International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-4, February 2019

FPGA based Pseudo Random Sequence Generator using XOR/XNOR for Communication Cryptography and VLSI Testing Applications

B.Murali Krishna, G.L.Madhumati, Habibulla Khan

Abstract: ***Random number generators are most prominently used in the area of communication to provide security for information systems through pseudo random sequences. It also applicable for key generation in cryptography applications and signature analyzer to generate test patterns for Built-In-Self Test. In conventional method, random numbers are generated by a reference value i.e., seed value, using a XOR gate. The new proposed methods present a linear feedback shift register (LFSR) which generates an arbitrary number based on XOR, XNOR gates with and without seed value using multiplexer. Multiplexer is append to generate a random value at user defined state in runtime. Hardware complexity and power consumption is reduced by replacing the multiplexer with tristate buffers. Result analysis indicates that proposed LFSR with and without seed value gives a better performance, low power consumption and improves more randomness in runtime with Partial Reconfiguration (PR). Resource utilization for standard XOR based LFSR is compared with proposed LFSR using XOR and XNOR logic. Proposed method is designed in Verilog HDL, simulated with ISE Simulator, synthesized and implemented using Xilinx ISE, targeted for Spartan3E XC3S500E-FG320-4 and Virtex-5 XUPV5LX-110T architecture.***

Keywords: ***LFSR, XOR, XNOR, Multiplexer, Xilinx, PR, FPGA.***

I. INTRODUCTION

A wide variety of wireless applications such as home automation and weather monitoring etc., are becoming popular as wireless sensor networks have become flexible and inexpensive. As secure information transfer through a channel has become a bottleneck, most of the mechanisms employed for security rely on random number generators to perform their transfer operation. With the diminishing device sizes, the cost of the hardware became cheaper; hence generation of random numbers using hardware is preferred when compared to software approach. An unpredictable random number should be generated for the sake of secure data transmission.

The next number is generated in uncorrelative manner, i.e., if it exhibits true randomness then it is called as true random number generator (TRNG). On the other hand the pattern generated using Pseudo random number generator (PRNG) appears to be repetitive and predictable, as the generation is based on the mathematical algorithms. Thus, if there is access to gather the generated random numbers until the sequence restarts from PRNG, then the hacker of the sequence generator can predict the rest of the sequence. The generated pattern using PRNG can be made more random and non-repetitive if very large sequence is generated using PRNG, thereby increasing the cycle of repetition of the generated sequence. Ideal direct sequence spread spectrum systems need spreading codes which would be truly random number sequences, where transceiver generates a sequence in order to communicate with one another. Perfect synchronization searches by changing the settings, until the transmitter timing is located. Generating a truly random sequence is very difficult and has a little application. Most of the applications deploy PRNG to generate the random sequences because TRNG usage requires predetermined sequences in all transmitters, with an equivalent copy in all receivers [1]. The total available bandwidth can be allotted to the available users without any misuse; the spectrum efficiency of CDMA is high, because it uses LFSR. During the information transmission, the pseudo random sequence (also called as pseudo-noise, PN) is multiplied with the actual information to accomplish a wideband spread spectrum signal. The resultant transmitted signal is demodulated at the receiver end by multiplying it with a synchronized copy of the PN sequence used at the transmission end. The noise signal PN is private to each user, which helps in data protection and in turn allowing the bandwidth sharing.

II. RANDOM NUMBER GENERATORS

LFSRs are used to generate random numbers or symbols which cannot be predicted. LFSRs have a wide range of applications in gambling, statistical sampling, simulation, cryptography, completely randomized design and mostly in the areas of mobile and space communications. Day-to-day advancements, the technology grow faster and denser it can be more optimum to implement the random number generator in FPGA. The reconfigurable nature of FPGA can adopt and perform as configured; it consists of reconfigurable logic blocks which use less hardware for generating random numbers. It provides flexibility to change the design in runtime using Partial Reconfiguration (PR). Effective hardware reuse, runtime selective design change, better throughput, low latency and power are an advantage using PR.

Manuscript published on 28 February 2019.
***Correspondence Author(s)
B.Murali Krishna, Research Scholar, Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.
G.L.Madhumati, Professor & HOD, Department of ECE, Dhanekula Institute of Engineering & Technology, Ganguru, AP, India.
Habibulla Khan, Professor & Dean, ,Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.

PR is applicable to diversified applications where there is demand with area and power. Random numbers are generated using Linear Feedback Shift Registers (LFSR) [2]. This technique can extend to an N number of bits to generate better uncorrelated sequences for random number generation. Linear Feedback Shift Register (LFSR) are often used to generate test patterns and to compare test outputs into signatures for Buit-In-Self-Test [3]. The bit positions that effect the serial input are called taps . The general form of an LFSR is a shift register with two or more flip-flop outputs XORed together and fed back to the a selective flip-flop based on a polynomial [4]. The term linear comes from the fact the fact that exclusive OR is equivalent to modulo-2 addition, and addition is linear operation. The below figure shows an example of 4-Bit LFSR.

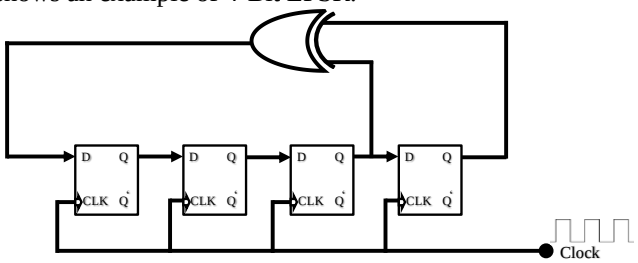

Figure-1: Block Diagram of Linear Feedback Shift Register (LFSR)

The outputs from the third and fourth flip-flops are XORed together and fedback into the input of the first D flip-flop.The taps are positions 3 and 4. LFSR can cycle through upto 2n-1 states. All the zero states are excluded from the sequence. Linear feedback shift registers are also referred as "maximal count" counters (a counter that counts 2n-1) unique states shown in figure 1. These counters are very fast and requires very less hardware because the counter is made up of a shift register and few XOR taps. In pseudo random generation, the initial value of the sequence is known as a seed. A constant to define the initial value and load it into the LFSR during system initialization.

III. CONVENTIONAL & PROPOSED LFSR USING XOR/XNOR

Random numbers are generated using different types of LFSRs. Random number is generated based on three factors. They are first factor is its polynomial, second factor depends on logic used either XOR or XNOR logic, third factor is feedback style may be internal feedback or external feedback to the no.of flip-flops. We proposed few LFSRs based on XOR and XNOR logic.They are.,

Design of Runtime Polynomial Change

Run time polynomial change uses two eight to one multiplexers, XOR gate and eight "D" flip-flops. Outputs of eight to one multiplexers are connected to XOR gate whose output is feedback to first flip-flop. Outputs of few flip-flops are connected as input to eight to one multiplexers, based on polynomial which generates the random number. Flip-flops combinations like 7,8, 6,7 5,8 4,7 6,8 3,7 4,5 and 1,8 outputs of these Flip-flops are selectively xored and feedback to first flip-flop. Selection lines of eight to one multiplexer are programmed such that taps for the LFSR gets selected. Design of Run Time Polynomial Change using XOR as shown in figure 2 and XNOR is shown in figure 3.

