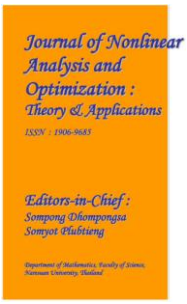
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1 : 2024
ISSN : **1906-9685**



FPGA IMPLEMENTATION OF EFFICIENT AND LOW POWERTEST PATTERN GENERATOR

D.Murali, M. Kavitha, N. Deepika, M. Sudha, P. Krishna Babu, Department of Electronics & Communication Engineering, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212

Abstract: Given the multiple challenges posed by testing VLSI circuits in terms of space overhead, power, and speed, the VLSI circuits must be put to the test. Generations of Low Power LFSR test patterns are one method for testing a complicated architecture of VLSI Circuits. VLSI circuits need to be tested both before and after construction, just like other electronic components do. A 32-bit Low Power-LFSR (LP-LFSR) is suggested in this paper as a means of evaluating the VLSI design. The standard LFSR and an extra combinational network are used in the construction of this 32-bit test pattern generator to achieve minimal power consumption. Using VIVADO version 2018, the LP-LFSR's design and verification were completed in Verilog HDL. Because there is less switching between the test vectors, there is less power usage. 16 mW of electricity is produced by the test pattern generator design. The production of 32-bit test patterns requires a high level of switching activity, which is improved in this research. The experimental findings demonstrate that a low power linear feedback shift register has a dynamic power that is 67.34 percent lower than a conventional LFSR.
Keywords: Test Pattern Generator (TPG), Linear Feedback Shift Register (LFSR), VLSI Testing, Low Transition LFSR, Switching activity.

1.Introduction

This paper proposes a method to extend the period of a linear feedback shift register (LFSR) by introducing an algorithm to generate primitive polynomials. This is achieved by utilizing a basic LFSR with a maximum period equal to a prime number. The proposed method achieves a statistically robust period extension with a very long extension of the LFSR period, reaching up to (2^120)!(2N - 1) for a 127-bit length register. By separating the setup and running phases in the algorithm, the speed characteristic of the LFSRs is preserved. Uniform Random Number Generators (URNGs) are crucial in many applications running on FPGA-based hardware accelerators. Typically, multiple LFSRs are required in a URNG because a normal LFSR can only generate one bit per cycle. This paper introduces a new type of URNG using the Leap-Ahead LFSR Architecture, which can generate an m-bits random number per cycle using only one LFSR. The architecture is analyzed, the period expression is presented, and guidelines for choosing the LFSR taps are provided. An 18-bit URNG is implemented on Xilinx Vertex IV FPGA, demonstrating that the Leap-Ahead LFSR Architecture URNG consumes less than 40 slices, which is only 10% of what the Multi-LFSRs architecture consumes. It also achieves very good Area Time performance and Throughput performance, which are 2.18×10-9 slices×sec per bit and 17.87×10^9 bits per sec, respectively. This paper directly compares a fast binary counter, built using a hierarchical Manchester carry chain, with a counter built using an LFSR. The comparison focuses on speed, power, and area consumption, demonstrating the use of LFSRs as an alternative to conventional binary event counters. To use an LFSR as a counter, an efficient algorithm is developed for decoding the pseudo-random bit patterns of the LFSR counter into a known binary count. 4-bit, 8-bit, 16-bit, and 32-bit LFSR and binary counters are implemented in a 0.5-µm CMOS process. Simulation and measurement results validate the hypothesis that LFSR counters lead to reduced area and higher speed. LFSR-based PN Sequence Generators are used for various cryptography applications and for designing encoders and decoders in different communication channels. It is essential to test and verify these applications by implementing them on hardware to achieve better efficiency. FPGAs

