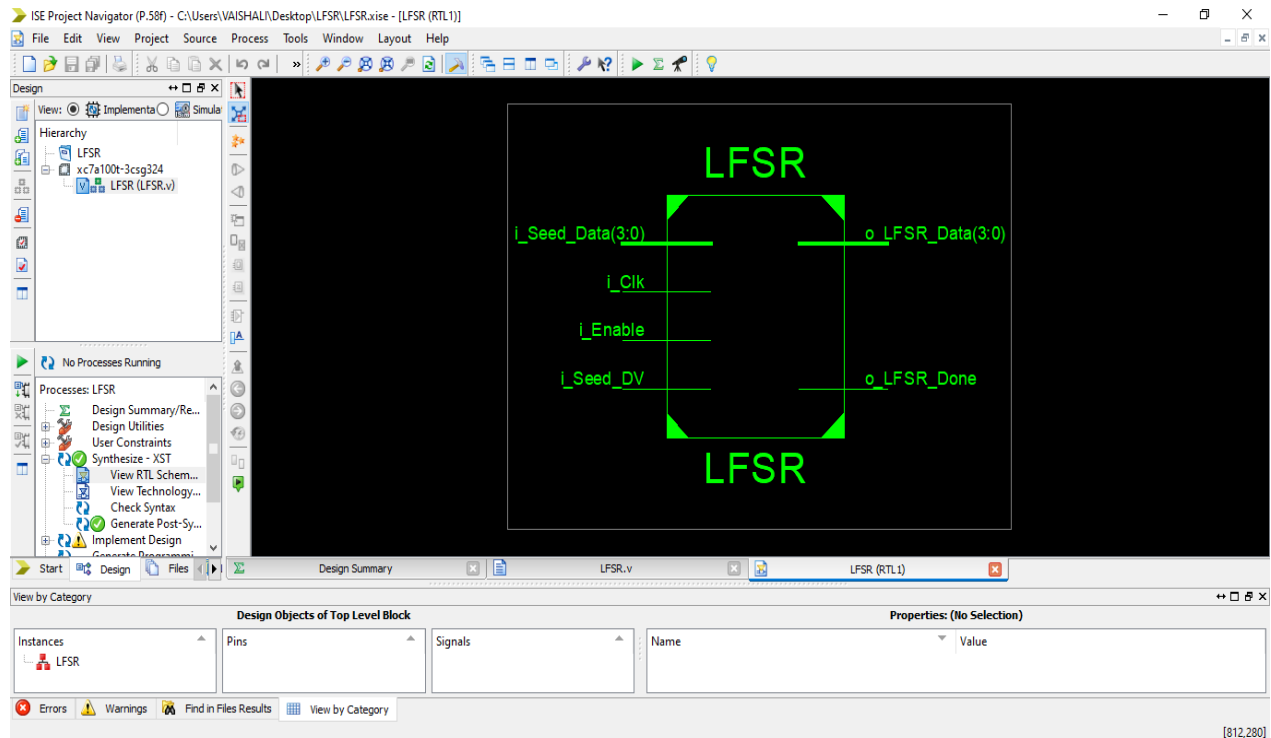


Fig:4.2 : External Block diagram of proposed FT-LFSR designed in Xilinx

The external block diagram of proposed FT-LFSR consists of:

Seed :

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream 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 that appears random and has a very long cycle.

In practice, to start an LFSR we must supply the starting values for the registers. These starting values are called the "seed" and putting them into the registers is called "seeding the LFSR". Any set of values can be used as a seed, but if we use all 0's as a seed, the only output you will get is all 0's.

Enable :

An Enable will allow an input signal, to pass the output, when the control signal is high. It will prevent a signal from passing when the control signal is switched to low. When a signal is enabled, the output will match the input.

Clock :

LFSR is a synchronous digital circuit, a clock signal oscillates between a high and a low state and it is used like a metronome to co-ordinate actions of digital circuits. A clock signal is used to trigger sequential circuits.

4.3 INTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR

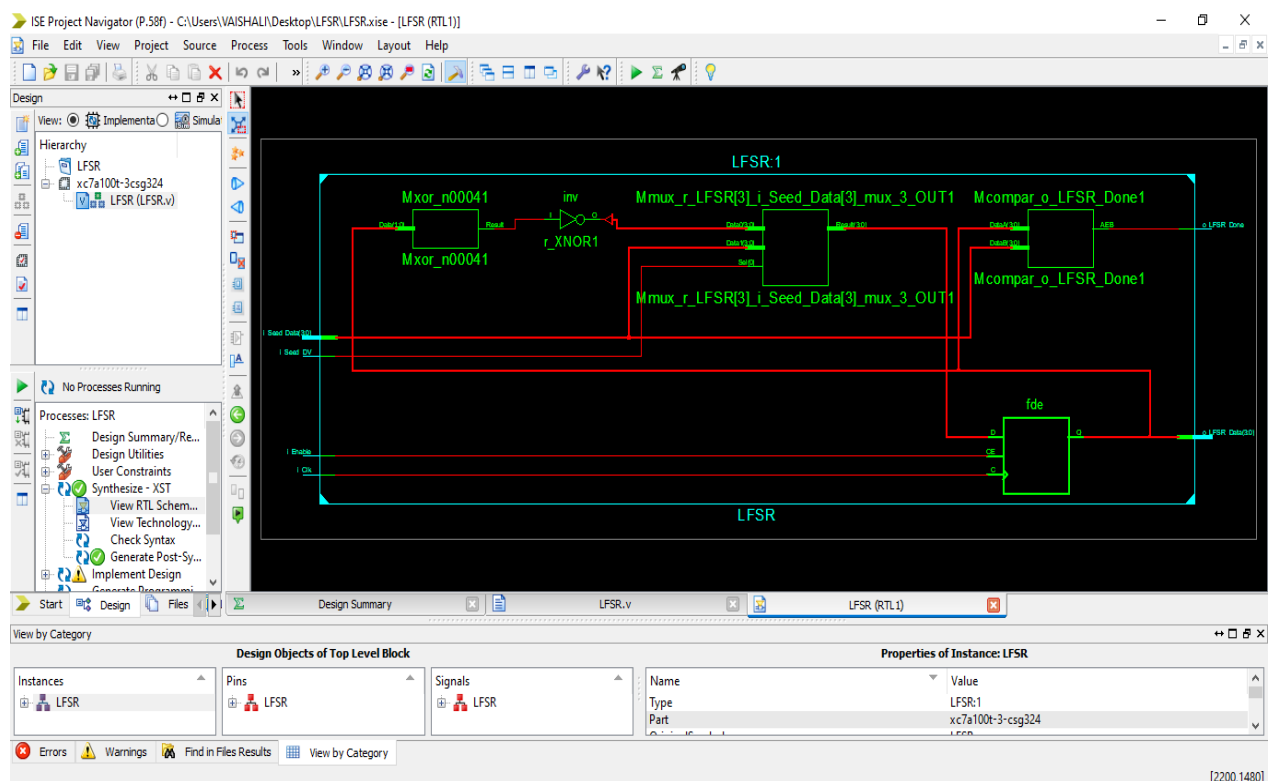


Fig:4.3 : Internal Block diagram of proposed FT-LFSR designed in Xilinx

The above shown Fig:4.3 is the internal block diagram of the proposed FT-LFSR designed in Xilinx Software.

The internal block diagram of FT-LFSR consists of:

- XOR GATE
- XNOR GATE
- MULTIPLEXER
- COMPARATOR
- D FLIP FLOP

Gate level modelling are done using Modelsim software and full architecture is constructed in Xilinx software.

4.3.1 XOR Gate

Here we used 2-input XOR gate. The XOR gate stands for the Exclusive-OR gate. This gate is a special type of gate used in different types of computational circuits. Apart from the AND, OR, NOT, NAND, and NOR gate, there are two special gates, i.e., Ex-OR and Ex-NOR. These gates are not basic gates in their own and are constructed by combining with other logic gates. Their Boolean output function is significant enough to be considered as a complete logic gate. The XOR and XNOR gates are the hybrids gates.

The 2-input OR gate is also known as the Inclusive-OR gate because when both inputs A and B are set to 1, the output comes out 1(high). In the Ex-OR function, the logic output "1" is obtained only when either A="1" or B="1" but not both together at the same time. Simply, the output of the XOR gate is high(1) only when both the inputs are different from each other.

The plus(+) sign within the circle is used as the Boolean expression of the XOR gate. So, the symbol of the XOR gate is $\oplus$. This Ex-OR symbol also defines the "direct sum of sub-objects" expression. There are two types of Exclusive-OR gate: 2-input XOR gate and 3-input XOR gate. Since we have used 2-input XOR gate, the 2-input XOR gate is explained below.

2-input XOR Gate:

This is a simple form of the hybrid gate XOR. In this type of XOR gate, there are only two input values and an output value. There are $2^2=4$ possible combinations of inputs. The output level is high when both inputs are set to a different logic level. The Boolean expression of 2-input XOR gate is as follows:

$$Y=(A\oplus B)$$

$$Y=(A' B+AB')$$

The truth table and logic design are given below:

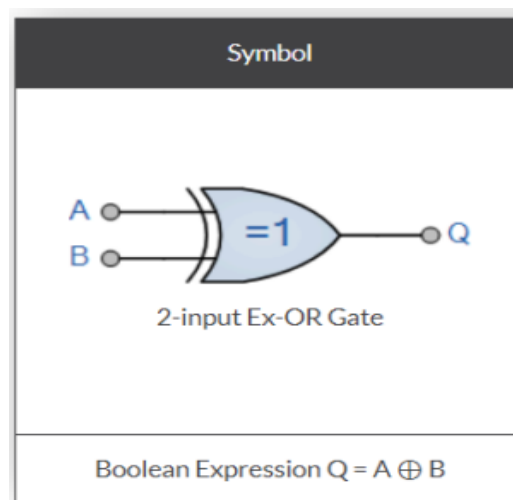


Fig:4.4 : Logic design of 2-input XOR gate

Truth Table		
B	A	Q
0	0	0
0	1	1
1	0	1
1	1	0
A OR B but NOT BOTH gives Q		

Fig:4.5 : Truth Table of 2-input XOR gate

4.3.2 XNOR Gate

Here we used 2-input XNOR gate. The XNOR gate is the complement of the XOR gate. It is a hybrid gate. Simply, it is the combination of the XOR gate and NOT gate. The output level of the XNOR gate is high only when both of its inputs are the same, either 0 or 1. The symbol of the XNOR gate is the same as XOR, only complement sign is added. Sometimes, the XNOR gate is also called the Equivalence gate.

2-input XNOR Gate:

It is a simple form of the hybrid gate XNOR. In this type of XNOR gate, there are only two input values and an output value. There are $2^2=4$ possible combinations of inputs. The output level is high when both inputs are set to high(1). The Boolean expression of 2-input XNOR gate is as follows:

$$Y=(A\oplus B)'$$

$$Y=((AB)'+AB)$$

The truth table and logic design are given below:

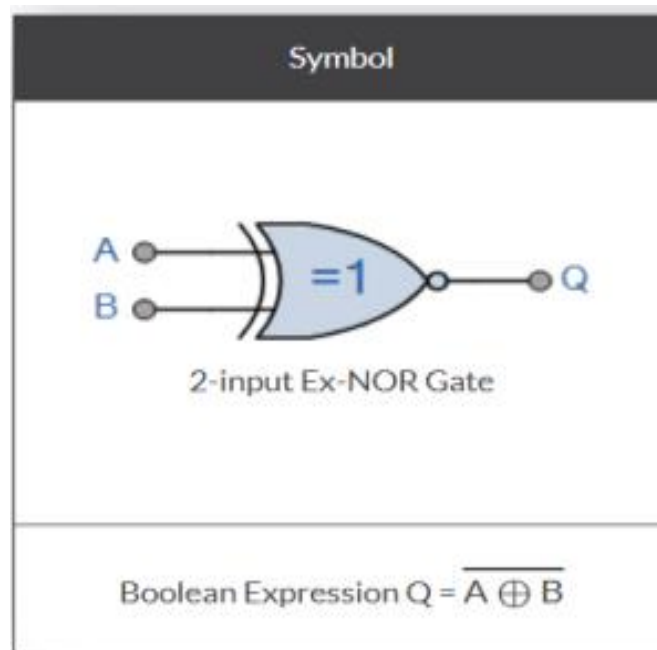


Fig:4.6 : Logic design of 2-input XNOR gate

Truth Table		
B	A	Q
0	0	1
0	1	0
1	0	0
1	1	1
Read if A AND B the SAME gives Q		

Fig:4.7 : Truth Table of 2-input XNOR gate

4.3.3 Multiplexer

A multiplexer is a combinational circuit that has 2^n input lines and a single output line. Simply, the multiplexer is a multi-input and single-output combinational circuit. The binary information is received from the input lines and directed to the output line. On the basis of the values of the selection lines, one of these data inputs will be connected to the output.

Unlike encoder and decoder, there are n selection lines and 2^n input lines. So, there is a total of 2^N possible combinations of inputs. A multiplexer is also treated as Mux.

There are various types of multiplexers. Since we have used 4 to 1 multiplexer, the 4 to 1 multiplexer is explained below

4x1 Multiplexer:

In the 4x1 multiplexer, there is a total of four inputs, i.e., A_0 , A_1 , A_2 , and A_3 , 2 selection lines, i.e., S_0 and S_1 and single output, i.e., Y . On the basis of the combination of inputs that are present at the selection lines S_0 and S_1 , one of these 4 inputs are connected to the output. The block diagram and the truth table of the 4x1 multiplexer are given below.