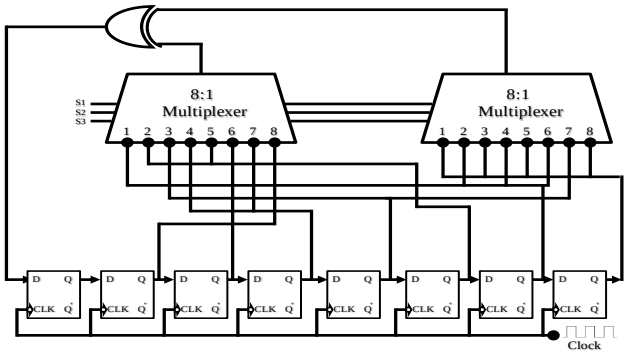

Figure-2: Design of Runtime Polynomial Change using XOR Logic

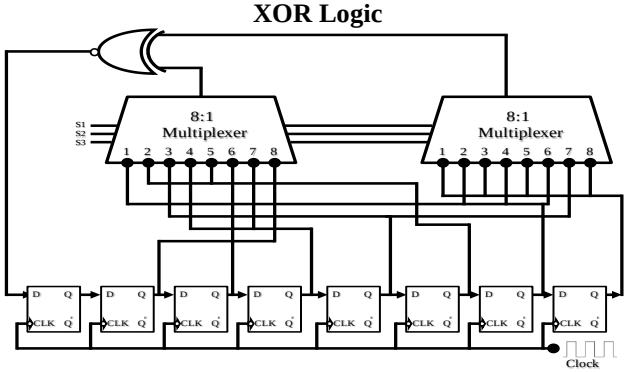

Figure-3: Design of Runtime Polynomial Change using XNOR Logic

A.Design of Jitter based LFSR

Jitter on Ring Oscillators is one of the concepts used for acquiring Random Numbers. Jitter is the timing variation of signal edges from their ideal values.

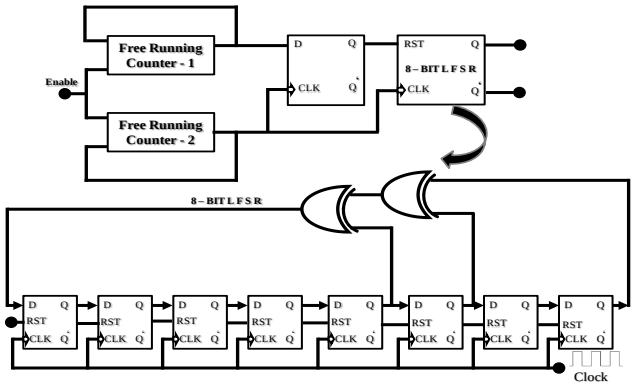

Figure-4: Design of Jitter based LFSR using XOR Logic

This variation gives a scope for the generation of random numbers. Ring Oscillators consists of definite frequency based on arrangement and placement of components and temperature. Thus, every ring oscillator has its own definite frequency. Instead of ring oscillators two free running counters are enabled which produce a slight frequency difference gives rise to jitter passed through a D Flip-Flop enables the LFSR which produces a random number unpredictable. The architecture for the XOR based random number generation using jitter is shown in figure 4 and XNOR based LFSR is shown in figure 5.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

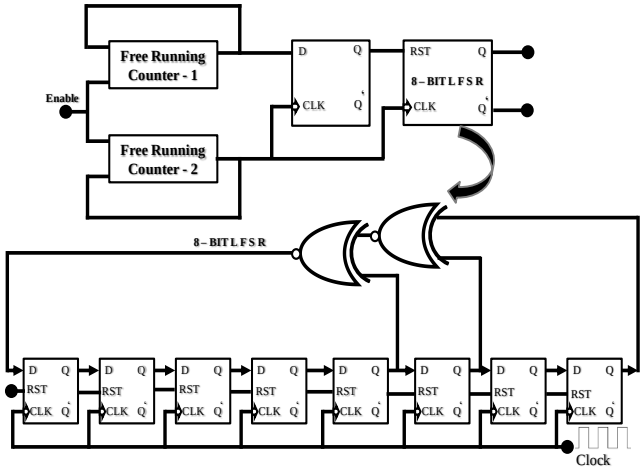

Figure-5: Design of Jitter based LFSR using XNOR Logic

B.Design of BRAM based LFSR

Random values are stored in FPGA Block RAM through IP Core with address. Eight bit counter used as address generator applied to BRAM, which generates pre stored random value through flip-flops. The flip-flops outputs q generates the original values stored in BRAM and q' generates the complement values. BRAM method for random number generation is shown in figure 6.

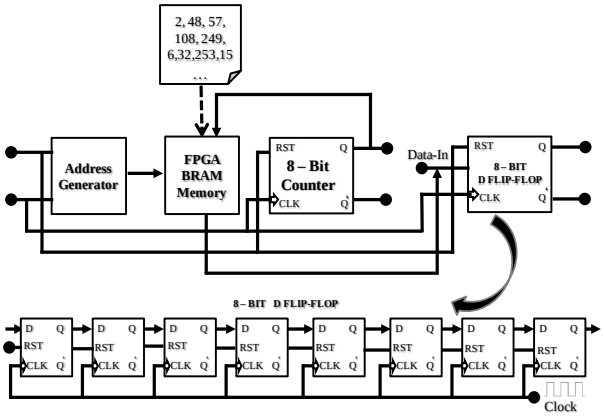

Figure-6: Design of BRAM based LFSR

C. Design of Galios LFSR

In galois LFSR the data flow is from left to right and the feedback path is from right to left. The polynomial increments from left to right. The construction of eight-bit Galois LFSR with eight D flip-flops and their respective outputs of fourth, fifth, sixth LFSRs, outputs are delivered to next neighborhood flip-flops through XOR gates based on polynomial. The output of the eight flip-flop is internal feedback to first input of three XOR gates amongst fifth, sixth and seventh flip-flops. The output of fourth, fifth, sixth flip-flops, are applied to second input of three XOR gates. These series of operations performed for every input of clock cycle, shifting the inputs to outputs through each flip-flop based on polynomial. Design of Galios LFSR shown in figure7 using XOR and in figure8 with XNOR logic, constructed with polynomial based on internal feedback.

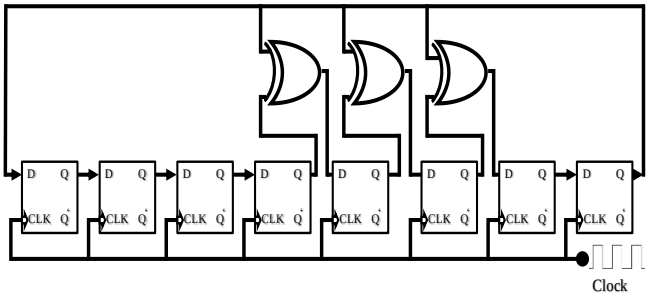

Figure-7: Design of Galios LFSR using XOR Logic

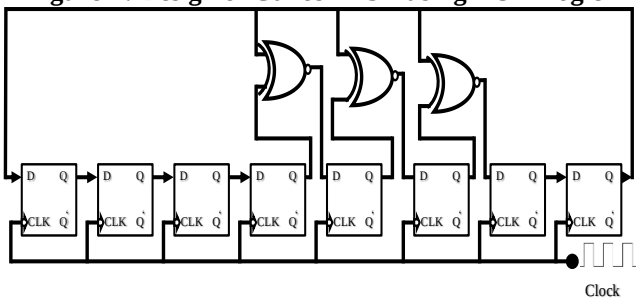

Figure-8: Design of Galios LFSR using XNOR Logic

D.Design of Fibonacci LFSR

Fibonacci topology using XOR as shown, figure 9 and XNOR is shown in figure 10 depicts that the data flow is from left to right and the feedback path is from right to left, like Galois implementation. The polynomial decrements from left to right. The construction of eight-bit Fibonacci LFSR with eight D flip-flops and three XOR gates at their respective outputs of fourth, fifth and sixth which is external feedback to the first flip-flop.