are used to implement logical functions for faster prototype development, making it necessary to implement existing LFSR designs on FPGAs to test and verify the simulated and synthesis results for different lengths. The total number of random states generated by an LFSR depends on the feedback polynomial. As a simple counter, it can count a maximum of 2^n-1 using the maximum feedback polynomial. This paper implements 8-bit, 16-bit, and 32-bit LFSRs on FPGA using VHDL to study the performance and analyze the randomness behavior. The analysis is carried out to determine the number of gates, memory, and speed requirements in FPGA as the number of bits is increased. A comparative study of 8-bit, 16-bit, and 32-bit LFSRs on FPGA is presented to understand on-chip verification. Additionally, the paper presents simulation problems for long-bit LFSRs on FPGA. When testing an integrated circuit, large chip size and excessive power dissipation are major issues. The power dissipation during testing mode is particularly high compared to its working mode. Additionally, the inefficiency of ATE and its time-consuming nature make external testing much more difficult. LFSRs are used for testing ASIC chips, generating pseudo-random variables used in the testing process. Pseudo-random variable testing has advantages, such as using simple hardware for on-chip test generation. BIST is one of the most efficient low-power testing methods, using LFSRs for test pattern generation. This paper compares various architectures of LFSRs for BIST and their associated power dissipation. A Shift Register is a cascade of flip-flops that share the same clock, with outputs connected to the data input of the next flip-flop in the chain. A Linear-Feedback Shift Register (LFSR) is a type of shift register whose input is a linear function of its previous state, commonly using an Exclusive OR (XOR) function. LFSRs are used to generate pseudo-random numbers, fast digital counters, pseudo-noise sequences, and whitening sequences. LFSRs can be realized using both hardware and software. For hardware implementation, the MOS current mode logic (MCML) method can be used to design the LFSR. However, the traditional MCML method has drawbacks, including static power dissipation, higher power consumption at low frequencies compared to CMOS circuits, unsuitability for large systems involving power-down modes, and lack of cost-effectiveness. To overcome these issues and achieve the high-speed characteristics of MCML, the paper presents a modified dynamic current mode logic, which is a good solution for battery-powered systems and portable solutions. Simulation results confirm this, showing that a 16-bit adder circuit fabricated using CMOS technology has only a delay of 1.22 ns and dissipates 19.0 mW at 400 MHz. The increase in the number of edge devices has led to the emergence of edge computing, where computations are performed on the device. Deep Neural Networks (DNNs) have become state-of-the-art methods in various applications, but their large size and computational expense make them unsuitable for power and memory-constrained edge devices. Sparsification techniques have been proposed to reduce the memory footprint of DNN models, but they often lead to substantial hardware and memory overhead. This article proposes a hardware-aware pruning method using Linear Feedback Shift Registers (LFSRs) to generate the locations of non-zero weights in real-time during inference, called LFSR-generated pseudorandom sequence-based sparsity (LGPS) technique. Two architectures are explored: (1) row/column indexing with LFSRs and (2) column-wise indexing with nested LFSRs. Using the proposed method, the paper presents energy and area savings of up to 37.47% and 49.93%, respectively, and a speedup of 1.53× compared to the baseline pruning method, for the VGG-16 network on down-sampled ImageNet. In a commonly used test data compression method, the on-chip decompression logic is based on a linear-feedback shift-register (LFSR), and compressed tests consist of seeds for the LFSR. A seed can be modified and used for applying several different tests, reducing the input test data volume further than the basic test data compression method. This article introduces a new approach for applying several different tests based on every seed, padding a seed in different ways to obtain new seeds for LFSRs with more bits. All of these can be implemented by a single programmable LFSR. Using LFSRs with more bits is effective in detecting more target faults, supporting test compaction with a reduction in the input test data volume. Experimental results for benchmark circuits demonstrate these advantages[1-8].

2. Proposed Method

A Low-Transition Pattern Generation Techniques

2.1Random Injection (RI) Technique

The RI method inserts a new test vector, Ti1, between two existing test vectors to ensure that the total number of transitions between Ti and Ti+1 and Ti and Ti1 matches the transition between Ti and Ti1. This insertion of Ti1 reduces the number of transitions between Ti and Ti+1. When two equal-bit locations in Ti and Ti+1 are identical, a bit is injected into the same position in Ti and Ti+1. For example, in Fig. 1, with R=0, the RI infusion process begins when a transition between Ti and Ti+1 is detected.

$$\sum^{n}_{j=1}|t^{i}_{j} - t^{i1}_{j}| + \sum^{n}_{j=1}|t^{i1}_{j} - t^{i+1}_{j}| = \sum^{n}_{j=1}|t^{i1}_{j} - t^{i+1}_{j}| \quad (1)$$

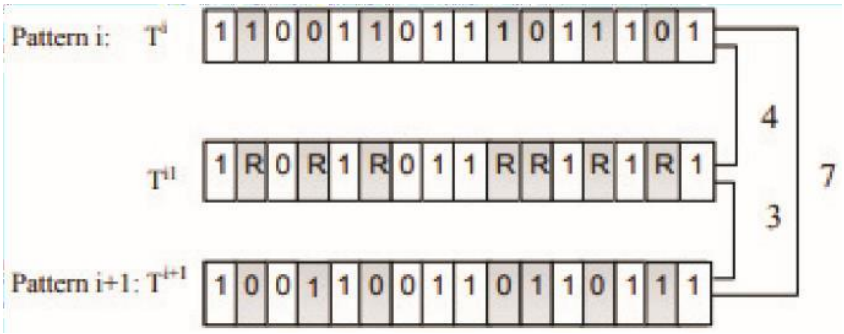


Figure.1 Mid-level pattern

Figure 1 shows an example of a mid-level pattern. The highlighted digit represents the number of transitions between the Ti and Ti+1-bit patterns, or (1=>0), before injection. The number of transitions is reduced to 4 or 3 after injecting a Ti1 bit pattern when R=0 or R=1. The worst-case scenario is estimated to have a maximum of four transitions.