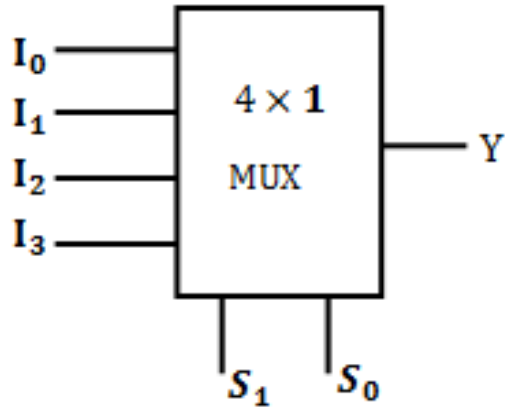


Fig:4.8 : Block diagram of 4x1 Multiplexer

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Fig:4.9 : Truth Table of 4x1 Multiplexer

The logical expression of the term Y is as follows:

$$Y = S_1' S_0' A_0 + S_1' S_0 A_1 + S_1 S_0' A_2 + S_1 S_0 A_3$$

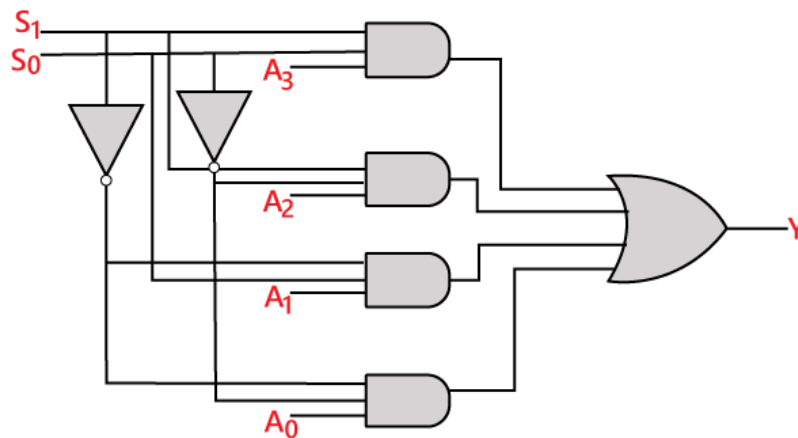


Fig:4.10 : Logic design of 4x1 Multiplexer

4.3.4 Comparator

Here we used 2-bit magnitude comparator. In electronics, a comparator is a device that compares two voltages or currents and outputs a digital signal indicating which is larger. It has two analog input terminals and one binary digital output. A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for $A > B$ condition, one for $A = B$ condition and one for $A < B$ condition. There are various types of comparators. Since we have used 2-bit magnitude comparator, the 2-bit magnitude comparator is explained below

2-Bit Magnitude Comparator:

A comparator used to compare two binary numbers each of two bits is called 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers. The logical expressions are:

$$A > B : A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$$

$$\begin{aligned} A = B &: A_1'A_0'B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0' \\ &: A_1'B_1' (A_0'B_0' + A_0B_0) + A_1B_1 (A_0B_0 + A_0'B_0') \\ &: (A_0B_0 + A_0'B_0') (A_1B_1 + A_1'B_1') \\ &: (A_0 \text{ Ex-Nor } B_0) (A_1 \text{ Ex-Nor } B_1) \end{aligned}$$

$$A < B : A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$$

The truth table and logic design are given below:

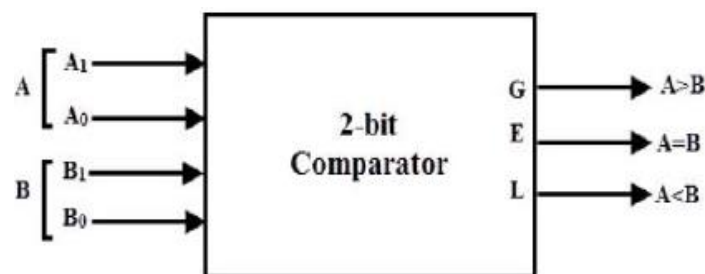


Fig:4.11 : Block diagram of 2-Bit Magnitude Comparator

Inputs				Outputs		
A ₁	A ₀	B ₁	B ₀	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Fig:4.12 : Truth Table of 2-Bit Magnitude Comparator

4.3.5 D-Flip Flop

In SR NAND Gate Bistable circuit, the undefined input condition of SET = "0" and RESET = "0" is forbidden. It is the drawback of the SR flip flop. This state:

1. Override the feedback latching action.
2. Force both outputs to be 1.
3. Lose the control by the input, which first goes to 1, and the other input remains "0" by which the resulting state of the latch is controlled.

We need an inverter to prevent this from happening. We connect the inverter between the Set and Reset inputs for producing another type of flip flop circuit called D flip flop, Delay flip flop, D-type Bistable, D-type flip flop. The D flip flop is the most important flip flop from other clocked types. It ensures that at the same time, both the inputs, i.e., S and R, are never equal to 1. The Delay flip-flop is designed using a gated SR flip-flop with an inverter connected between the inputs allowing for a single input D(Data).

This single data input, which is labeled as "D" used in place of the "Set" input and for the complementary "Reset" input, the inverter is used. Thus, the level-sensitive D-type or D flip flop is constructed from a level-sensitive SR flip flop.

So, here $S=D$ and $R=\sim D$ (complement of D)

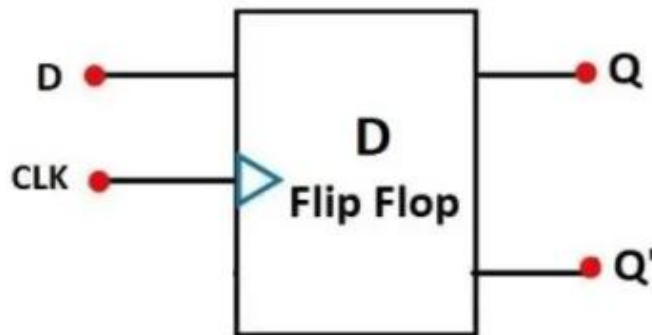


Fig:4.13 : Block diagram of D-Flip Flop

D	CLK	Q	Q'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Fig:4.14 : Truth Table of D-Flip Flop

The "CLOCK" or "ENABLE" input is used to avoid this for isolating the data input from the flip flop's latching circuitry. When the clock input is set to true, the D input condition is only copied to the output Q. This forms the basis of another sequential device referred to as D Flip Flop.

When the clock input is set to 1, the "set" and "reset" inputs of the flip-flop are both set to 1. So it will not change the state and store the data present on its output before the clock transition occurred. In simple words, the output is "latched" at either 0 or 1.

4.4 WORKING OF THE PROPOSED FT-LFSR

Linear Feedback Shift Register (LFSR) is the main component of the embedded systems that are based on pseudorandom bit sequences such as pseudo-random number and pseudo-noise sequence generators, fast digital counters, and Built-In-Self Tests. In real-time and safety-critical embedded applications, incorrect functionality may cause catastrophic effects and thus, the fault-tolerance is known as a traditional design requirement. In recent years, the continuous reduction of the transistor dimensions and supply voltage levels have led to an increase in the vulnerability of Integrated Circuits (ICs) against various hardware faults. Therefore, fault tolerant design of LFSR considering hardware faults is very essential in safety-critical embedded applications.

4.4.1 Triple Modular Redundancy Technique

The key concept of fault tolerance is redundancy; i.e. the extra capabilities are added to the original design which are not necessary in fault-free operation. The redundancy can be spatial and/or temporal. Triple Modular Redundancy (TMR) is a well-known and common fault tolerant technique based on spatial redundancy. In TMR, the design is triplicated to perform a process in parallel and their result is processed by a majority-voting system to produce a single output. Despite its efficacy, there is a major vulnerability in TMR technique called as Single-Point-of-Failure (SPoF). SPoF is defined as the design point that will result in the entire system failure if it fails. The majority-voting system in TMR is accounted as its SPoF. So, the number of SPoF is required to be considered in TMR-based fault tolerant techniques. Some previous works have proposed fault tolerant LFSR designs which are based on TMR approach. Although, they have provided fault tolerance against single fault, the number of SPoFs is increased significantly by triplicating the majority-voting system, leading to a considerable decrease in the reliability of the design.

A new fault tolerant LFSR (named as FTLFSR) is proposed in which the number of SPoFs is significantly reduced. To this end, a modified version of TMR empowered with some extra controlling units for identifying the operational module is used. We also introduce a novel comparison metric called Reliability-Area-Factor (RAF) to compare the

TMR-based fault tolerance techniques in which the number of SPoFs and area overhead of TMR approach is taken into consideration. The proposed design is simulated and implemented using Verilog on Xilinx and Modelsim. The fault injection-based experimental results show that, FT-LFSR tolerates all single transient and permanent faults (except the ones which occur in its limited SPoFs) and many patterns of multiple faults.

4.5 XILINX ISE SOFTWARE

Xilinx ISE (Integrated Synthesis Environment) is a discontinued software tool from Xilinx for synthesis and analysis of HDL designs, which primarily targets development of embedded firmware for Xilinx FPGA and CPLD integrated circuit (IC) product families. XILINX is an analog circuit simulator capable of performing transient, steady state and frequency domain analysis.



4.6 MODELSIM SOFTWARE

ModelSim is a multi-language environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartus Prime, PSIM, Xilinx ISE or Xilinx Vivado. Simulation is performed using the graphical user interface (GUI), or automatically using scripts.



CHAPTER 5

RESULTS AND DISCUSSION

The existing and proposed architecture are simulated by using Xilinx ISE simulator and Modelsim simulator. Output of the existing FT-LFSR and proposed FT-LFSR are analysed from the results and it is concluded that the proposed architecture reduces the power consumption compared with the existing system. The Synthesis is done using Xilinx ISE and Modelsim software.

5.1 SOURCE CODE AND SIMULATION OF EXISTING LFSR

The verilog source code and simulation result of existing Linear Feedback Shift Register is given below. When RST=1, the output will be 0. When RST=0, the output will be according to linear feedback shifting property.