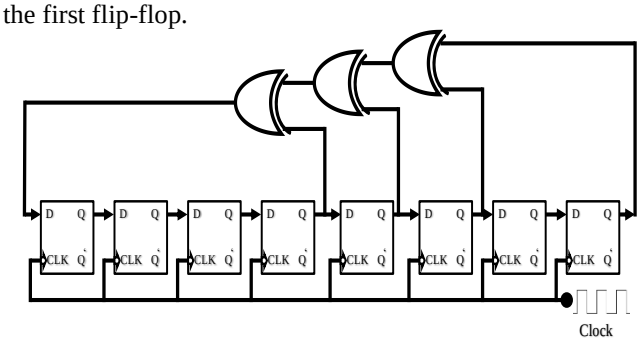

Figure-9: Design of Fibonacci LFSR using XOR Logic

Random numbers are generated only a clock as an input to drive the flip-flops and no external input is applied for XNOR logic. The output of flip-flop D8 and D6 is applied to third XOR gate and feedback second XOR gate with D5 output. Output of second XOR gate with D4 output is feedback to first flip-flop D1. These series of operations performed for every input of clock cycle, shifting the inputs to outputs through each flip-flop based on polynomial. This will generate an 8-bit random sequence for a given input of non-zero pattern by shifting each value from one flip-flop to another for every clock pulse. The LFSR which generates a maximum length sequence and 28 -1=255 non-repetitive patterns excluding all zeros in the input pattern.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

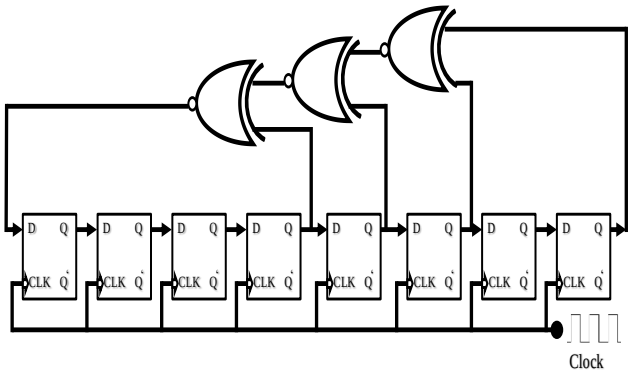

Figure-10: Design of Fibonacci LFSR using XNOR Logic

E. Design of LFSR

Design of conventional LFSR using feedback as XOR logic shown in figure11 and proposed LFSR using XNOR feedback logic in figure12.

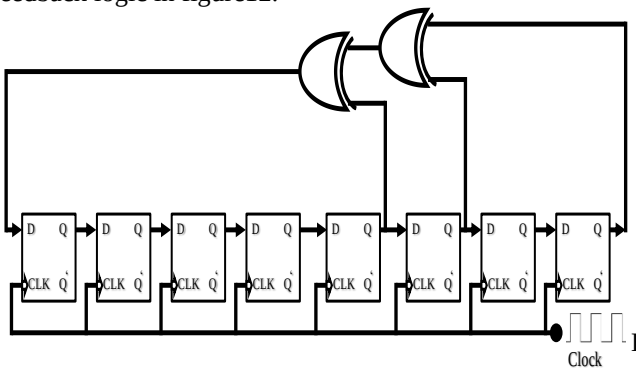

Figure-11: Design of LFSR using XOR Logic

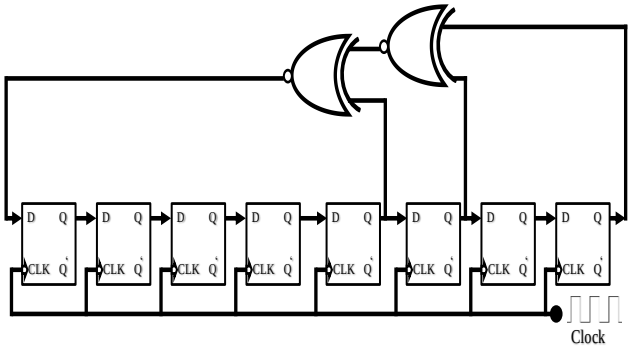

Figure-12: Design of LFSR using XNOR Logic

F. Design of LFSR XOR

Combined LFSR/XOR is random number generator which is combination of LFSR and an XOR gate. It requires at least two seed to produce arbitrary number which are used as key for encryption purpose. Combined LFSR/XOR requires (N-1) bit LFSR for N bit random number generation. The architecture for Combined LFSR/XOR is as shown in figure 13, and LFSR/XNOR is as shown in figure 14 where the output of each flip-flop is Xored to produce N^{th} bit. The output of D7 and D1 flip-flops are applied to XOR gate and feedback to D1. It is used to produce pseudo exhaustive test pattern. These series of operations performed for every input of clock cycle, shifting the inputs to outputs through each flip-flop based on polynomial with a combination of LFSR and a XOR linear network. The network is based on linear sums or linear codes.

H.

I.

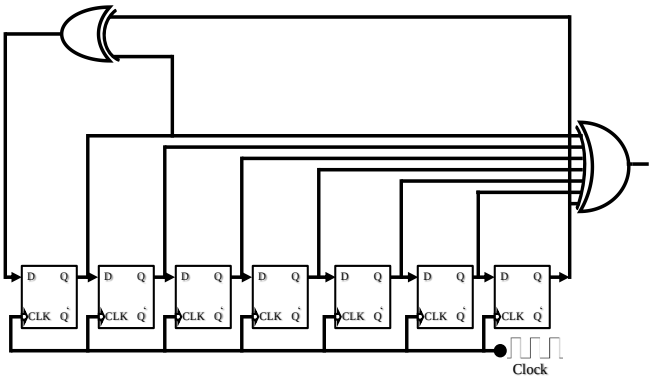

Figure-13: Design of Combined LFSR using XOR Logic

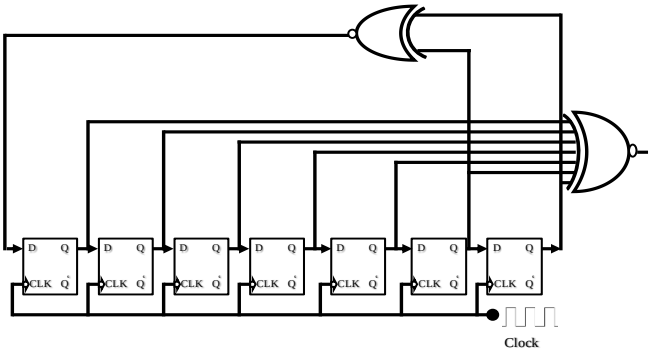

Figure-14: Design of Combined LFSR using XNOR Logic

G. Design of LP LFSR

A linear-feedback shift register, whose input bit is a linear function of its previous state. Generally, a flip-flop consists of clock and reset pins. Since, in LFSR, the clock paths to all flip-flop toggles at every clock cycle, they consume a significant amount of power. Due to advancements in technology, the demand for portable computing devices is increasing rapidly in communication system. Challenging areas in VLSI are performance, cost, testing, area, reliability and power. These applications require low power dissipation for VLSI circuits.

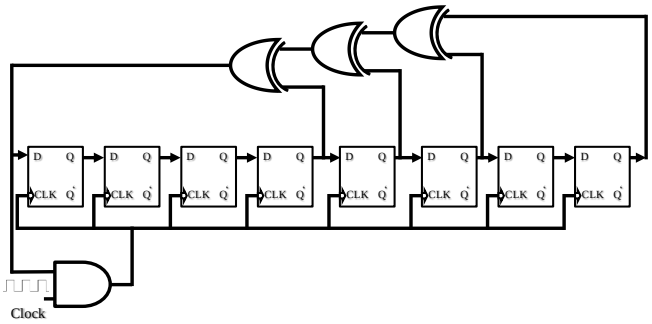

Figure-15: Design of LP LFSR using XOR Logic

The power dissipation during test mode is more than in normal mode. Hence, it is important to optimize power during testing. Different methods are proposed for the purpose of random number generation. In the present scenario where power reduction is the predominant issue. Clock gating is introduced to reduce the power consumption. The purpose of clock gating, to apply the clock pulse to flip-flops through and gate, when switching activity required to generate a random number.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Clock pulse is applied to one input of and gate and feedback signal is connected to second input. Average power consumed by the LFSR is reduced by using clock gating. In order to reduce power dissipation in digital VLSI design, the EDA tools offers commands like example lp_insert clock gating low, medium and high levels, which can be applied to design after synthesize to produce optimized low power design. The block diagram for the low power LFSR with clock gating using XOR is as shown in above figure 15 and XNOR in figure 16.