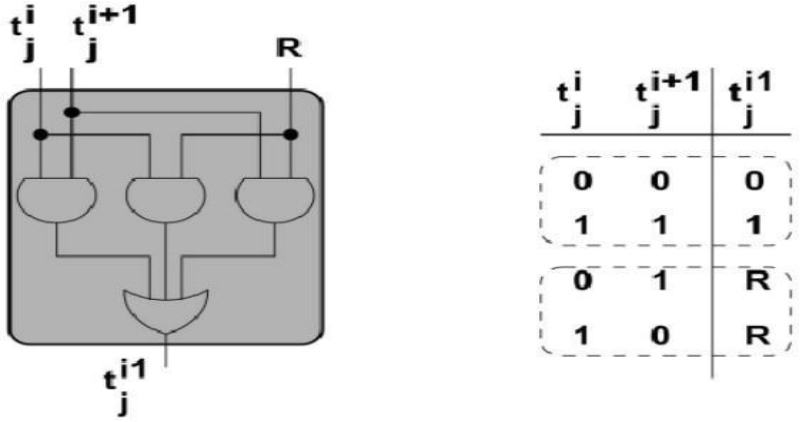


Figure. 2. RI Injection circuit

2.2 Bipartite LFSR Method Architectural Framework

To measure design parameters like dissipated power, the realization of an LFSR may be modified during testing. Altering the pattern sequence or introducing new patterns can modify the total unpredictability. One method involves sandwiching a test pattern between two random patterns to reduce the number of transitions between them. In this approach, two non-overlapping clock (CLK) signals are used to partition the LFSR into two equal portions. Alternatively, both halves of the LFSR could be inactive when one is active. A few flip-flops are successively linked along a clock input (a) to form an LFSR, as seen in Figure 3, which depicts the basic setup of the Bipartite LFSR to create pattern Ti1.

A bipartite LFSR uses half of each of two consecutive random patterns to create an intermediate pattern. It is suggested to use a random shape generator that combines two methods of test pattern generation, such as R-Injection and Bipartite LFSR.

The main advantage of the proposed technique is its applicability to both combinational and sequential circuits without deteriorating the randomness quality of patterns. An LFSR is a shift register whose

input bit is the output of a linear function of two or more of its previous states (taps). LFSRs are fundamental components of many running-key generators for stream cipher applications.
An LFSR can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops and feeding those outputs back into the input of one of the flip-flops.
In the bipartite LFSR design, when clk1clk2 = 10, the dummy flip-flop appends by storing the LFSR's n/2nd bit and passing it to the (n/2+1) flip-flop during the subsequent half, resulting in operation when clk1clk2 = 1. The LFSR process may be divided into two identical halves, and the protected flip-flop joins the two halves together. An n-bit LFSR is divided into two double n/2-bit LFSRs, which reduces testing and clock tree power consumption. However, this method requires creating and allocating two non-overlapping clock signals (at half frequency), increasing the area overhead and reducing the LFSR's ability to handle uncertainty.
RI and Bipartite LFSR are two test pattern generating techniques that can be accomplished using the suggested method, which also minimizes power consumption. Combining these two methods with an LFSR configuration yields the LP-LFSR, which has a lower average power consumption than Bipartite LFSR methods. The injected patterns from the middle patterns can complement the data produced by a conventional LFSR for error detection, as the injected patterns are random. Figure 4 shows a 32-bit LP-LFSR configuration combining random injection and bipartite LFSR. The lower bits of a bipartite LFSR are used as the input source for an injector circuit, demonstrated in Fig. 5, while the upper bipartite LFSR is used as the output source. Figure 4 displays the non-overlapping clock input pattern. The Mux network selects the bipartite LFSR bit or the injector digit bit. Control signals, or clocks clk1 and clk2, are used, along with sel1 and sel2 signals to choose the higher or lower digit of the RI injection network, and the overlapping clock signals clk1 and clk2 can all be generated using finite state automation (FSA).