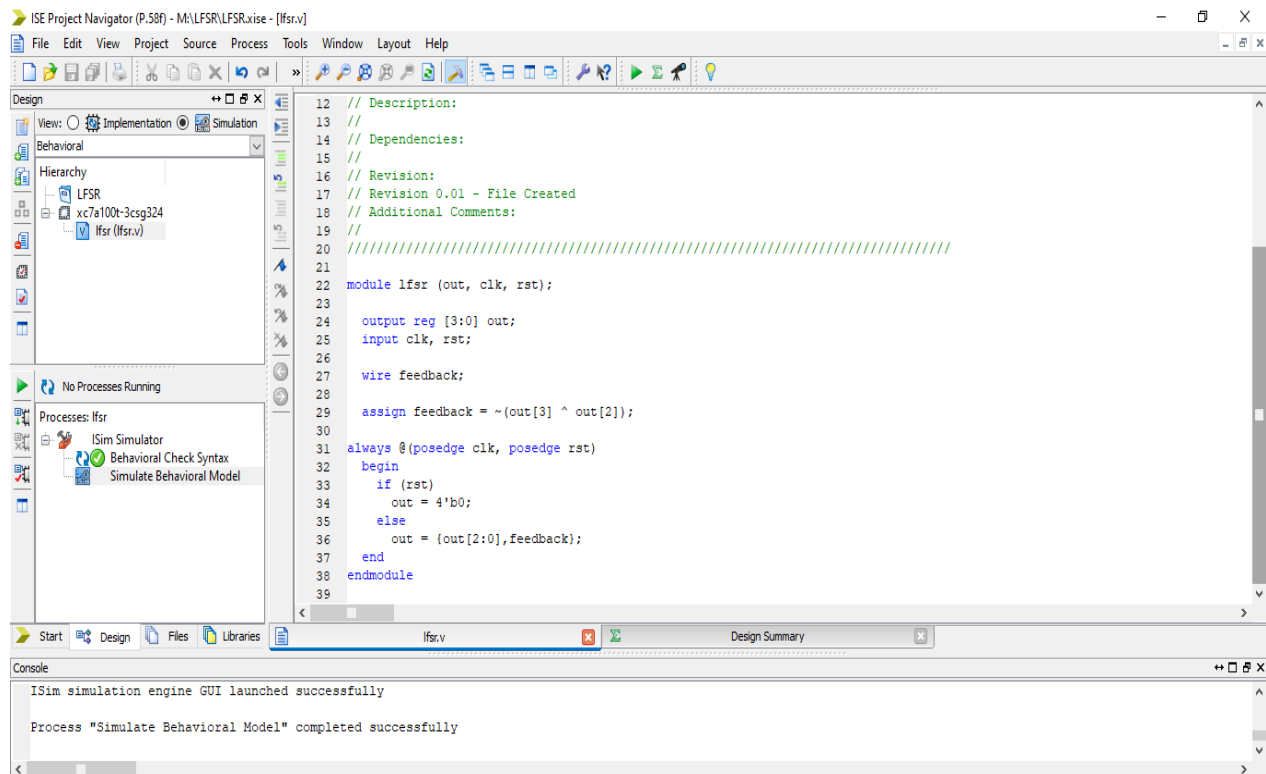


Fig:5.1 : Verilog Source code of Existing LFSR

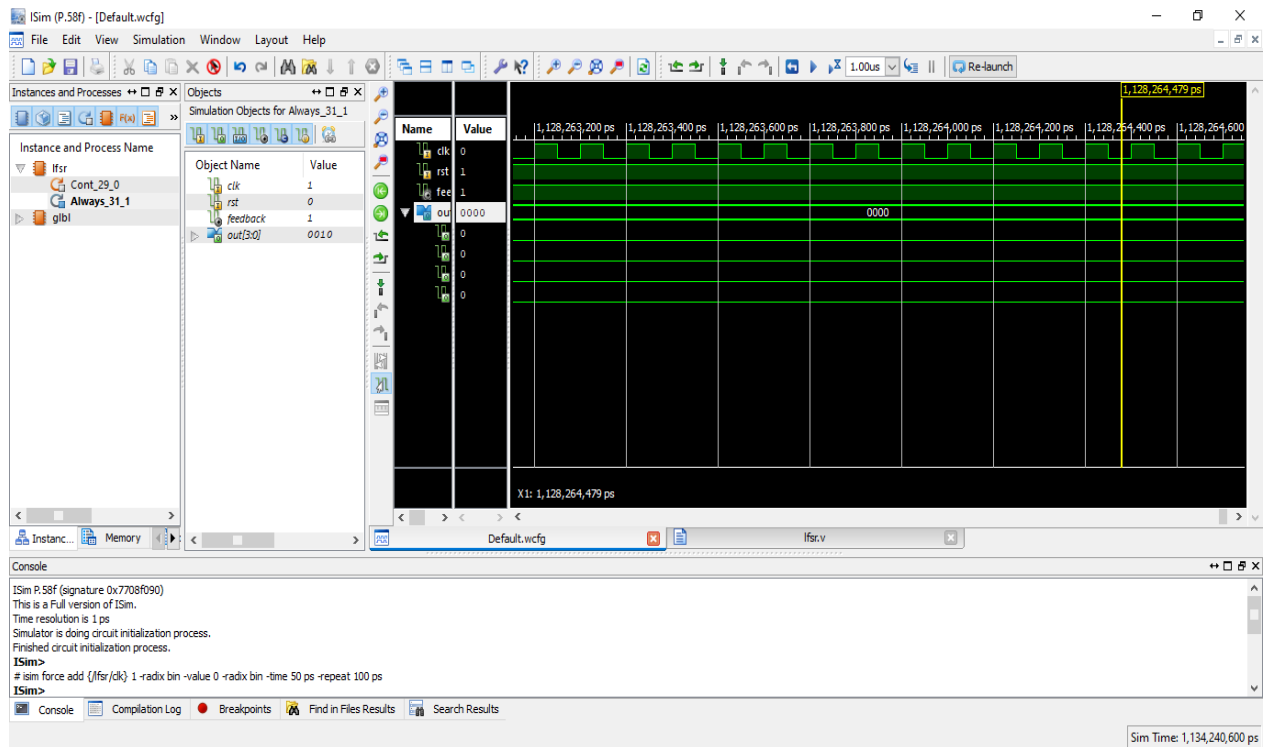


Fig:5.2 : Simulation of Existing LFSR, when RST=1

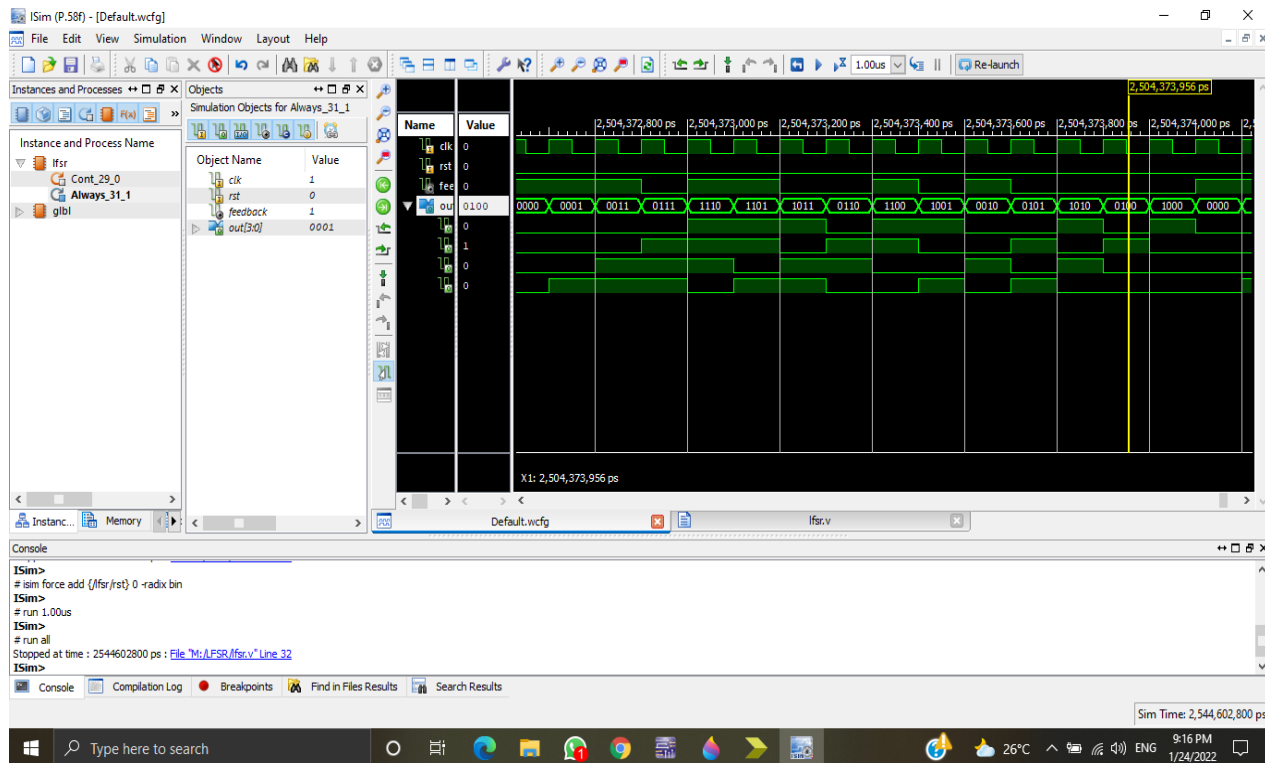


Fig:5.3 : Simulation result of Existing LFSR, when RST=0

5.2 SOURCE CODE AND SIMULATION OF INTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR

The internal block of the proposed FT-LFSR is designed in xilinx software and all the gate level modelling in the internal block diagram are done using modelsim software.

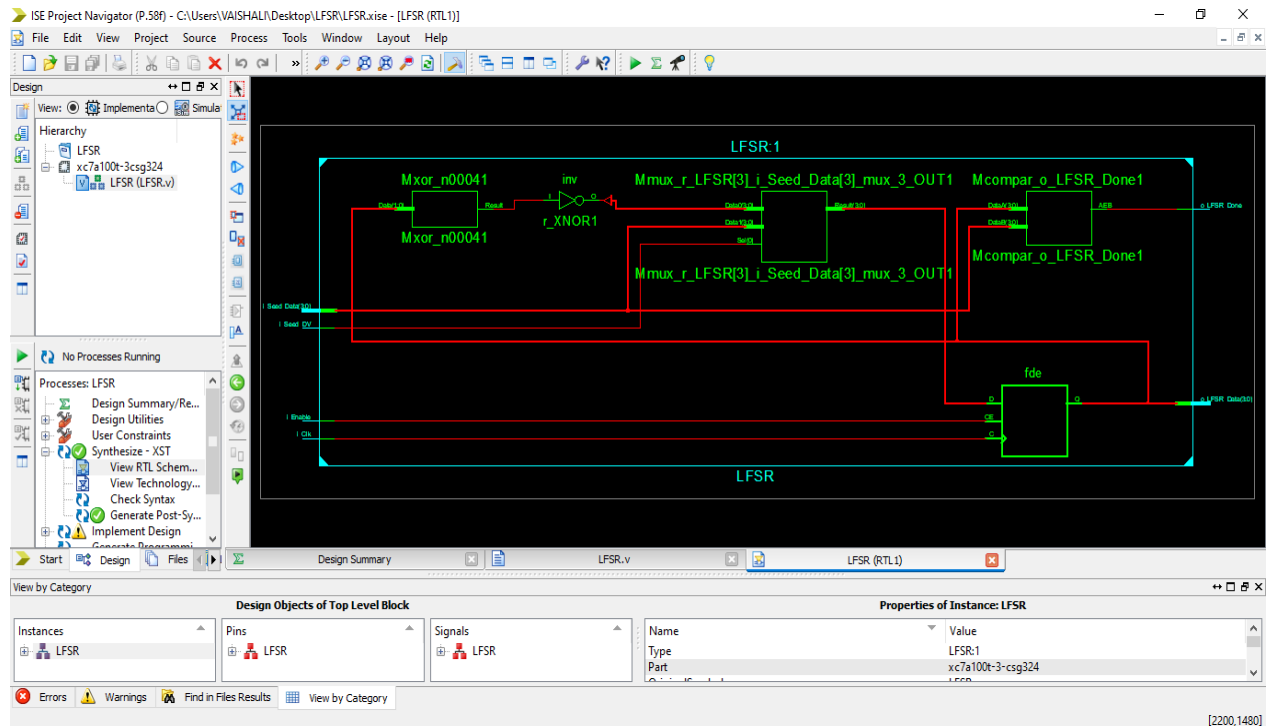


Fig:5.4 : Internal block design of the proposed FT-LFSR

The internal block diagram of FT-LFSR consists of:

- 2-INPUT XOR GATE
- 2-INPUT XNOR GATE
- 4x1 MULTIPLEXER
- 2-BIT MAGNITUDE COMPARATOR
- D FLIP FLOP

All these gate level modelling are done using Modelsim software.