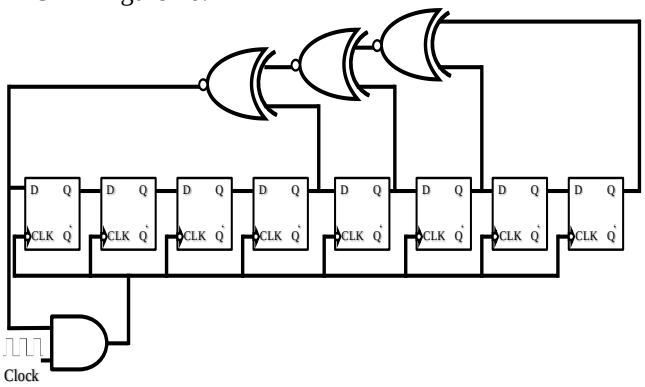

Figure-16: Design of LP LFSR using XNOR Logic

IV. DESIGN OF BIT SWAPPING LFSR

The term swapping defines the interchanging of bits, which may be two or more. The bit swapping process depends on the operation of the multiplexer.

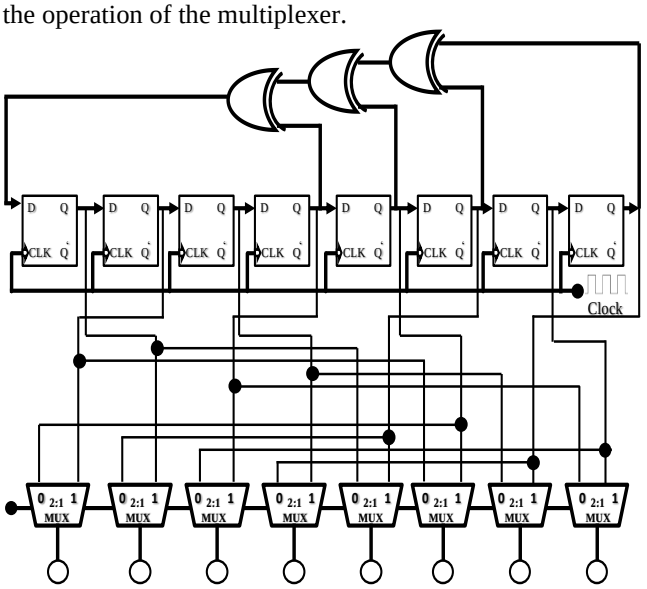

Figure-17: Design of Bit Swapping LFSR using XOR Logic

This select line plays a major role in swapping of different pair of bits. Consider an eight bit number, named as from A8A7A6 A5 A4 A3 A2 A1. When the selection line input given as zero the first four bits A8A7A6 A5 are swapped into the position of A4 A3 A2 A1 vice-versa. Likewise, when the select line input is set as one then every nibble two adjacent bit positions are swapped with their neighboring adjacent two bits in the series like A8A7 positions with A6 A5, similarly the positions of A4 A3 to A2 A1 vice-versa. The construction of bit swapping LFSR Architecture is using XOR shown in figure 17 and XNOR is shown in figure 18.

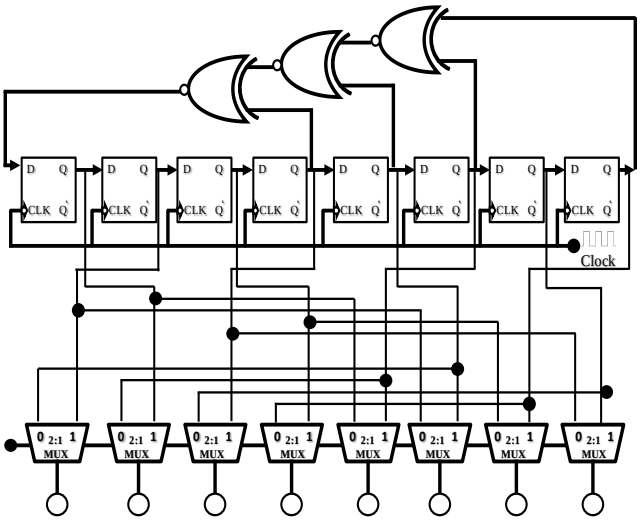

Figure-18: Design of Bit Swapping LFSR using XNOR Logic

I. Design of Cellular Automata LFSR

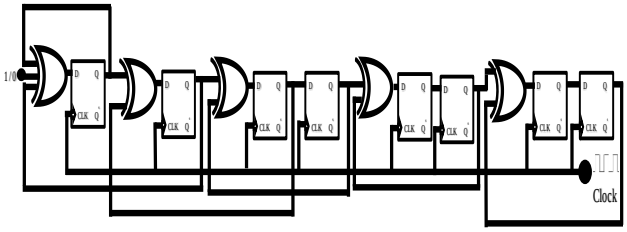

Figure-19: Design of Cellular Automata LFSR using XOR Logic

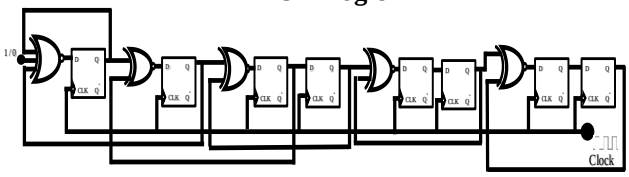

Figure-20: Design of Cellular Automata LFSR using XNOR Logic

A cellular automaton is a discrete model, which consists of a regular grid of cells. The grid can be in any finite number of dimensions. Each cell consists of two states as ON and OFF or logic'0' and logic'1' in binary form. For each cell, a set of cells called its neighborhood. A cell and its neighbors form a neighborhood of 3 cells, so there are 23=8 possible patterns for a neighborhood. The rule defining the cellular automaton must specify the resulting state for each of these possibilities so there are 256 = 28 possible elementary cellular automata. These 256 cellular automata are invented by Wolfram and given each rule a number from 0 to 255.

Rule 90 is an elementary cellular automaton based on exclusive-or function. It consists of a one-dimensional array of cells, each of which holds a single binary value (0 or 1). All cells obey the same rule. A pre-specified rule determines the new value of each cell as a function of its previous value and of the values in its two neighboring cells. Rule 150 is an elementary cellular automaton introduced by Stephen Wolfram in 1983. It consists of a one-dimensional array of cells, each of which holds a single binary value (0 or 1). All cells obey the same rule.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

It specifies the next cell, depending on its value and its immediate neighbors. Cellular Automata LFSR shown in figure19 using XOR and in figure20 with XNOR logic based on Rule-90 and Rule-150 random numbers are generated.

V. APPLICATIONS OF LFSR

The LFSR can be applicable in generation of pseudo-random noise, counters, cryptography techniques and circuit testing using random test values [5]. LFSRs are also implemented in generation of white noise sequences [6].

A. *Pseudo Random Noise Generators*

A pseudo random noise generator generates a sequence of codes which are required for Code Division Multiple Access (CDMA) technique [7]. These pseudo random codes in CDMA are used to generate code for each and every individual at the transmission interface by using these Linear Feedback Shift Registers (LFSRs) shown in figure 21. Code Division Multiple Access (CDMA) uses a most popular Direct Sequence Spread Spectrum (DS-SS) technique, used to transmit data over a wide radio spectrum using this pseudo random binary sequence [8].