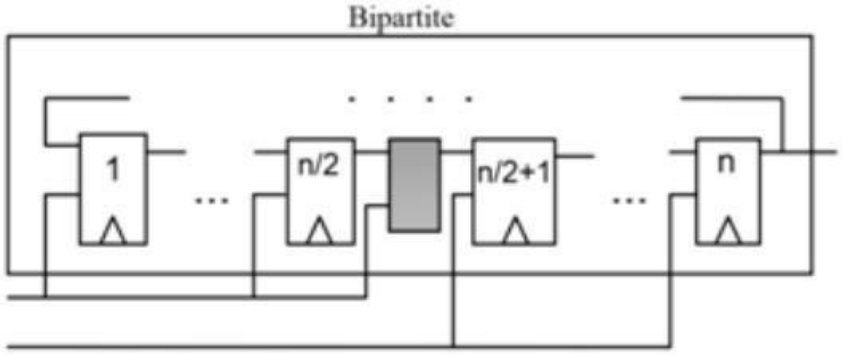


Figure.3. Bipartite LFSR
To enhance test pattern creation, a finite-state automation (FSA) generates the following control signals: In the first clock cycle, Sel1 and Sel2 are set to 10 and 11, respectively. The bipartite LFSR is active in the top half and inactive in the bottom half. When Sel1Sel2 = 11, the bipartite LFSR outputs are forwarded to both halves (Out). Assuming Ti is generated in this scenario. In the second clock cycle, Sel1Sel2 = 10 and clk1clk2 = 00. Both sides of the bipartite LFSR are inactive. Outputs [31:16] and [15:0] are the outputs of the RI injector circuit and the higher half of the bipartite LFSR, respectively. Ti1 is generated in this step.During the third clock cycle, clk1clk2 = 01 and Sel1Sel2 = 11. The top half of the bipartite LFSR is inactive, while the bottom half is active. It is locked to the output (Out[31:0]) of the Bipartite LFSR, generating Ti2.In the fourth clock cycle, Sel1Sel2 = 01 and clk1Clk2 = 00. Both parts of the bipartite LFSR are inactive. The bottom half of the bipartite LFSR is forwarded to the outputs (Out[15:0]) along with the injector outputs (Out[31:16]) when Sel1Sel2 = 01, producing Ti3.
The process returns to step 1 in the fifth clock cycle to obtain Ti+1. The architecture shown in Figure 4 includes two "16-bit higher D-type flip-flops" and one "16-bit lower D-type flip-flop." The last bit produced by the higher flip-flop during the clock cycle is stored in the "dummy ff."

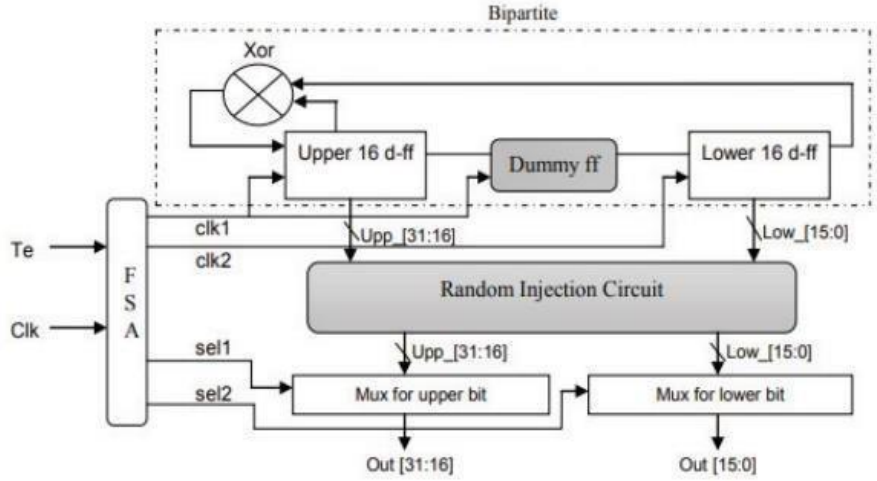


Figure.4 32-bit LP-LFSR Structure

The XOR procedure is used to generate multiple patterns. A 4-cycle clock is required for the "non-overlapping enable signal" and "Mux select line." Select Signal 1 and Select Signal 2 are used to select the Mux for the upper bit and lower digit, respectively. Figure 5 shows the LFSR low-power algorithm. The Low Power LFSR Algorithm helps in managing flow and generating a pseudo-random pattern generator by sequentially applying the process.