5.2.1 Source code and simulation of 2-input XOR gate

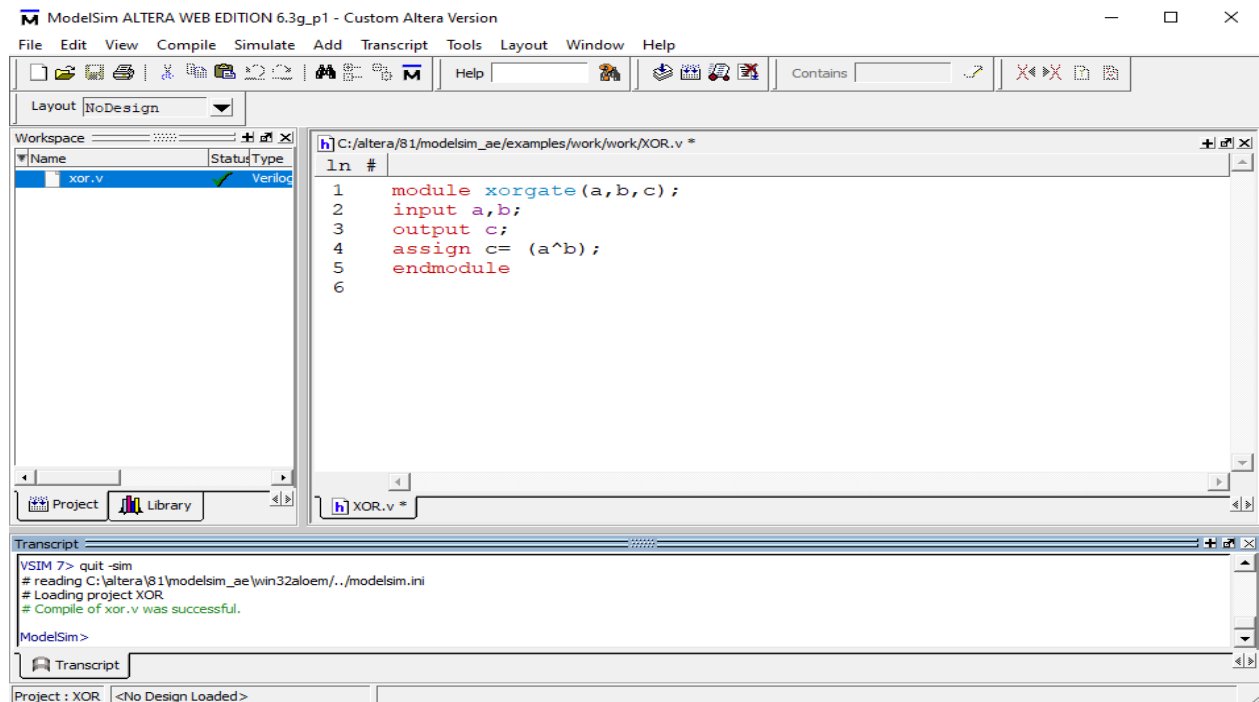


Fig:5.5 : Verilog source code of 2-input XOR gate

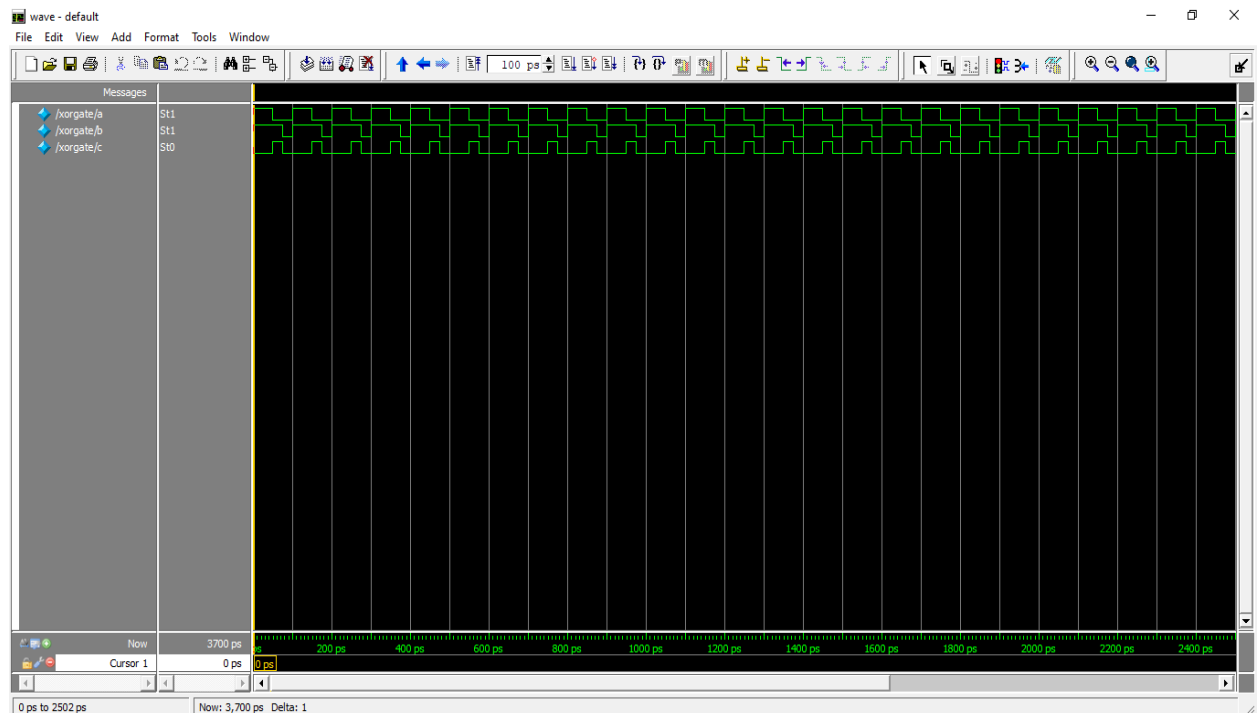


Fig:5.6 : Simulation of 2-input XOR gate

5.2.2 Source code and simulation of 2-input XNOR gate

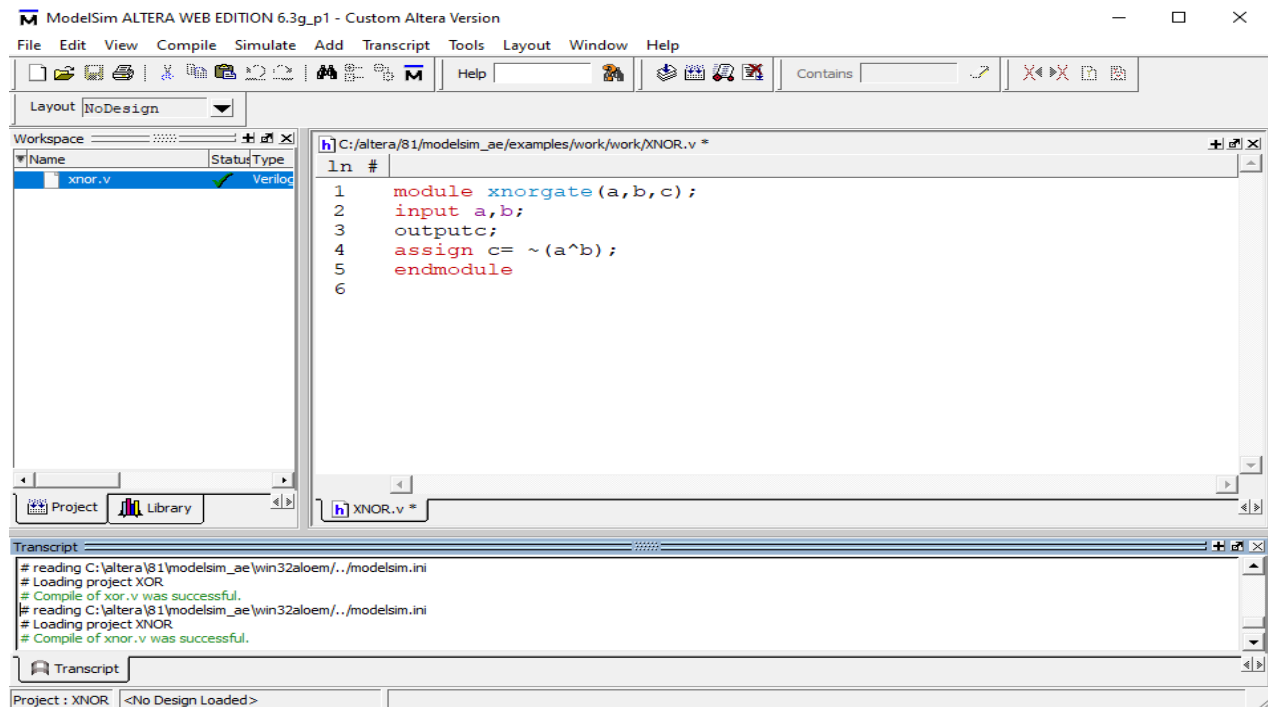


Fig:5.7 : Verilog source code of 2-input XNOR gate

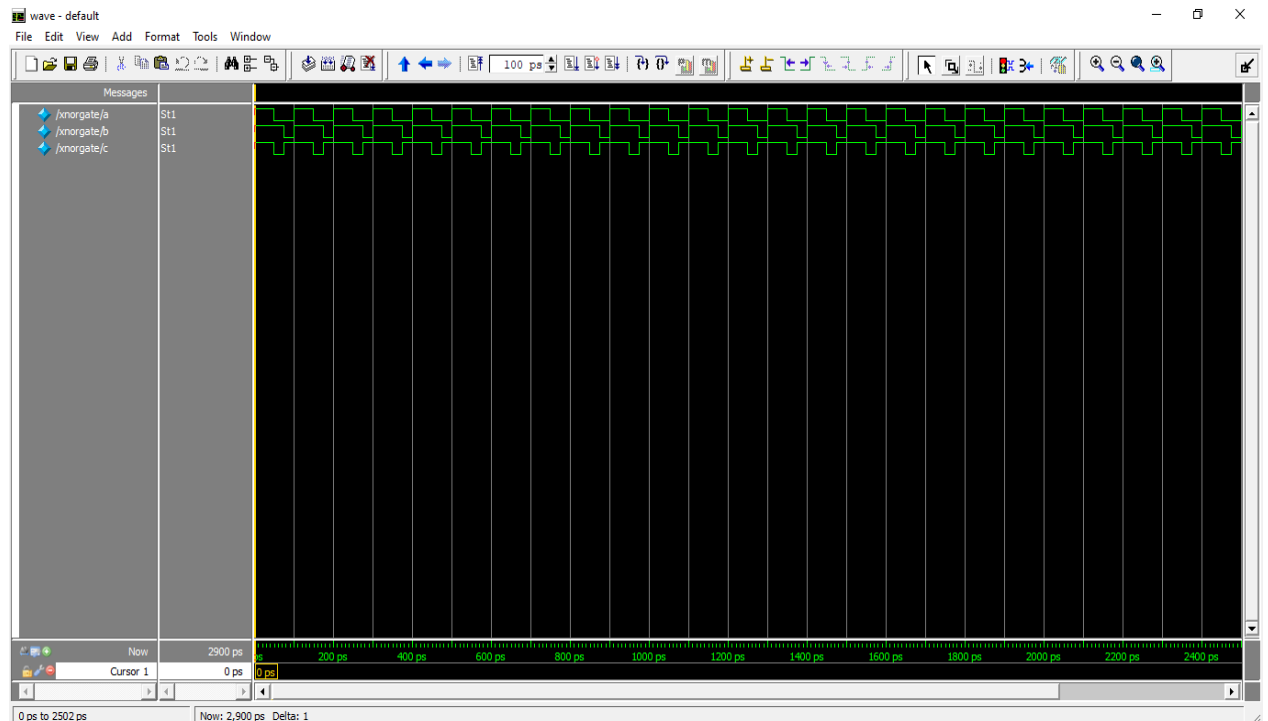


Fig:5.8 : Simulation of 2-input XNOR gate

5.2.3 Source code and simulation of 4x1 Multiplexer

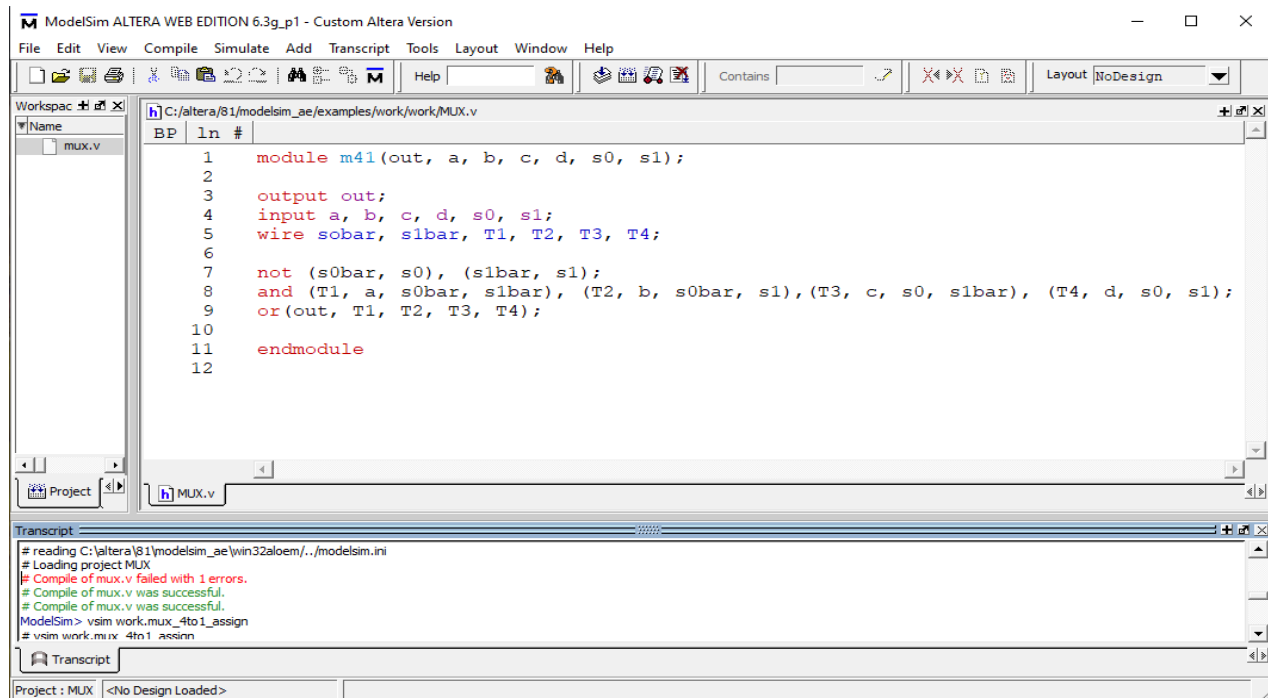


Fig:5.9 : Verilog source code of 4x1 Multiplexer

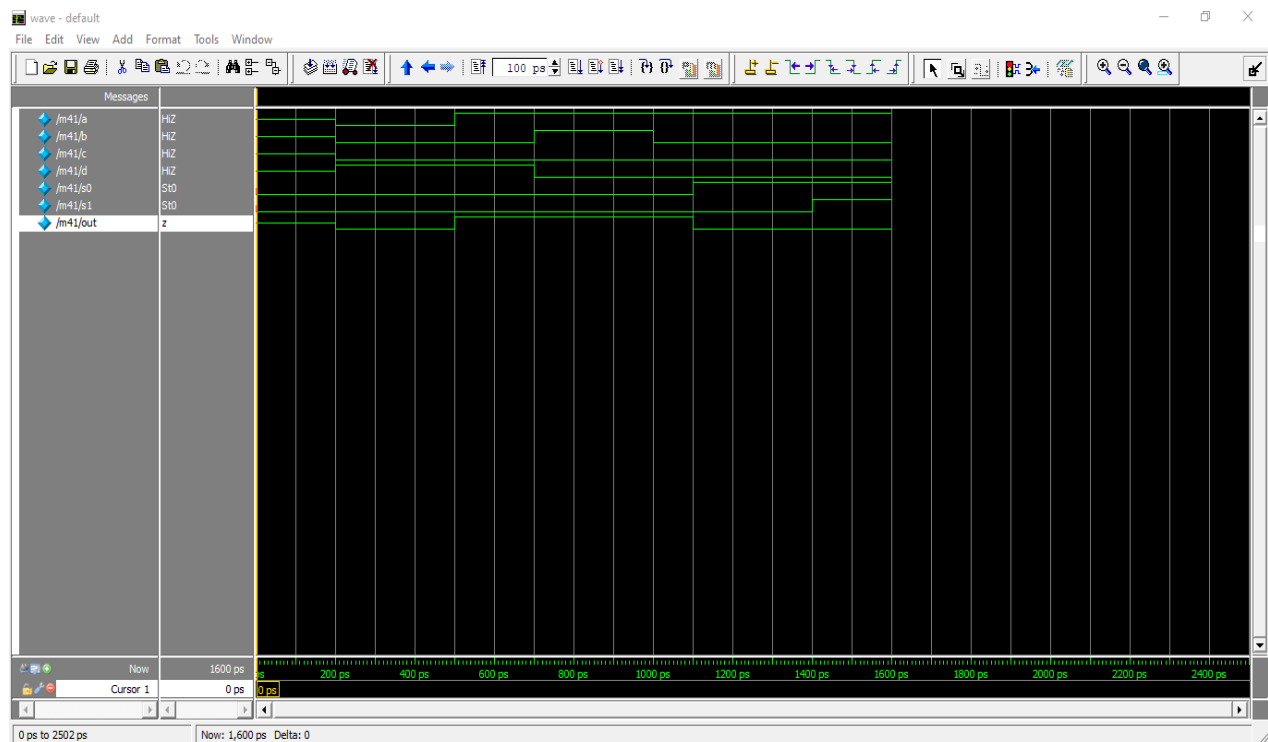
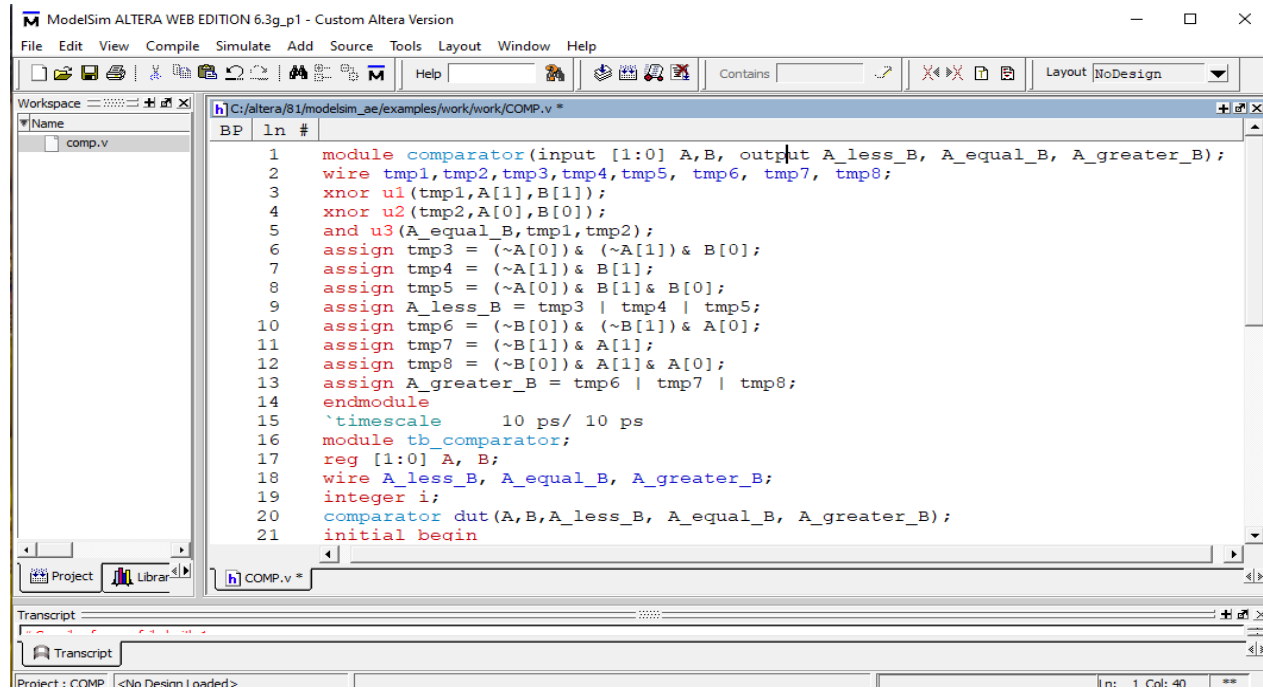