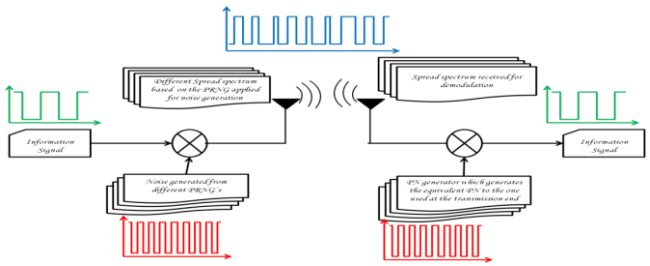

Figure-21: Sequence of codes for CDMA using LFSR

B. *Counters*

The continuous random number generation in LFSR makes it used as a counter or a clock divider. These counters have simple feedback logic than Gray-code or natural binary counters. LFSR should never enter an all-zeros state and at start-up it should be started with any of the states in the sequence [9].

C. *Cryptography*

LFSRs are used to generate pseudo-random numbers which are used as keys in Symmetric Applications. Public Key is used to encrypt the given plain text to outline cipher at message transmitting end. Private Key is used to decrypt the cipher back to plain text for appearance the original message.

D. *Circuit Testing*

LFSRs used in circuit testing generate different test patterns for exhaustive testing, pseudo-exhaustive testing, pseudo-random testing, signature analyzer and Built-In-Self-Test (BIST) technique [10]. LFSR is mostly used in pattern generators for exhaustive testing as they cover overall inputs for an n-input circuit. LFSR is used in BIST technique by storing all the circuit output in a compressed form as a signature and this signature is compared to the good circuit without any defects known as a golden signature [11].

E. *Digital Broadcasting and Communications*

To prevent the generation of repetitive sequences of 0s and 1s a special techniques known as scrambling and chipping code [12]. When the LFSR runs with the same bit rate of the transmitted message, this is known as scrambling [13]. The LFSR runs a faster bit rate than the transmitted message; this is known as chipping code. The chipping code along with the message using the exclusive or operation using binary phase-shift keying produces as signal of higher bandwidth than the transmitting message this is the method of spread-spectrum communication [14]. When several signals are transmitted in same channel is distinguished at the time and frequency. LFSRs are used in radio signal jammer which generates pseudo-random noise to raise the noise floor at the receiver [15].

VI. RANDOM NUMBER GENERATOR USING DUAL FEEDBACK LOGIC

A. *Conventional Method*

Conventional method presents the 4-bit linear feedback shift register (LFSR) circuit uses an XOR, and Flip Flops shown in figure 1. To generate random value at run time, multiplexer is appended before each flip-flop to serve two purposes shown in figure 22.

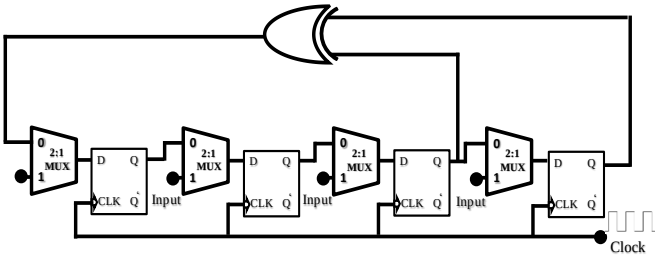

Figure-22: Conventional method of LFSR with XOR, Multiplexer and D-Flip Flops

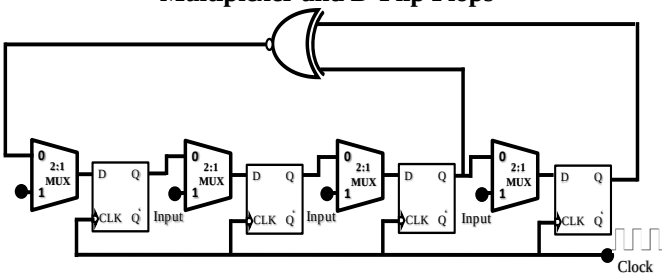

Figure-23: Proposed method of LFSR with XNOR, Multiplexer and D-Flip flops

First purpose is to feedback the previous value, second one is to apply user input seed value at second input to each multiplexer. To generate the random number it requires the initial seed value.

B. *Proposed Method*

Proposed method presents the 4-bit linear feedback shift register (LFSR) circuit uses an XNOR, and Flip Flops to generate random value at run time, shown in figure 23. To generate the random number it doesn't requires any initial seed value, which is the advantage of proposed method.

1. METHOD -I

The proposed Method-I presents the 4-bit LFSR which generates a random number using XOR, XNOR logic.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

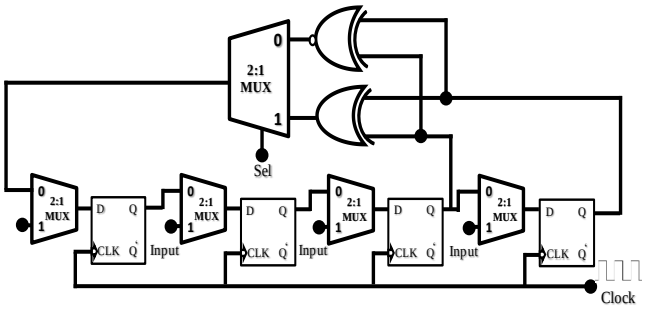

Figure-24: Proposed Method-I LFSR with XNOR, XOR Multiplexer and D-Flip flops.

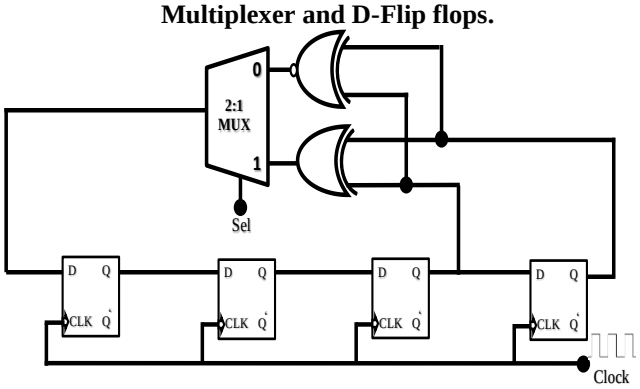

Figure-25: Proposed Method-I LFSR with XNOR, XOR and D-Flip flops without seed value.

2. METHOD -II

Method- I is a combination of conventional and proposed methods. XNOR gate doesn't require any seed value to be given as an input. A "0" state is generated using XNOR gate and "F" state is generated using XOR gate.

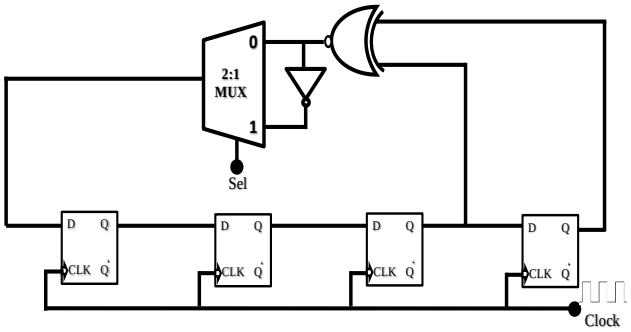

Figure-26: Proposed method-II LFSR with XNOR, NOT gate and D-Flip flops without seed value.

Selectively the logic is reconfigured at runtime by varying the selection line of multiplexer. User defined random numbers are generated by giving the seed value to each multiplexer appended before each flip-flops shown in figure 24. The random numbers are generated, by loading the flip flops initial state using XNOR gate to avoid reference seed value shown in figure 25. Hardware complexity is reduced, because multiplexers before flip-flops which are used to load seed value are removed.

Hardware complexity can be further reduced by replacing XOR gate with an inverter at the output of XNOR gate shown in figure 26. Random numbers with different logic are generated at runtime by programming the selection line. The multiplexer can be replaced with tri-state buffers and XNOR gate output is passed through an inverter which replaces the XOR gate. Design utilizes less hardware resources. The proposed model with less hardware without seed value shown in figure 27. When selection input is "0" circuit generates a random number using XNOR gate logic, when selection input is "1" random number is generated using XOR gate logic.