3. Results and Discussion

3.1. LTLFSR

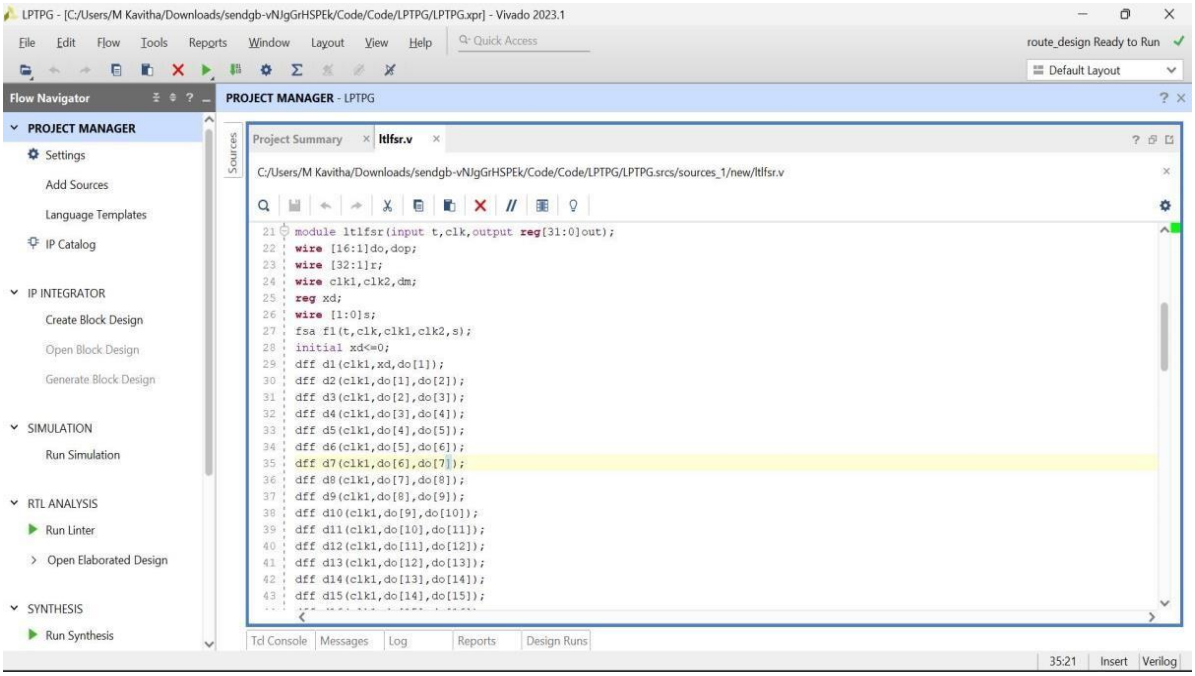


Figure.5. Code for D-Flip Flop of 32-bit LP-LFSR

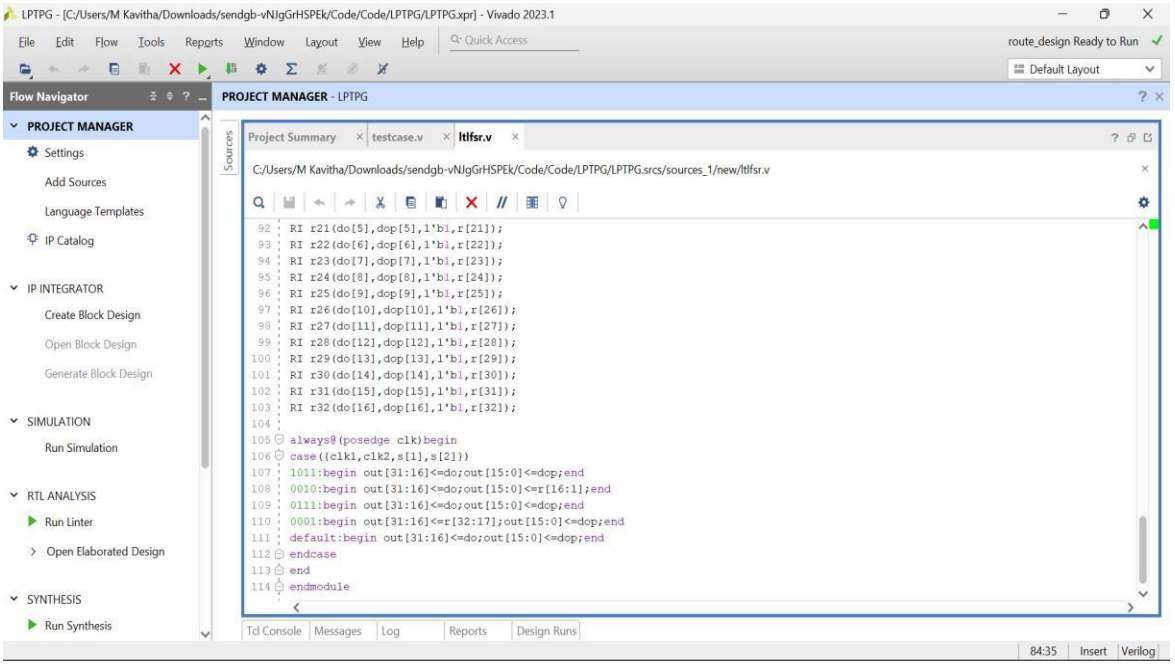


Figure.6. Code for D-Flip Flop of 32-bit LP-LFSR