Fig:5.10 : Simulation of 4x1 Multiplexer

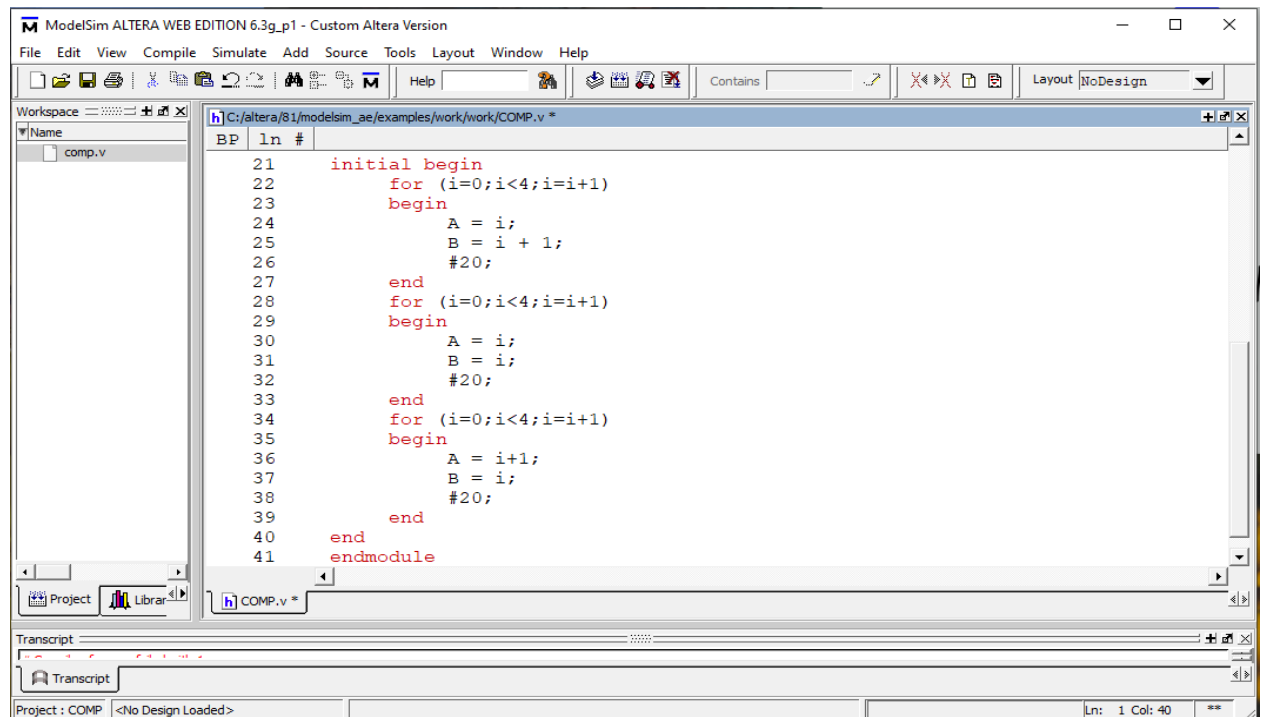
5.2.4 Source code and simulation of 2-bit magnitude comparator



```
ModelSim ALTERA WEB EDITION 6.3g_p1 - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Window Help

h C:/altera/81/modelsim_ae/examples/work/work/COMP.v *
BP ln #
1 module comparator(input [1:0] A,B, output A_less_B, A_equal_B, A_greater_B);
2 wire tmp1,tmp2,tmp3,tmp4,tmp5, tmp6, tmp7, tmp8;
3 xnor u1(tmp1,A[1],B[1]);
4 xnor u2(tmp2,A[0],B[0]);
5 and u3(A_equal_B,tmp1,tmp2);
6 assign tmp3 = (~A[0]) & (~A[1]) & B[0];
7 assign tmp4 = (~A[1]) & B[1];
8 assign tmp5 = (~A[0]) & B[1] & B[0];
9 assign A_less_B = tmp3 | tmp4 | tmp5;
10 assign tmp6 = (~B[0]) & (~B[1]) & A[0];
11 assign tmp7 = (~B[1]) & A[1];
12 assign tmp8 = (~B[0]) & A[1] & A[0];
13 assign A_greater_B = tmp6 | tmp7 | tmp8;
14 endmodule
15 `timescale 10 ps/ 10 ps
16 module tb_comparator;
17 reg [1:0] A, B;
18 wire A_less_B, A_equal_B, A_greater_B;
19 integer i;
20 comparator dut(A,B,A_less_B, A_equal_B, A_greater_B);
21 initial begin
Transcript
Project: COMP <No Design Loaded> ln: 1 Col: 40 **
```

Fig:5.11 : Verilog source code of 2-bit magnitude comparator



```
ModelSim ALTERA WEB EDITION 6.3g_p1 - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Window Help

h C:/altera/81/modelsim_ae/examples/work/work/COMP.v *
BP ln #
21 initial begin
22     for (i=0;i<4;i=i+1)
23     begin
24         A = i;
25         B = i + 1;
26         #20;
27     end
28     for (i=0;i<4;i=i+1)
29     begin
30         A = i;
31         B = i;
32         #20;
33     end
34     for (i=0;i<4;i=i+1)
35     begin
36         A = i+1;
37         B = i;
38         #20;
39     end
40 end
41 endmodule
Transcript
Project: COMP <No Design Loaded> ln: 1 Col: 40 **
```

Fig:5.12 : Verilog source code of 2-bit magnitude comparator (cont)

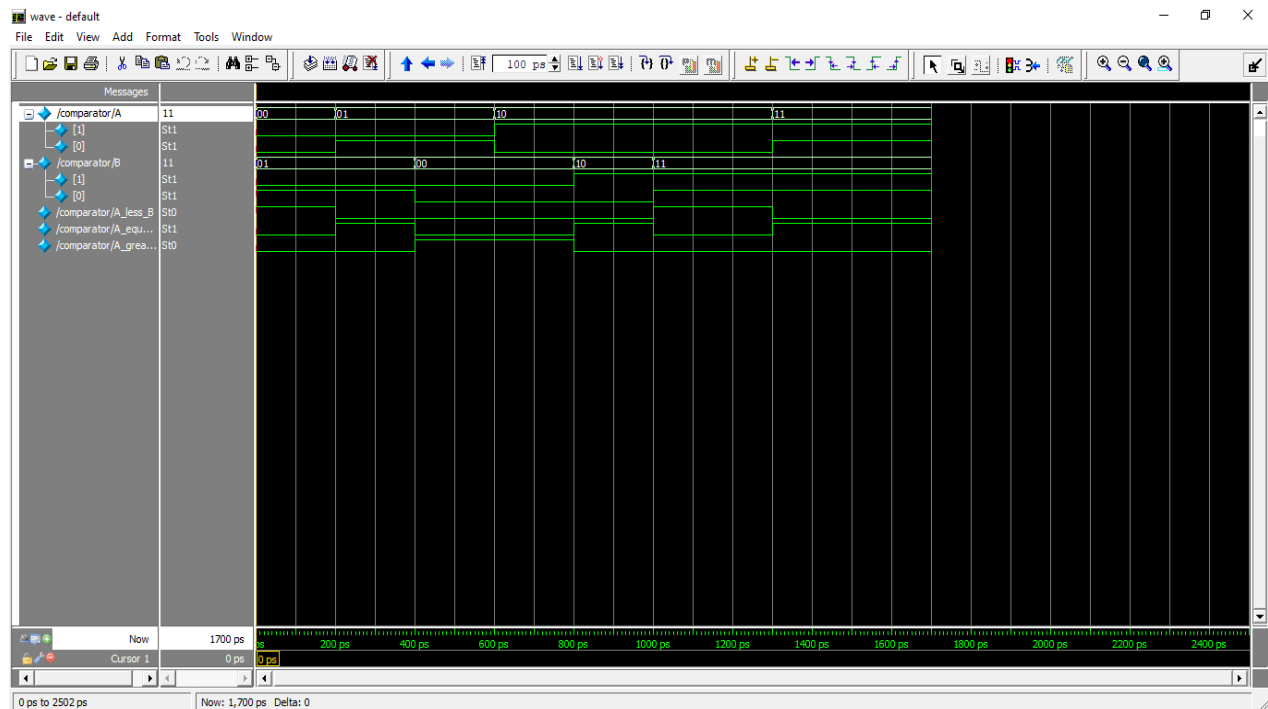


Fig:5.13 : Simulation of 2-bit magnitude comparator

5.2.5 Source code and simulation of D-Flip Flop

```

1  module d_ff(d,clk,clear,q,qbar);
2  input d, clk, clear;
3  output reg q, qbar;
4  always@(posedge clk or posedge clear)
5  begin
6  if(clear== 1)
7  begin
8  q <= 0;
9  qbar <= 1;
10 end
11 else
12 begin
13 q <= d;
14 qbar = !d;
15 end
16 end
17 endmodule
18

```

Transcript

```

# Compile of mux.v failed with 1 errors.
# Compile of mux.v was successful.