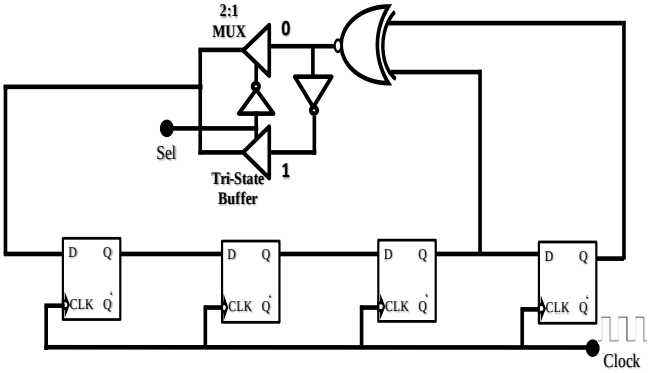

Figure-27: Proposed method-II LFSR with XNOR, low complexity multiplexer and an inverter.

VII. SIMULATION

Simulation results of various LFSRs are shown below. Simulation response clearly represents that conventional approach using XOR and proposed approach using XNOR logic, the generated test vector randomness is high. It can be applicable to N-Bit for diversified applications discussed above.

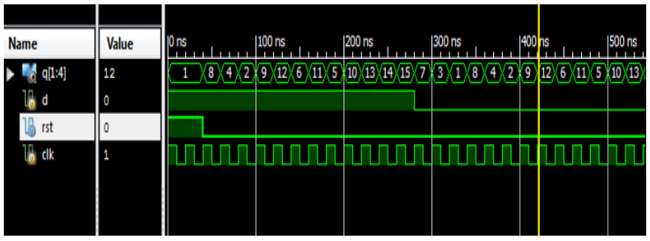

Figure-28:4-Bit LFSR using XOR Logic

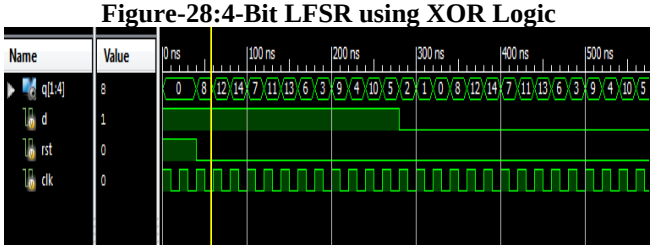

Figure-29:4-Bit LFSR using XNOR Logic

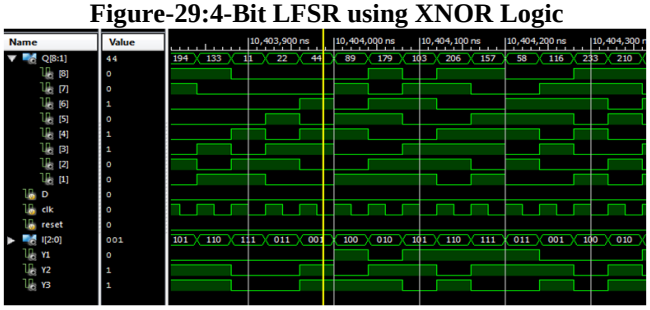

Figure 30 Run Time Polynomial Change based LFSR using XOR Logic

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

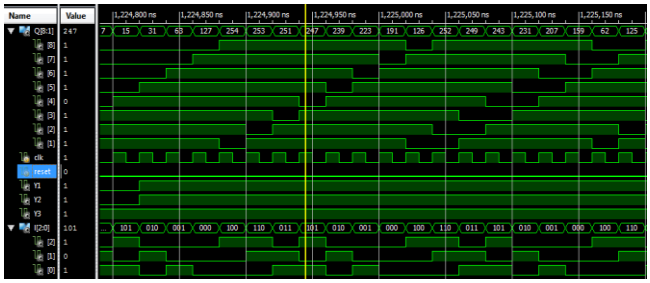

Figure 31 Run Time Polynomial Change based LFSR using XNOR Logic

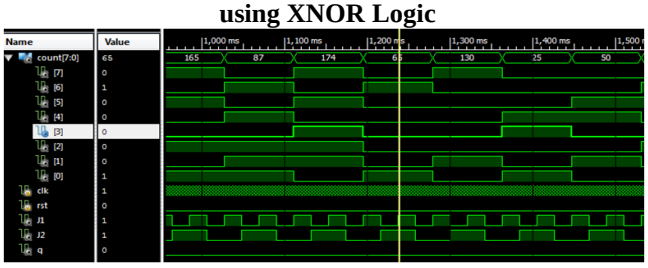

Figure 32 Jitter based LFSR using XOR Logic

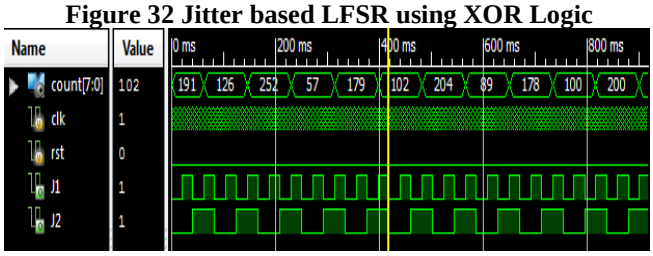

Figure 33 Jitter based LFSR using XNOR Logic

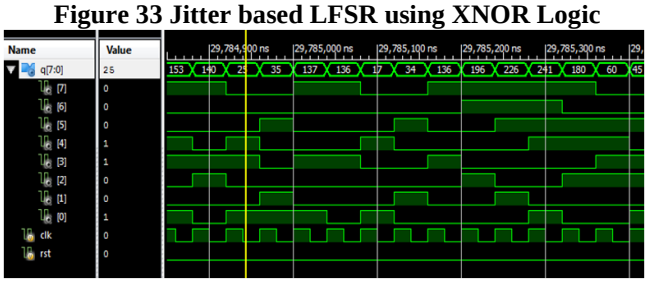

Figure 34 Low Power LFSR using XOR Logic

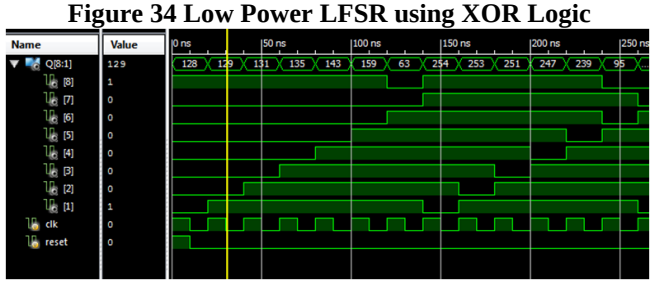

Figure 35 Low Power LFSR using XOR Logic

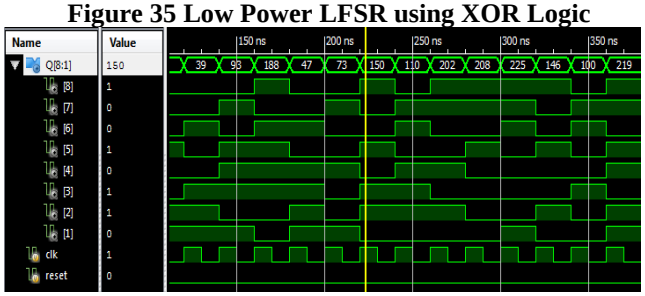

Figure 36 Cellular Automata LFSR using XOR Logic

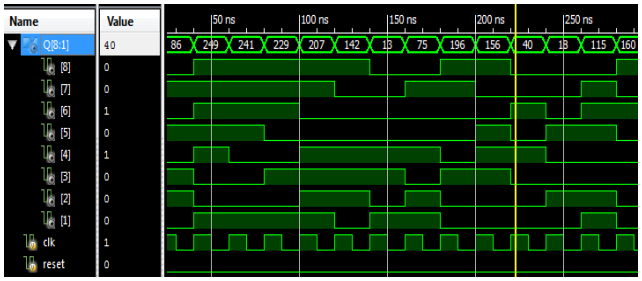

Figure 37 Cellular Automata LFSR using XNOR Logic

The figure 28 represents simulation output for 4-Bit conventional LFSR using XOR gate. Whereas, figure 29 signifies the simulation output for 4-Bit proposed LFSR using XNOR gate. The simulation output of 8-Bit Run Time Polynomial Change based LFSR using XOR logic is shown in figure 30 and XNOR logic in figure 31. The simulation output of 8-Bit Jitter based LFSR using XOR logic is shown in figure 32 and XNOR logic in figure 33. The simulation output of 8-Bit LP LFSR using XOR logic is shown in figure 34 and XNOR logic in figure 35. The simulation output of Cellular Automata LFSR using XOR logic is shown in figure 36 and XNOR logic in figure 37.