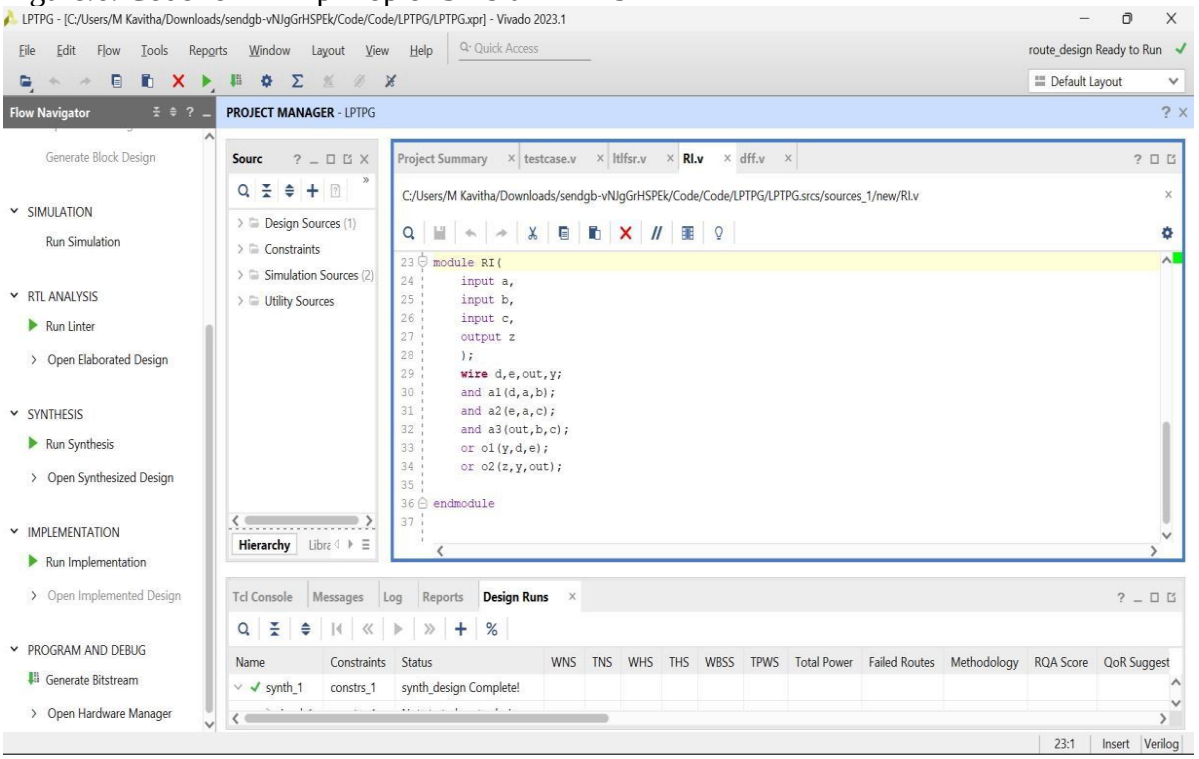


Figure.7. Code for RI circuit of 32-bit LP-LF

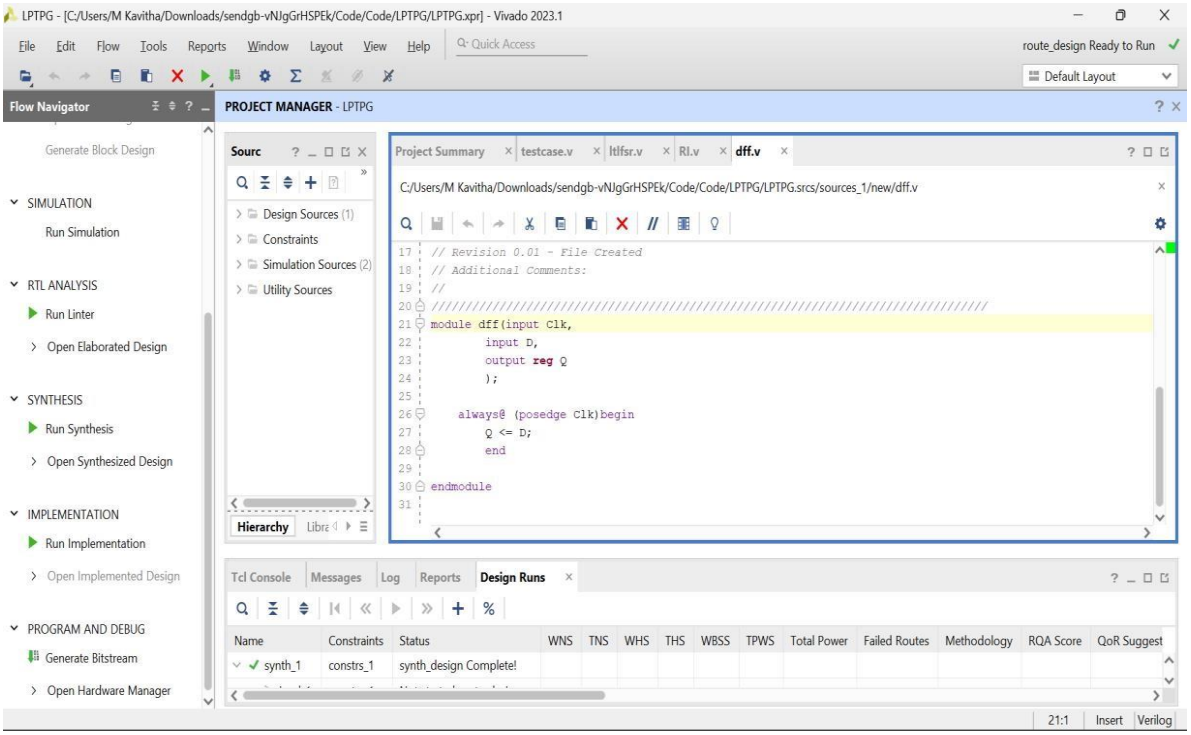


Figure.8. Code for 32-Bit LP-LFSR

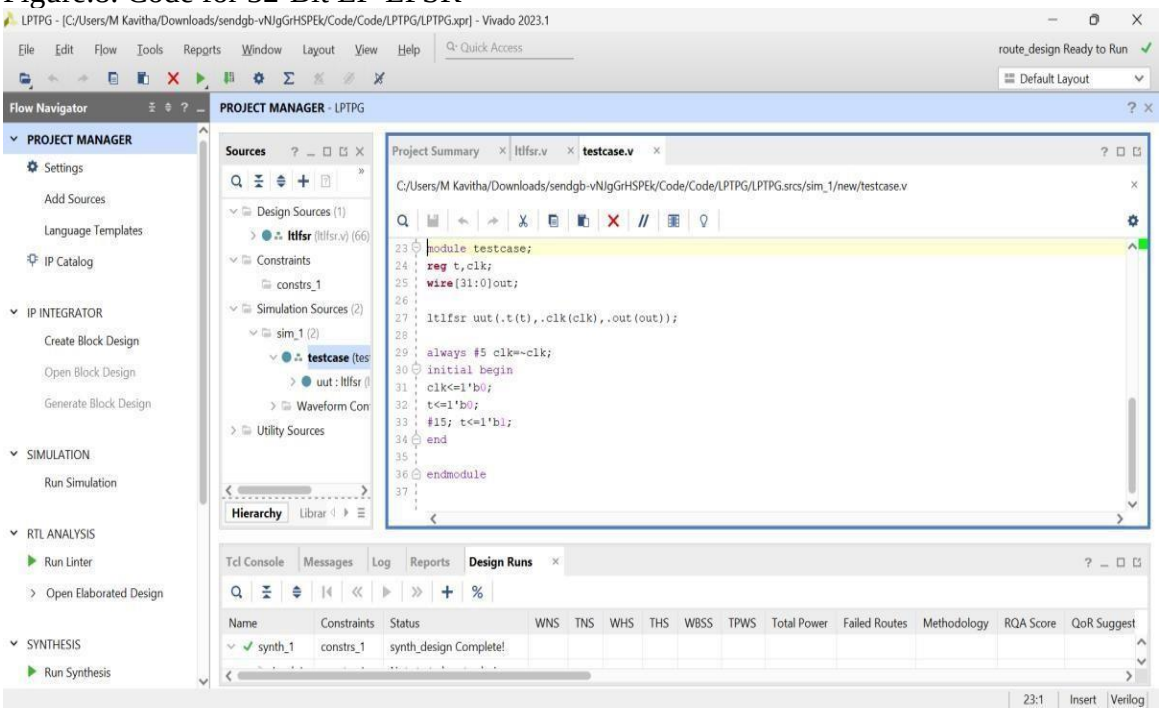
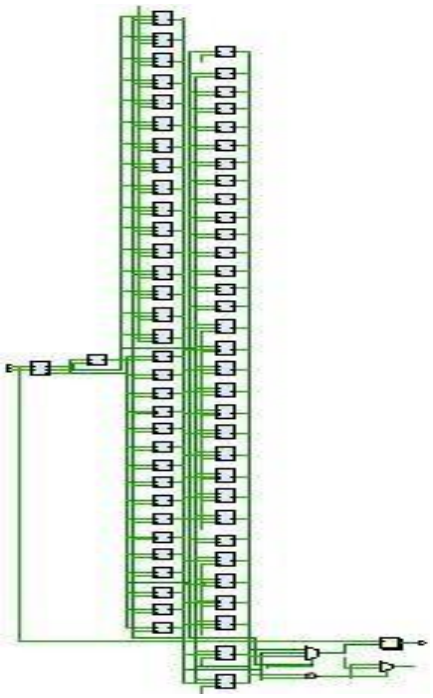