```

Project : DFF <No Design Loaded>

Fig:5.14 : Verilog source code of D-Flip Flop

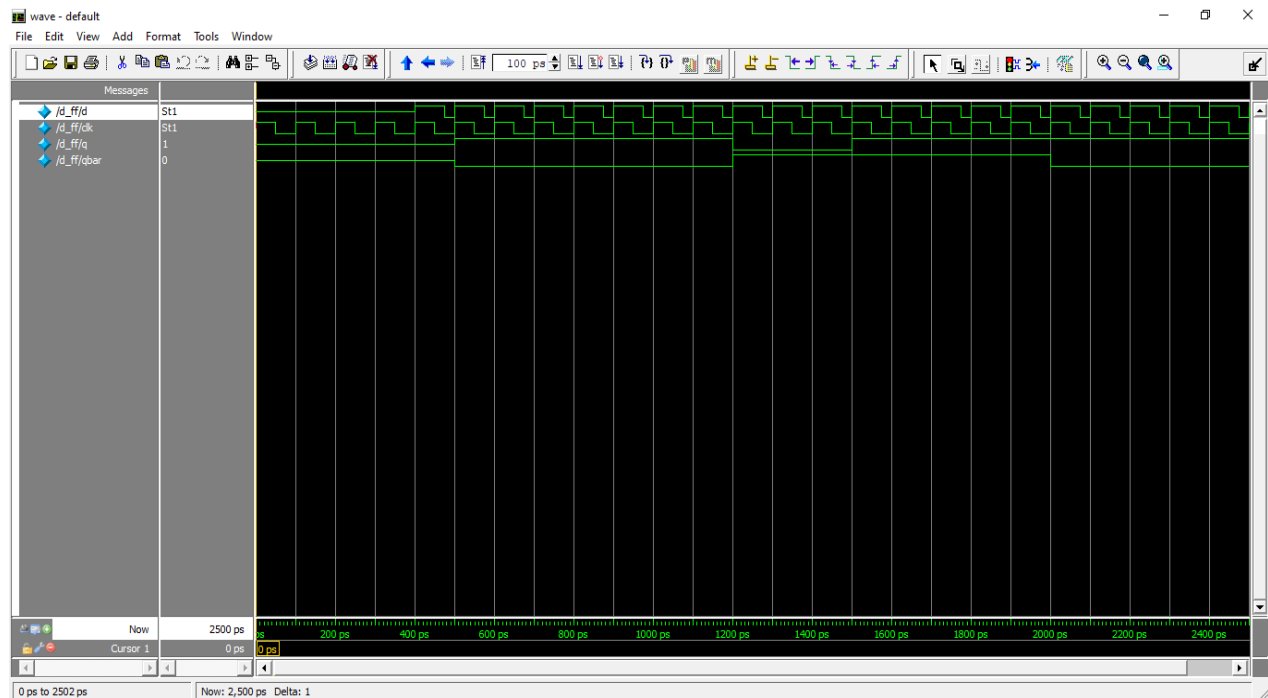


Fig:5.15 : Simulation of D-Flip Flop

5.3 SOURCE CODE AND SIMULATION OF EXTERNAL BLOCK DESIGN OF THE PROPOSED FT-LFSR

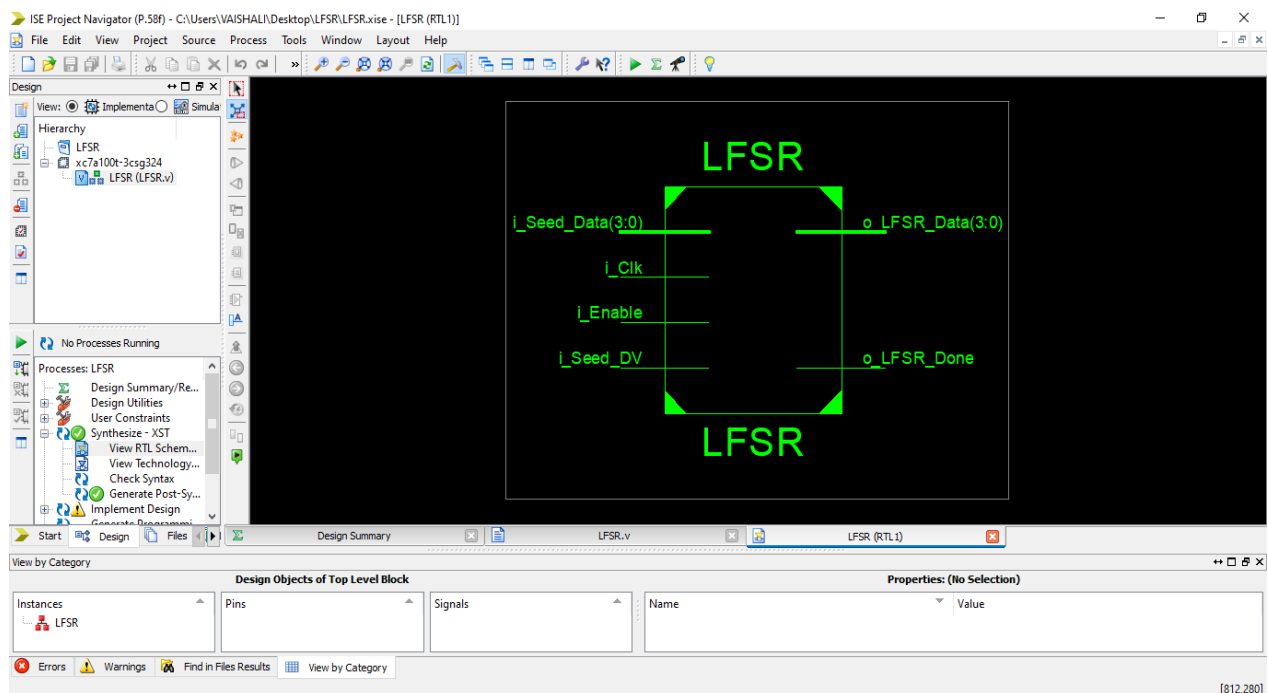


Fig:5.16 : External block design of the proposed FT-LFSR

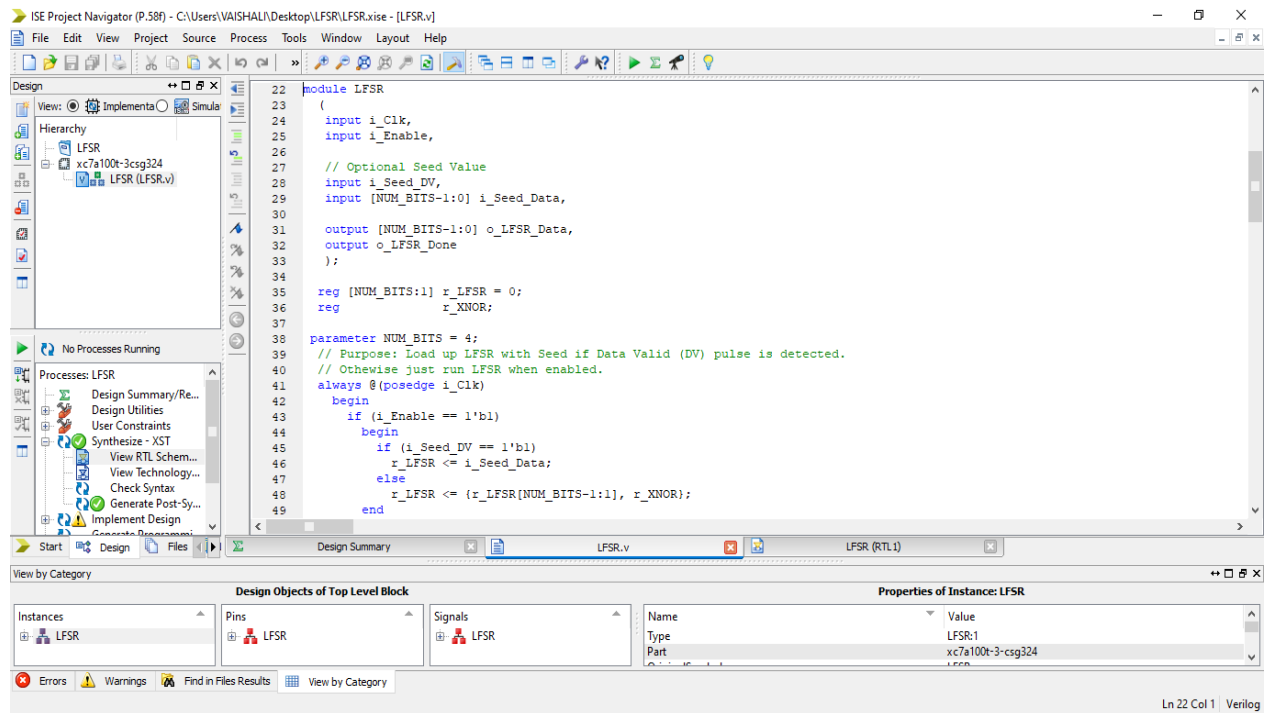


Fig:5.17 : Verilog source code of the proposed FT-LFSR

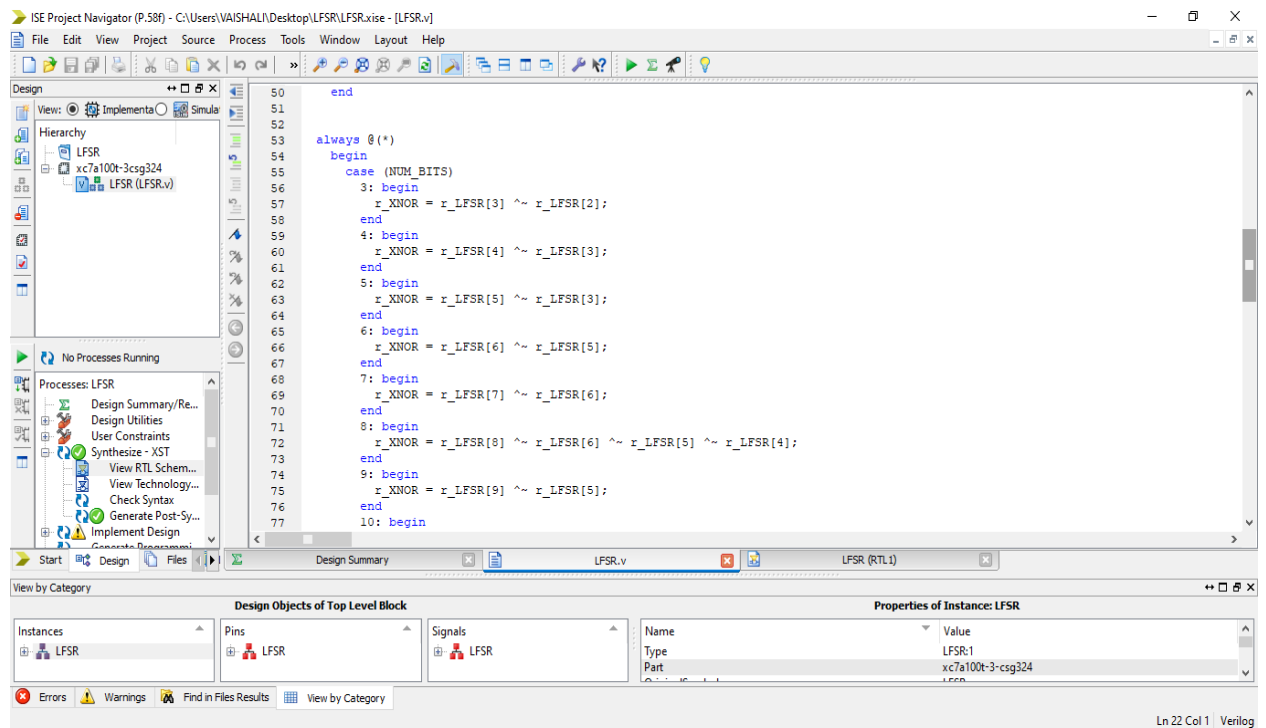


Fig:5.18 : Verilog source code of the proposed FT-LFSR (cont-1)

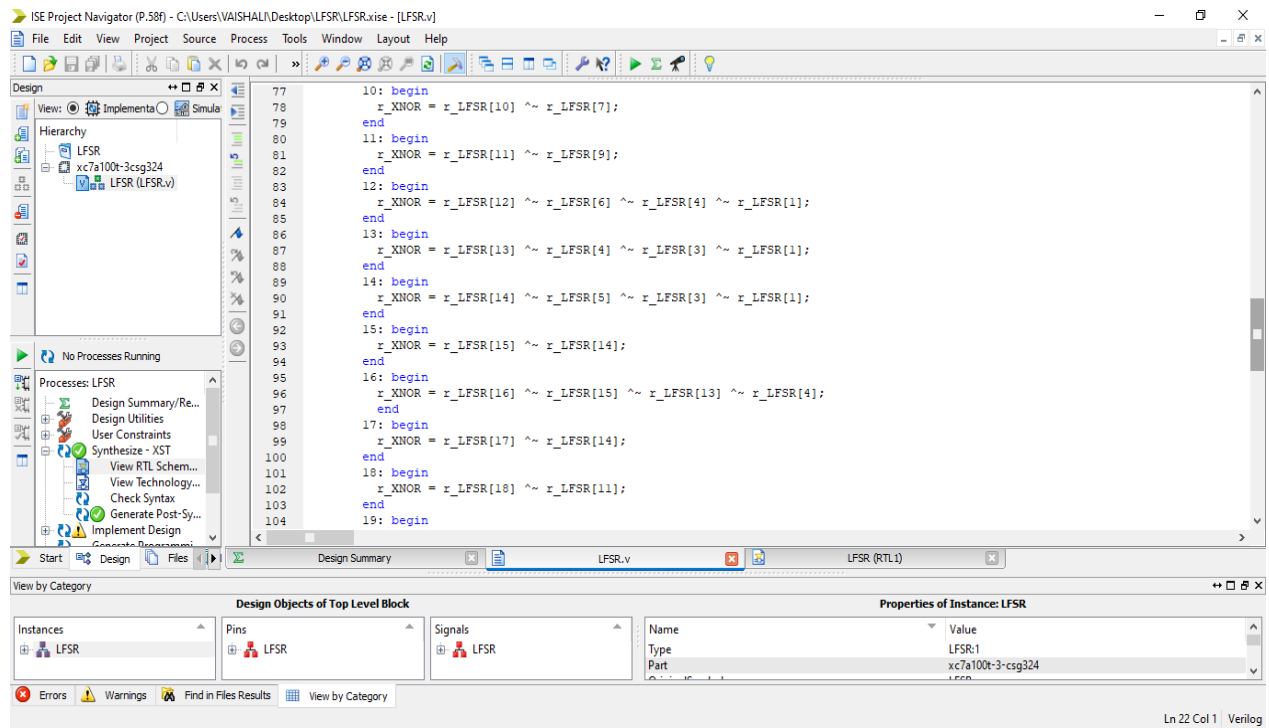


Fig:5.19 : Verilog source code of the proposed FT-LFSR (cont-2)

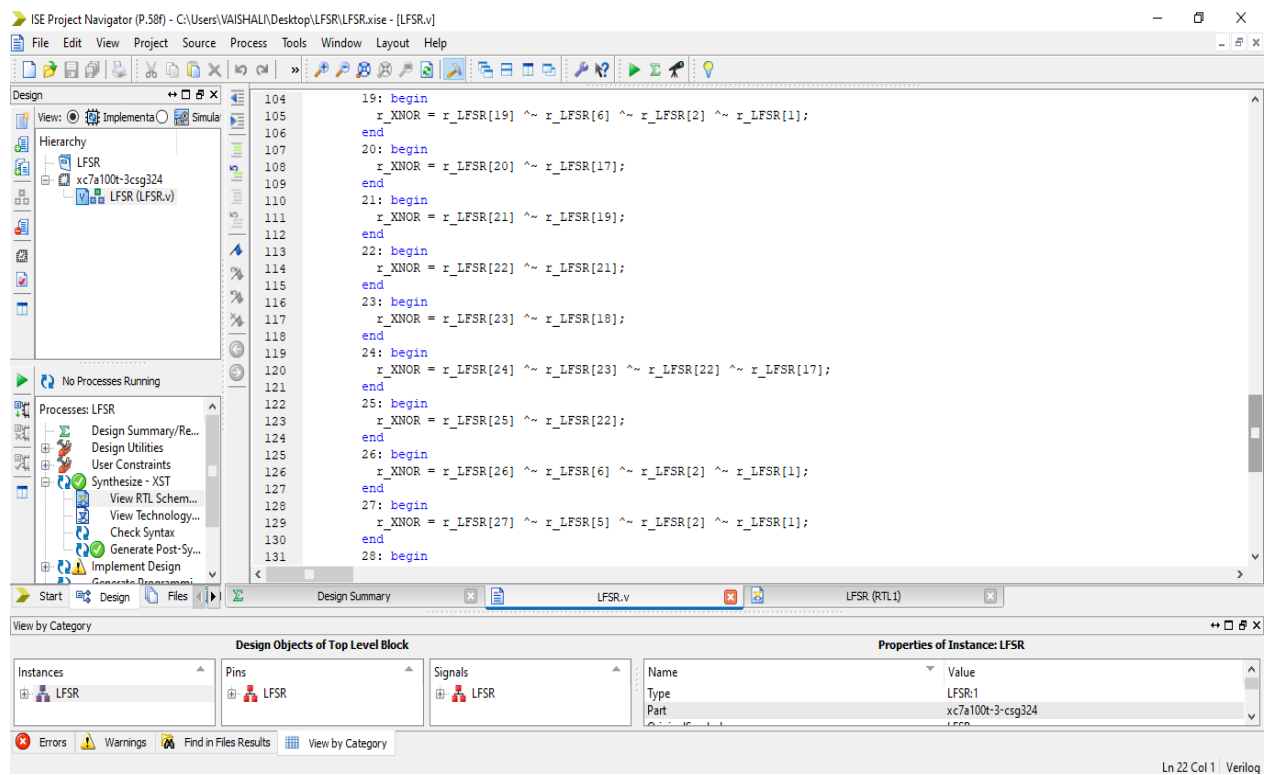


Fig:5.20 : Verilog source code of the proposed FT-LFSR (cont-3)