VIII. HARDWARE IMPLEMENTATION

Hardware implementation of runtime logic reconfigurable for Spartan3E architecture and partially reconfigurable for Virtex-5 LX110T architecture. A 4-Bit and 8-Bit LFSR is considered, using partial reconfiguration any one of the technique can be configured in run time which gives more randomness.

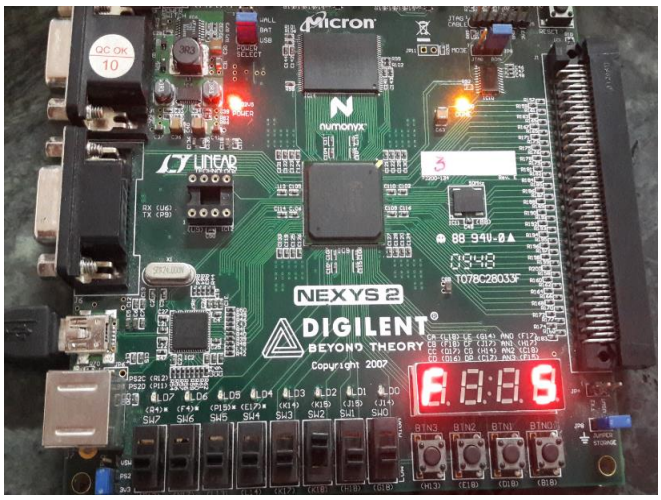

Figure-38: Implementation of 4-Bit LFSR using XOR Logic in Nexys2 FPGA

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

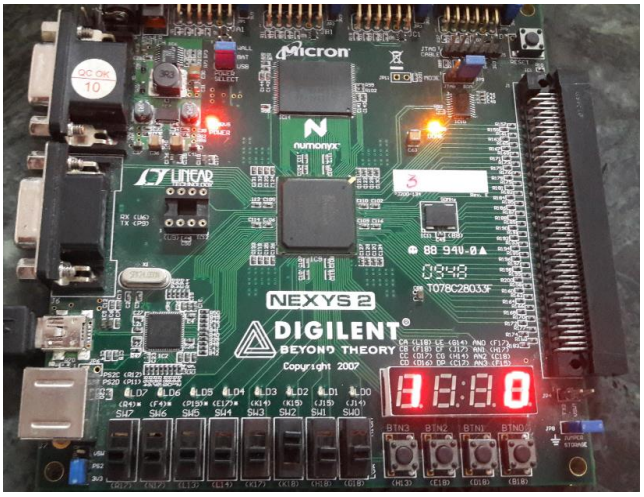

Figure-39: Implementation of 4-Bit LFSR using XNOR Logic in Nexys2 FPGA

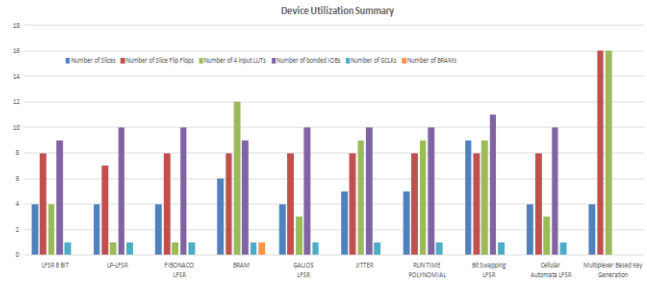

Figure-40: Device Utilization Summary for Different 8-Bit LFSRs

Figure-41: Delay (ns) comparison for Different 8-Bit LFSRs

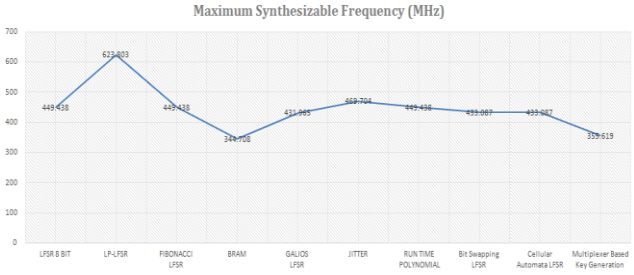

Figure-42: Frequency (MHz) comparison for Different 8-Bit LFSRs

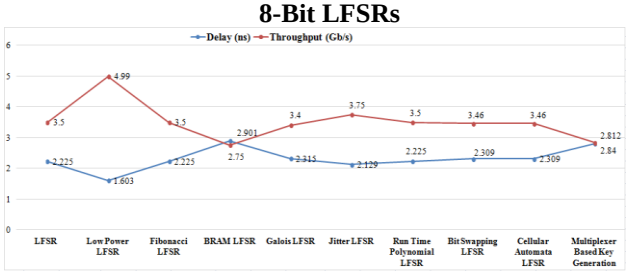

Figure-43: Delay vs. Throughput comparison of various LFSRs for Virtex-5 FPGA Architecture

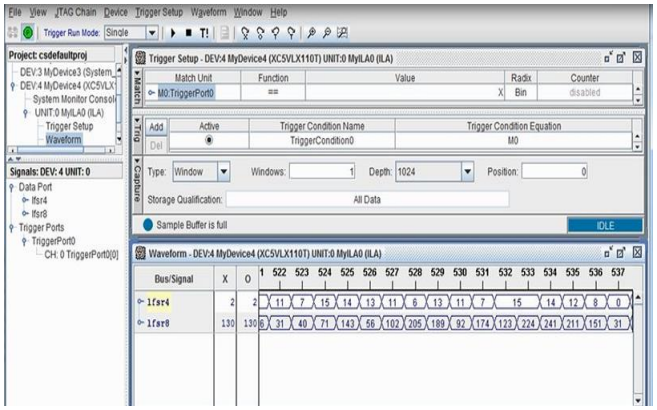

Figure-44: Partially Reconfigurable variable bit length of different LFSRs implementation in Virtex-5 FPGA Architecture

Table 1 shows the resources comparison of different LFSRs, for conventional and proposed. Several parameters like number of LUTs, delay, maximum synthesizable frequency for different device architectures are compared with proposed methodologies. It illustrate that proposed methods furnish better results. The Table 2 shows the series of random numbers generated by various LFSR's using XOR and XNOR logic which shows more randomness among each other.