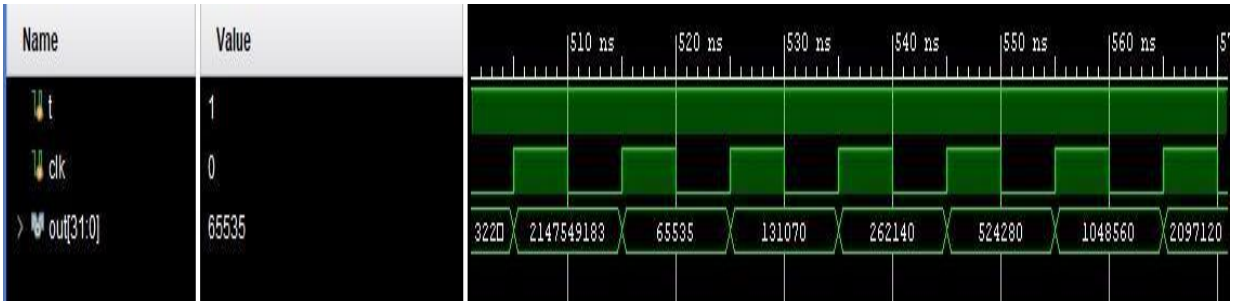


Figure.9. Module Testcase

3.2 RTL Schematic



3.3Simulation Results



3.4 Advantages and Applications

3.4.1 Advantages

- Dynamic power is reduced.
- Switching activity improved.