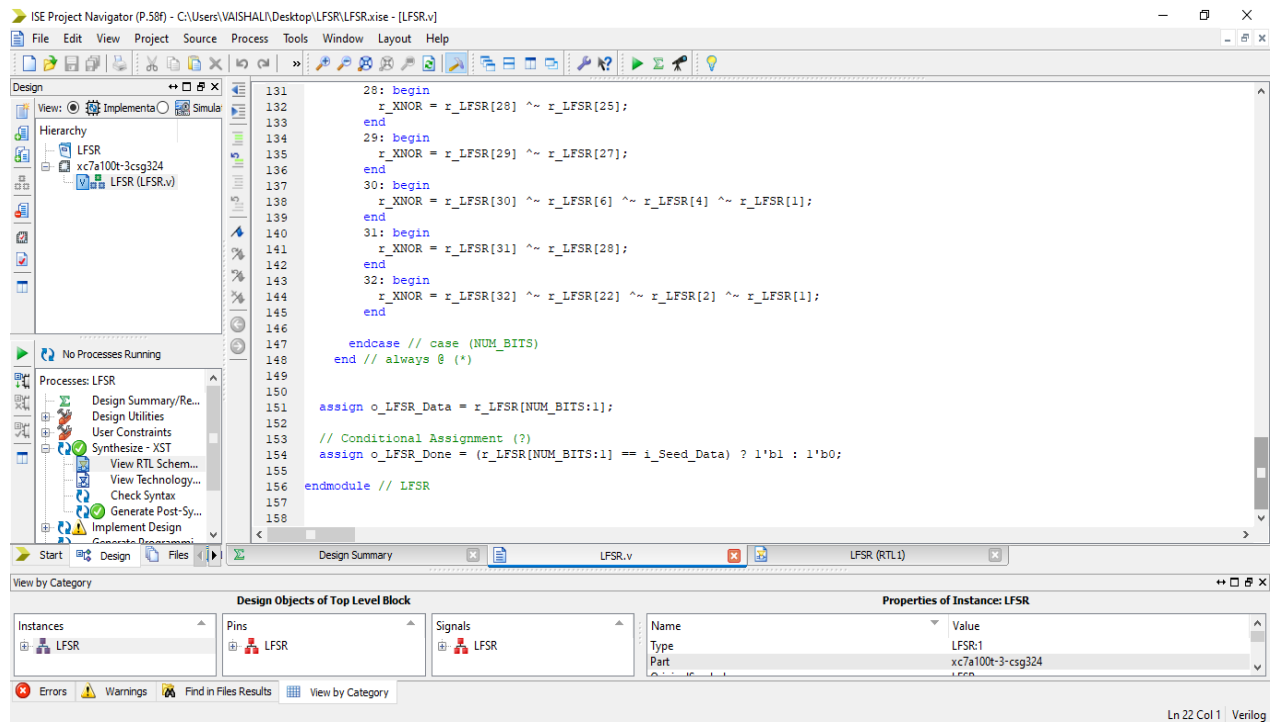


Fig:5.21 : Verilog source code of the proposed FT-LFSR (cont-4)

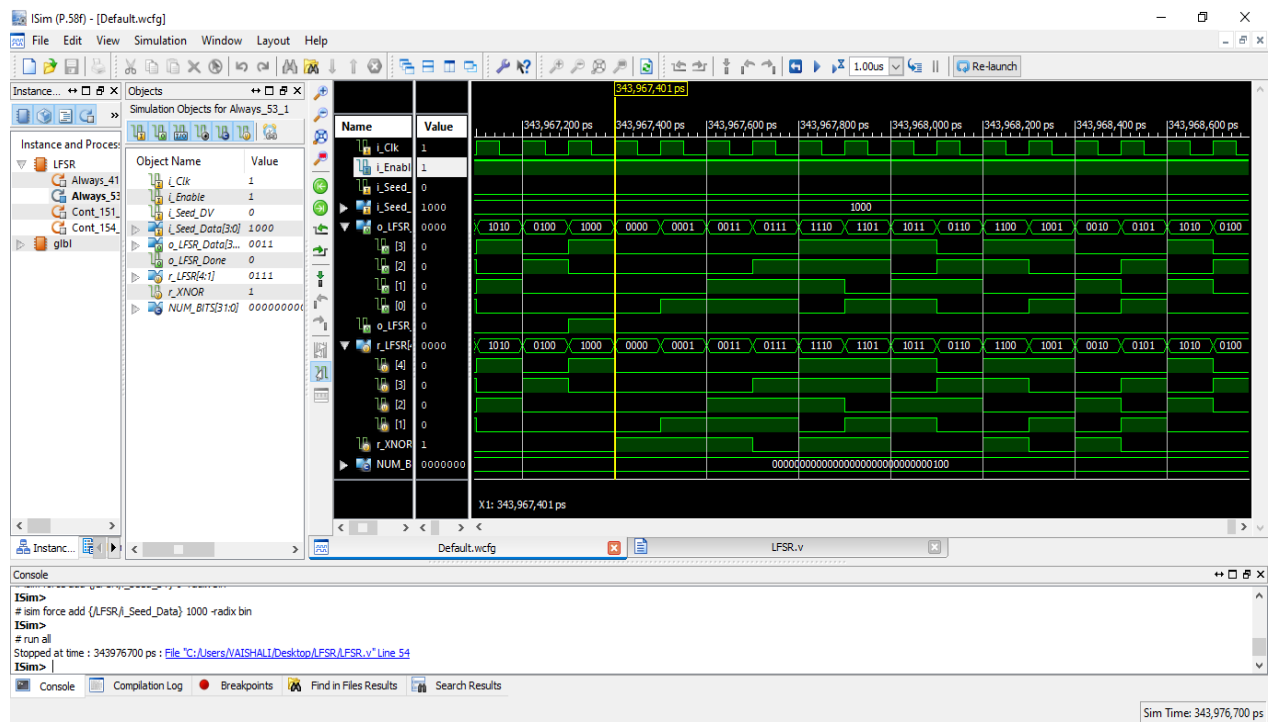


Fig:5.22 : Simulation of the proposed FT-LFSR, when seed data=1000

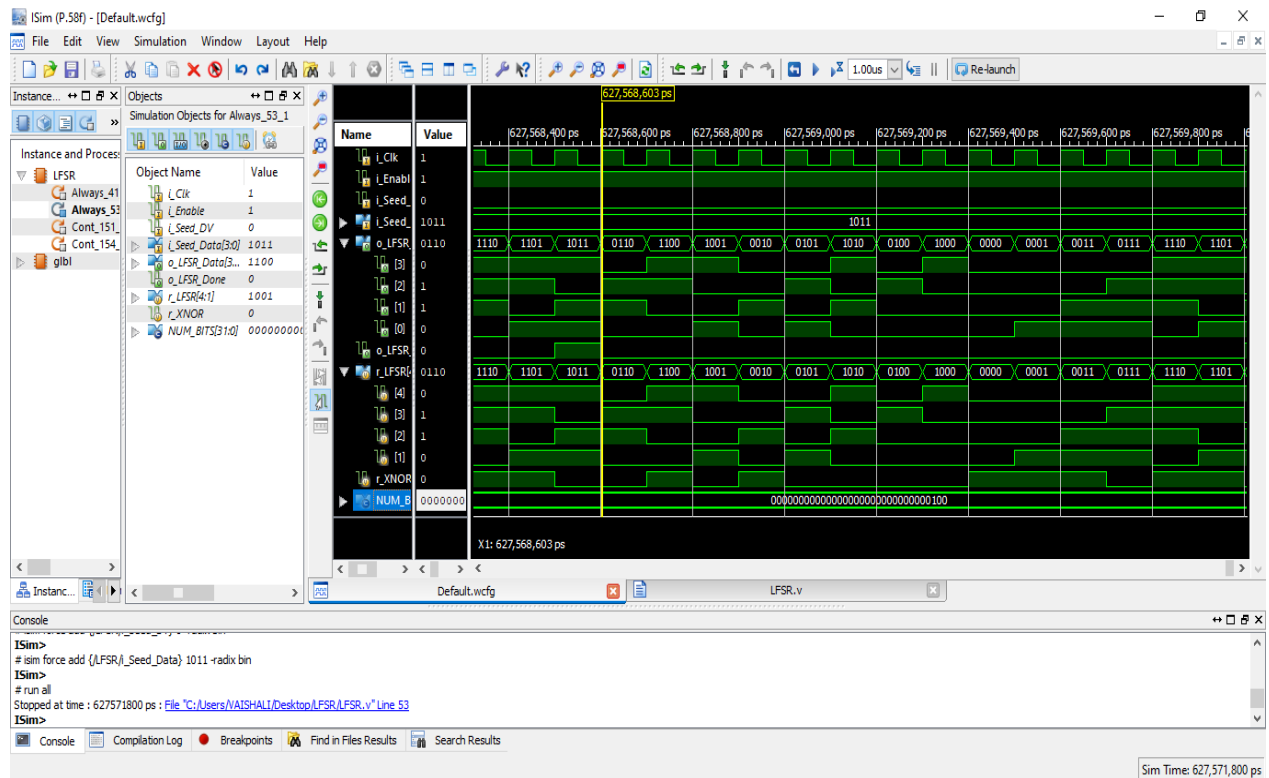


Fig.5.23 : Simulation result of the proposed FT-LFSR, when seed data=1011

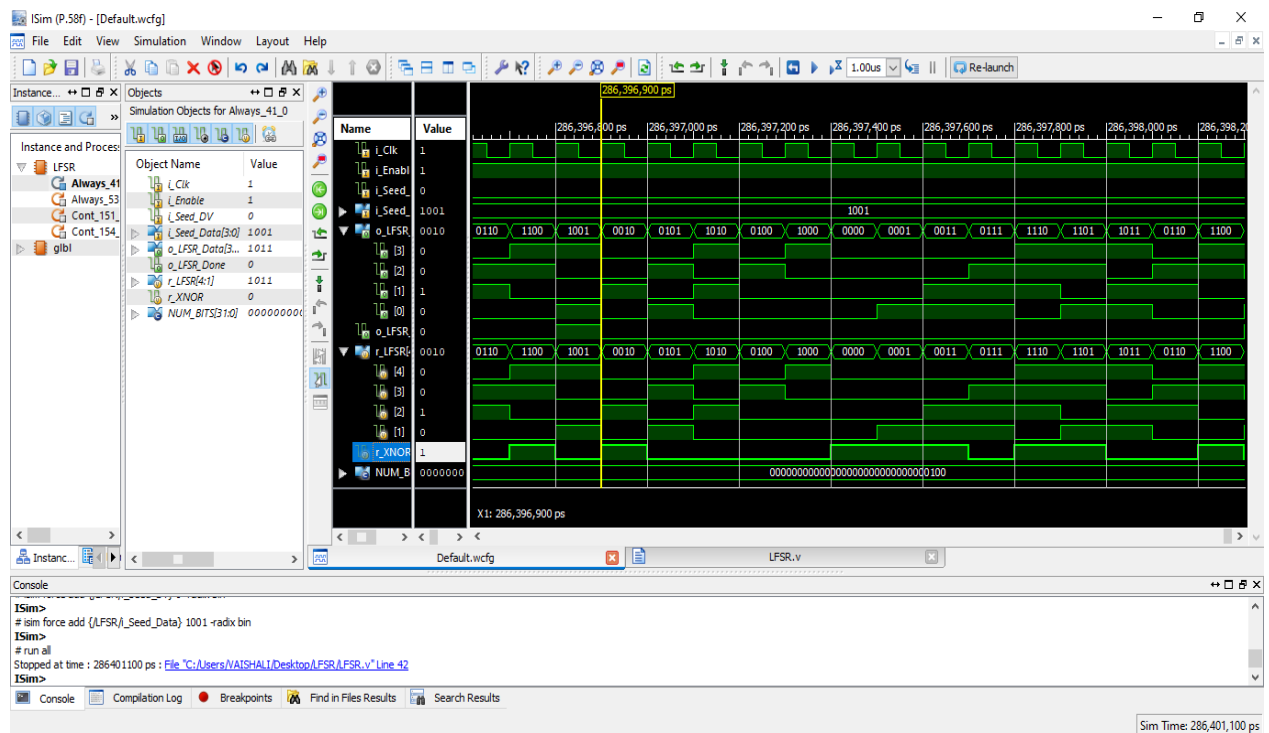


Fig.5.24 : Simulation result of the proposed FT-LFSR, when seed data=1001

5.4 POWER CONSUMPTION AND CLOCK PERIOD OF THE PROPOSED FT-LFSR

Power Analysis : The power consumption of the proposed FT-LFSR is very less when compared to the existing LFSR. The power analysis is done using XILINX software.