Table 1 Device Utilization Summary comparison of conventional and proposed Linear Feedback Shift Registers

Journal / Conference	Elsevier Conference	Proposed JARDCS Journal	Hindwai Journal	Proposed	IEEE Conference	Proposed JARDCS Journal	IEEE Conference	Proposed	IEEE Devices for Integrated Circuits	Proposed JARDCS Journal	Springer Journal	Proposed
Year	2012	2017	2012	2016	2012	2017	2014	2016	2017	2017	2014	2016
Method	LFSR	LFSR	CA	CA	Polynomial LFSR	Run Time Polynomial LFSR	Bit Swapping LFSR using BIST	Bit Swapping LFSR using BIST	Multibit LFSR	Multiplexer based LFSR	Clock Controlled LFSR (10Bit)	Jitter Based LFSR
Device	Virtex-6	Virtex-5	Spartan 3E	Virtex-5	Spartan 3	Virtex-5	Virtex6	Virtex-5	Spartan 6	Virtex-5	Virtex-4	Virtex-5
Number of Slice LUT's	78	4	8	3	1	9	280	9	16	16	4	5
Delay (ns)	13.078	2.25	9	2.309	20	2.25	10.92	2.309	1.507	2.812		2.129
Maximum Frequency (MHz)	76.464	449.438	110	433.087	50	449.438	91.5	433.087	663.46	355.619		469.704

Table 2 Series of Random Numbers generated by various LFSR's

Name of the Method	Logic	Random Number Series
Galois LFSR	XOR	139,103,206,237,171,39,78,156,73,146,85,170,37,74
Galois LFSR	XNOR	144,33,50,20,88,192,129,3,118,156,57,2,116
Fibonacci LFSR	XOR	40,81,162,69,139,22,45,91,183,110,220,184,113,22,6
Fibonacci LFSR	XNOR	1,3,7,15,30,61,123,246,236,216,177,99,198,141
Combined LFSR	XOR	71,15,30,60,249,114,228,72,17,34,197,139,150,172
Combined LFSR	XNOR	128,129,131,135,143,159,63,254,253,251,247,239,95
Low Power LFSR	XOR	153,140,25,35,137,136,196,226,241,180,60,45
Low Power LFSR	XNOR	204,89,178,100,200,81,162,68,136,16,225,3,199,79,158,60,185,114,228,9,211
Cellular Automata LFSR	XOR	39,93,188,47,73,150,110,202,208,225,146,100,219
Cellular Automata LFSR	XNOR	86,249,241,229,207,142,13,75,196,156,40,18,115,160
Bit Swapping LFSR	XOR	00110100 01111000 11100000 01000011 10000111 00001110 S = 0 S = 0 S = 0
Bit Swapping LFSR	XNOR	00000010 00000011 00001111 11010011 00000001 00000011 00001111 00111101 S = 1 S = 1 S = 1 S = 0
Jitter LFSR	XOR	165,87,174,66,130,25,50
Jitter LFSR	XNOR	126,252,57,179,102,204,89,178,100,200,81,162,68,136,16
BRAM LFSR	Memory	80,100,151,67,25,55,86,121,51,70,69,9,147,84,114,49,50
Run Time Polynomial Change LFSR	XOR	194,133,11,22,44,89,179,103,206,157,58,116,233,210
Run Time Polynomial Change LFSR	XNOR	7,15,31,63,127,254,253,251,247,239,223,191,126,252,249,243,231,207,159,62,125

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Figure 38 shows the Implementation of 4-Bit conventional LFSR using XOR in Nexys2 FPGA. State "F" is generated using XOR gate on left side of seven segment display. State "0" is generated using XNOR gate on right side of seven segment display as depicted in figure-39. Device Utilization Summary for Different LFSRs using XOR and XNOR is shown in figure 40. Delay (ns) comparison for Different LFSRs using XOR and XNOR is shown in figure 41. Maximum frequency (MHz) for Different LFSRs using XOR and XNOR is shown in figure 42. Figure 43 shows the delay versus throughput comparison of various Linear Feedback Shift Registers for Virtex-5 FPGA Architecture. Figure 44 shows different LFSRs are partially reconfigurable in runtime for variable bit length. Chipscope pro logic analyzer is binded to the design to capture the logic signals of various LFSRs running on Virtex-5 FPGA Architecture.

IX. CONCLUSION

LFSR is most prominently used for several applications. In communication field pseudo-random sequences are generated using LFSR, where as key generation in cryptography applications uses LFSR to encrypt and decrypt the messages. LFSRs used in steganography applications for hiding the data encrypted with random key appends in an image. The proposed LFSR method is designed using XNOR does not require any seed value, whereas XOR based LFSR in conventional method requires initial seed value, to generate random numbers. Multiplexer was replaced with tri-buffers and inverter. Proposed methods improve randomness in comparison with XOR and XNOR as shown in Table-2. Parameters like area, delay, maximum synthesizable frequency and throughput for different LFSRs are compared with conventional approach. The proposed method is applicable to N-Bit, where different LFSRs with XOR and XNOR logic is configured at runtime for variable bit length using dynamic partial reconfiguration.

REFERENCES

1. S. Ergun and S. Ozoguz, Truly Random Number Generators Based on a Non-autonomous Chaotic Oscillator, AEU-International Journal Electronics & Communications,Vol. 61, No. 4, 2007, pp. 235-242.
2. YilongLiao, XiangningFan Mathematical calculation of sequence length in LFSR- dithered MASH digital delta-sigma modulator with odd initial condition, AEU - International Journal of Electronics and Communications Volume 82, December 2017, Pages 533-542.
3. MariosKalyvas, Kostas,Yiannopoulos, Thanassi,s Houbavlis, Hercules Avramopoulos Design Algorithm of All-Optical Linear Feedback Shift Registers AEU - International Journal of Electronics and Communications Volume 57, Issue 5, 2003, Pages 328-332.
4. Efficient Parallel Architecture for Linear Feedback Shift Registers, J. Jung and H. Yoo and Y. Lee and I. C. Park, IEEE Transactions on Circuits and Systems II: Express Briefs, Nov 2015, volume 62, pp.1068-1072.
5. Test vector encoding using partial LFSR reseeding, C. V. Krishna and A. Jas and N. A. Touba, Proceedings International Test Conference, 2001, pp. 885-893.
6. The K-distribution of Generalized Feedback Shift Register Pseudorandom Numbers, Fushimi, M. and Tezuka, S.,Communications of the ACM, July 1983, volume 26, pp. 516--523.
7. MC-DS-CDMA pseudo-noise acquisition algorithm research using computer model, A. D. Zolotuev and F. G. Khisamov and M. V. Milovanov and D. M. Sobachkin, 23rd Telecommunications Forum Telfor (TELFOR), Nov2015, pp.329-332.
8. Low Complexity Wiener Filtering in CDMA Systems Using a Class of Pseudo-Noise Spreading Codes, R. Carvajal and K. Mahata and J. C. Aguero, IEEE Communications Letters, Nov 2012, volume 16, pp.1357-1360.
9. Design and analysis of linear feedback shift register(LFSR) using gate diffusion input(GDI), R. Sharma and B. Singh, 5th International Conference on Wireless Networks and Embedded Systems (WECON), Oct 2016,pp.1-5.
10. Multiple test set generation method for LFSR-based BIST, Youhua Shi, Zhe Zhang, Proceedings of the 2003 Asia and South Pacific Design Automation Conference, Nov 2003, pp. 863-868.
11. Low-Power Programmable PRPG With Test Compression Capabilities,M. Filipek and G. Mrugalski and N. Mukherjee and B. Nadeau-Dostie and J. Rajski and J. Solecki and J. Tyszer, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, June 2015, volume 23, pp.1063-1076.
12. An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA A. P. Johnson, R. S. Chakraborty and D. Mukhopadyay, in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 64, no. 4, pp. 452-456, April 2017.
13. Cellular Automata-Based Parallel Random Number Generators Using FPGAs David H. K. Hoe, Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and Mukul V. ShirvaikarInternational Journal of Reconfigurable Computing Volume 2012, Article ID 219028, 13 pages
14. Design and Implementation of Multibit LFSR on FPGA to Generate Pseudorandom Sequence NumberDebarshi Datta, Bipa Datta, Himadri Sekhar Dutta2017 Devices for Integrated Circuit (DevIC), 23-24 March, 2017, Kalyani, India
15. Low Power Memory Built in Self Test Address Generator Using Clock Controlled Linear Feedback Shift Registers K. Murali Krishna, M. Sailaja Journal of Electronic Testing Issue 1/2014

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)