3.4.2 Applications

Test Pattern Generation: A low-power test pattern generator is proposed using LFSR, a binary to excess-4 converter, and a binary ripple counter. The generated test vectors exhibit high correlation between successive vectors, leading to minimal switching. The proposed method demonstrates a significant reduction in dynamic power consumption compared to existing approaches in test benchmark circuits.

Key Generation for Lightweight Cryptography: Key generation for lightweight cryptography using low-power LFSR involves configuring the LFSR with specific tap positions and clocking it to produce a key sequence. The tap position determines the feedback mechanism within the LFSR, influencing the randomness and cryptographic strength of the generated key.

Embedded Systems: LFSR is a component in complex cryptographic schemes that enhances high-level security in embedded systems. Its implementation with simple hardware components reduces the complexity and cost of embedded system design.

Signal Processing: Low-power LFSRs are commonly used in signal processing applications such as generating pseudorandom sequences for spread spectrum communication, scrambling data in digital communications to minimize interference, and generating test patterns for circuit testing. Additionally,

they are used in cryptography for generating cryptographic keys and in error detection and correction algorithms.
Digital Design: Designing a low-power LFSR for digital applications involves optimizing the circuitry to minimize power consumption while maintaining functionality. Techniques like clock gating, power gating, and optimizing transistor sizing can be employed. Low-power design methodologies such as voltage scaling and reducing switching activity can further enhance power efficiency.

4. Conclusion

The design and verification of a Verilog HDL for low power use for a test pattern generator using the Low Power LFSR approach is demonstrated in this work. It also discusses a theory for describing the design of a test pattern using an architecture with a low power linear feedback shift register. This method, when paired with the LP-LFSR algorithm, can use less power than the conventional LFSR approach. The low-power LFSR uses 67.34 percent less dynamic power than the conventional LFSR, as can be observed. The results demonstrate that the LP-LFSR is suitable for situations where power consumption is a key factor.

Reference:

1. A. Molina-Rueda, F. Uceda-Ponga, and C. F. Uribe, “Extended period LFSR using variable TAP function,” Proceedings - 18th International Conference on Electronics, Communications and Computers, CONIELECOMP, pp. 129–132, doi: 10.1109/CONIELECOMP.2008.8.
2. X. C. Gu and M. X. Zhang, “Uniform random number generator using Leap-Ahead LFSR Architecture,” International Conference on Computer and Communications Security, ICCCS 2009, pp. 150–154, doi: 10.1109/ICCCS.2009.11.
3. R. L. ur S. Avinash Ajane, Paul M. Furth, Eric E. Johnson, “Comparision of Binary and LFSR Counters and Efficient LFSR Decoding Algorithm,” Proceedings of IEEE conference, 2011, pp. 0–3.
4. A. K. Panda, P. Rajput, and B. Shukla, “FPGA implementation of 8-, 16- and 32-bit LFSR with maximum length feedback polynomial using VHDL,” Proceedings - International Conference on Communication Systems and Network Technologies, CSNT pp. 769–773, doi: 10.1109/CSNT.2012.168.
5. Z. Dai, L. Nan, X. Yang, and X. Li, “Design and implementation of configurable LFSR instructions targeted at stream cipher processing,” J. Circuits, Syst. Comput., vol. 22, no. 10, pp. 11–12, 2013, doi: 10.1142/S0218126613400367.
6. P. Dhanesh and A. Jayanth Balaji, “Dual threshold bit-swapping LFSR for power reduction in BIST,” ICACCS - Proceedings of the 2nd International Conference on Advanced Computing and Communication Systems, 2015, pp. 5–7, doi: 10.1109/ICACCS.2015.7324061.
7. R. Jamgade, S. Ambatkar, and S. Kakde, “HDL Implementation of PN Sequence Generator Using Vedic Multiplication and Add & Shift Multiplication,” Proceedings - 5th International Conference on Communication Systems and Network Technologies, CSNT, pp. 854– 858, doi: 10.1109/CSNT.2015.176.
8. S. Ukey, S. Rathkanthiwar, and S. Kakde, “VLSI implementation of low power scan based testing,” International Conference on Communication and Signal Processing, ICCSP, pp. 866–870, doi: 10.1109/ICCSP.2016.7754