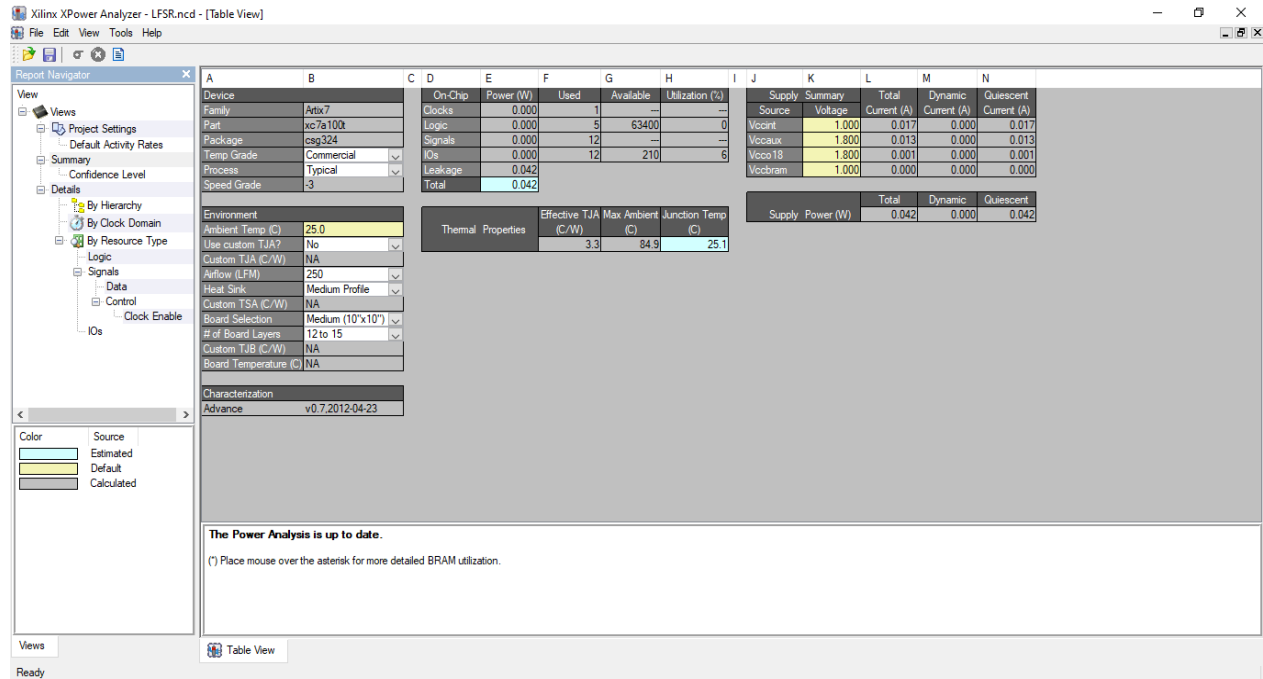


Fig.5.25 : Power Analysis of the proposed FT-LFSR

Clock Period: One of the important metrics to compare different designs is power. We expect that power consumption of FT-LFSR will be less than P-FT-LFSR. It is because the number of logics in FT-LFSR is less than P-LFSR based on different configurations. For a fair comparison all implementations for power are on Virtex 4. As it can be seen for 16 and 32 bit widths, both static and dynamic powers for FT-LFSR are less than PLFSR.

Table 5.1 Clock period of the Proposed and Existing LFSR

LFSR width	clock period		
	FT-LFSR	Conventional LFSR	P-LFSR
16	8.591	1.273	5-10
32	8.941	1.273	5-10
64	9.468	1.273	5-10

5.5 CALCULATION OF AREA, DELAY AND TIMING OF THE PROPOSED FT-LFSR

Area Required:

Table:5.2 : Area Analysis of the proposed FT-LFSR

Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	4	126,800	1%
Number used as Flip Flops	4		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	5	63,400	1%
Number used as logic	5	63,400	1%
Number using O6 output only	4		
Number using O5 output only	0		
Number using O5 and O6	1		
Number used as ROM	0		
Number used as Memory	0	19,000	0%
Number used exclusively as route-thrus	0		
Number of occupied Slices	2	15,850	1%
Number of LUT Flip Flop pairs used	5		
Number with an unused Flip Flop	1	5	20%
Number with an unused LUT	0	5	0%
Number of fully used LUT-FF pairs	4	5	80%
Number of unique control sets	1		

Timing in nanoseconds:

Data Sheet report:

All values displayed in nanoseconds (ns)

Setup/Hold to clock i_Clk

Source	Max Setup to clk (edge)	Process Corner	Max Hold to clk (edge)	Process Corner	Internal Clock(s)	Clock Phase
i_Enable	-0.524 (R)	FAST	2.408 (R)	SLOW	i_Clk_BUFGP	0.000
i_Seed_DV	-0.444 (R)	FAST	2.520 (R)	SLOW	i_Clk_BUFGP	0.000
i_Seed_Data<0>	-0.404 (R)	FAST	2.268 (R)	SLOW	i_Clk_BUFGP	0.000
i_Seed_Data<1>	-0.431 (R)	FAST	2.307 (R)	SLOW	i_Clk_BUFGP	0.000
i_Seed_Data<2>	-0.334 (R)	FAST	2.162 (R)	SLOW	i_Clk_BUFGP	0.000
i_Seed_Data<3>	-0.481 (R)	FAST	2.399 (R)	SLOW	i_Clk_BUFGP	0.000

Fig:5.26 : Setup to clock Timing of the proposed FT-LFSR

Clock i_Clk to Pad							
Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase	
o_LFSR_Data<0>	7.550(R)	SLOW	3.120(R)	FAST	i_Clk_BUF	0.000	
o_LFSR_Data<1>	7.701(R)	SLOW	3.205(R)	FAST	i_Clk_BUF	0.000	
o_LFSR_Data<2>	7.587(R)	SLOW	3.143(R)	FAST	i_Clk_BUF	0.000	
o_LFSR_Data<3>	7.672(R)	SLOW	3.169(R)	FAST	i_Clk_BUF	0.000	
o_LFSR_Done	8.589(R)	SLOW	3.285(R)	FAST	i_Clk_BUF	0.000	

Fig:5.27 : Clock to Pad Timing of the proposed FT-LFSR

Clock to Setup on destination clock i_Clk				
Source Clock	Dest:Rise	Dest:Fall	Dest:Rise	Dest:Fall
i_Clk	0.787			

Fig:5.28 : Clock to Setup Timing of the proposed FT-LFSR

Delay:

Pad to Pad		
Source Pad	Destination Pad	Delay
i_Seed_Data<0>	o_LFSR_Done	5.769
i_Seed_Data<1>	o_LFSR_Done	5.859
i_Seed_Data<2>	o_LFSR_Done	5.380
i_Seed_Data<3>	o_LFSR_Done	5.287

Fig:5.29 : Pad to Pad Delay of the proposed FT-LFSR

5.6 COMPARISON OF THE EXISTING AND THE PROPOSED FT-LFSR

Table:5.3 : Comparison of the existing and the proposed design

	POWER	DELAY(ns)	SLICE	TIMING CONSTARINT (ns)
EXISTING DESIGN	0.723	6.934	LUT: 8 FLIP-FLOPS: 4 IOB : 23	SETUP TO CLOCK: 5.389 CLOCK TO TIMING: 6.159 CLOCK TO SETUP: 1.521
PROPOSED DESIGN	0.042 (37%)	5.287 (32%)	LUT: 5 FLIP-FLOPS: 4 IOB : 11	SETUP TO CLOCK: 2.399 CLOCK TO TIMING: 3.285 CLOCK TO SETUP: 0.787

The power consumption plays a vital role in energy saving and the increase in speed provides better system performance. The major performance parameters such as power, hardware utilization, delay and timing constraint of FT-LFSR architecture are simulated and compared with the existing one.

In proposed design 37% of power is reduced and improved speed of 32% when compared with the existing LFSR. So, the proposed LFSR is more efficient than the existing LFSRs in all other ways.

CHAPTER 6

CONCLUSION

Linear feedback shift registers play very important role in many fields like for testing, cryptography, BCH, CRC encoders, BIST, Pseudo Random sequences . In cryptography multibit LFSR is used because it is more secure as compare to conventional LFSR. Traditional fault tolerant LFSRs include large number of Single-Point-of-Failures (SPoFs) in which any fault results in the whole system failure. To overcome that single point failures a new fault tolerant architecture is proposed for LFSR named as Fault Tolerant-Linear Feedback Shift Register (FT-LFSR) in which the number of SPoFs are significantly reduced. To this end, a modified version of Triple Modular Redundancy (TMR) empowered with some extra controlling units for identifying the operational module is used in this project. The proposed approach shows the concept of reducing the transitions in the test pattern generated. The transition is reduced by increasing the correlation between the successive bits. The simulation results shows that how the patterns are generated for the applied seed vector. The LFSR is a shift register that has some of its outputs together in exclusive-OR configurations to form a feedback path. LFSRs are frequently used as pseudorandom pattern generators to generate a random number of 1s and 0s. Each output of the LFSR is multiplexed with an ASIC input and, when the device is placed in the LFSR (test) mode, the random, high-toggle-rate patterns produced are extremely good for generating high-fault coverage. This project presents a new low-power LFSR to reduce the average and peak power of combinational and sequential circuits during the test mode. It is observed that the total power consumed in proposed FT-LFSR is less than the power consumed with normal LFSR and output dynamic power is decreased. This is with almost no increase in test length to hit a target fault coverage. FT-LFSR significantly reduces the instantaneous power violation compared to the LFSR and thus avoids putting stress on the circuit during test which increases reliability. It is concluded that low power LFSR is very useful for BIST implementation in which the CUT may be Combinational, sequential and memory circuits.

REFERENCES

- [1] Parhi, K.K., 2004, Eliminating the fanout bottleneck in parallel long BCH encoders, IEEE Transactions on Circuits and Systems I: Regular Papers, 51(3), pp.512-516.
- [2] Kennedy, C. and Reyhani-Masoleh, A., 2009, June. High-speed CRC computations using improved state-space transformations, In Electro/Information Technology, 2009. eit'09. IEEE International Conference on (pp. 9-14). IEEE.
- [3] Ayinala, M. and Parhi, K.K., 2010, October, Efficient parallel VLSI architecture for linear feedback shift registers, In Signal Processing Systems (SIPS), 2010 IEEE Workshop on (pp. 52-57). IEEE.
- [4] Ayinala, M. and Parhi, K.K., 2011, High-speed parallel architectures for linear feedback shift registers, IEEE transactions on signal processing, 59(9), pp.4459-4469.
- [5] Muthiah, D. and Raj, A.A.B., 2012, February, Implementation of high-speed LFSR design with parallel architectures, In Computing, Communication and Applications (ICCCA), 2012 International Conference on (pp. 1-6). IEEE.
- [6] Jung, J., Yoo, H., Lee, Y. and Park, I.C., 2015, Efficient parallel architecture for linear feedback shift registers, IEEE Transactions on Circuits and Systems II: Express Briefs, 62(11), pp.1068-1072.
- [7] Hu, G., Sha, J. & Wang, Z., 2017, High-Speed Parallel LFSR Architectures Based on Improved State-Space Transformations, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(3), pp 1159-1163.
- [8] Datta, D., Datta, B. and Dutta, H.S., 2017, March. Design and implementation of multibit LFSR on FPGA to generate pseudorandom sequence number, In Devices for Integrated Circuit (DevIC), 2017 (pp. 346-349). IEEE.
- [9] Kumar, A., Saraswat, S. and Aggarwal, T., 2017, July, Design of 4-bit LFSR on FPGA, In Computing, Communication and Networking Technologies (ICCCNT), 2017 International Conference on (pp. 1-6).IEEE.

- [10] Madhushree, K. and Rajan, N., 2017, March, Dynamic power optimization of LFSR using clock gating, In Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017 International Conference on (pp. 1-4). IEEE.
- [11] Thubrikar, T., Kakde, S., Gaidhani, S., Kamble, S. and Shah, N., 2017, April, Design and implementation of low power test pattern generator using low transitions LFSR, In Communication and Signal Processing (ICCSP), 2017 International Conference on (pp. 0467-0471). IEEE.
- [12] Devika, K.N. and Bhakthavatchalu, R., 2017, April, Design of reconfigurable LFSR for VLSI IC testing in ASIC and FPGA, In Communication and Signal Processing (ICCSP), 2017 International Conference on (pp. 0928-0932). IEEE.
- [13] Leonard Colatitude and Dennis Silage, "Efficient PGA LFSR Implementation Whitens Pseudorandom Numbers, 2009 International Conference on Reconfigurable Computing and FPGAs, 2009
- [14] Nagaraj s vannal, saroja v siddamal, shruti v bidaralli, mahalaxmi s bhille, Design and testing of combinational Logic circuits using built in self Test scheme for fpgas, 2015 fifth international conference on communication systems and network technologies, 2015.
- [15] Cannon, Matthew & Keller, Andrew & Thurlow, Corbin & Perez-Celis, Andres & Wirthlin, Michael. (2019). Improving the Reliability of TMR with Non-Triplicated I/O on SRAM FPGAs. IEEE Transactions on Nuclear Science. PP. 1-1. 10.1109/TNS.2019.2956473.
- [16] V. Petrovi, Z. Stamenkovi, M. Stojev, T. Nikoli, and G. Jovanovi, "FaultTolerant Reconfigurable Low-Power Pseudorandom Number Generator," IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Karlovy Vary Czech Republic, pp. 279-282, April 2013