# Design and Simulation of 8, 16 and 32 bit LFSR with Maximum Length Feedback Polynomial using VHDL

Deepali R.Badre
*IV Semester, M.Tech, Dept of Electronics Engg.(Communication)*
*V.I.T.*
*Nagpur, State-MH, Country-India*
deepalibadre90@gmail.com

Prof. N.P.Bodane
*HOD, Dept of Electronics Engg.(Communication)*
*V.I.T.*
*Nagpur, State-MH, Country-India*
nileshbodane@gmail.com

***Abstract*— LFSR based PN Sequence Generator technique is used for various cryptography applications and for designing encoder, decoder in different communication channel.. LFSRs have long been used as pseudo-random number generators for use in stream ciphers (especially in military cryptography), due to the ease of construction from simple electromechanical or electronic circuits, long periods, and very uniformly distributed output streams. However, an LFSR is a linear system, leading to fairly easy cryptanalysis. Here in this paper we design and simulation for Fibonacci LFSRs and Galois LFSRs with feedback polynomial to study the performance and analysis the behavior of randomness. As FPGAs is used to implement any logical function for faster prototype development, it is necessary to implement the existing design of LFSR on FPGA to test and verify the simulated & synthesis result between different lengths. Also the simulation problem for long bit LFSR on FPGA is presented. It is more important to test and verify by implementing on any hardware for getting better efficient result.**

***Keywords*— LFSR, FPGA, VHDL**

## I. Introduction

For generating data encryption keys, random numbers are very much useful in the various applications such as communication channel, bank security; etc. It is used to design encoder and decoder for sending and receiving data in noisy communication channel. In contrast, the use of feedback shift registers permits very fast generation of binary sequences. Shift register sequences with minimum length feedback polynomial 8, 16 and 32-Bit LFSR based PNRG design and simulation on FPGA. As we change the feedback polynomial the run-length as well randomness also changes. Here we have to design 16 Bit Fibonacci LFSR and Galois LFSR on FPGA using VHDL with maximum length feedback polynomial to understand the memory utilization and speed requirement. Also we have presented the comparison of performance analysis based on synthesis and simulation result as well identify the simulation problem for long bit LFSR.

FPGA is a predesigned reconfigurable IC. The FPGA configuration is generally defined using a hardware description language (HDL), similar to that used for an application specific integrated circuit (ASIC).The HDLs are VHDL and Verilog. We prefer VHDL for programming because of its widely in use. FPGAs can be used to implement any logical function that an ASIC can perform. Because of various advantages and rapid prototype development can possible, so FPGA is chosen here. The programming of the FPGA is done using a logic circuit diagram or a source code using a Hardware Description Language (HDL) to specify how the chip should work. FPGAs have programmable logic components called ‚logic blocks', and a hierarchy or reconfigurable interconnects which facilitate the ‚wiring' of the blocks together. The programmable logic blocks are called configurable logic blocks and reconfigurable interconnects are called switch boxes. Logic blocks (CLBs) can be programmed to perform complex combinational functions, or simple logic gates like AND and XOR. In most FPGAs the logic blocks also include memory elements, which can be as simple as a flip-flop or as complex as complete blocks of memory.

## II. Linear Feedback Shiet Register

LFSR is a Linear Feedback 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. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences.

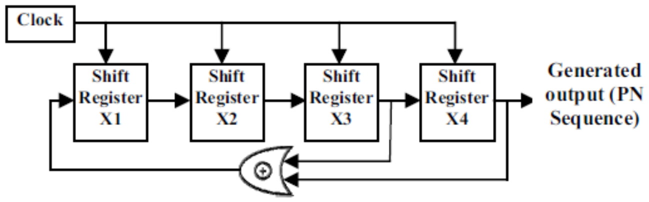

Fig. 1 Basic block diagram of 8 Bit LFSR

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.A maximum length of LFSR produces an m sequence. (Possible $2^n$ -1 state). For 8 bit, example if the taps are at the 7th and 6th bits , the feedback polynomial is $x^7+x^6+1$.

## III. Proposed Work

*A.* First select the feedback polynomial. The 'one' in the polynomial does not correspond to a tap it corresponds to the input to the first bit. 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. LFSR will only be maximum-length if the number of taps is even. There must be no common divisor to all taps. Possible valid feedback polynomial for LFSR is given on table 1.

TABLE I. POSSIBLE AND MAXIMUM LENGTH POLYNOMIAL

| Size of LFSR | Possible Feedback Polynomial | Maximum Length Feedback polynomial |
|---|---|---|
| 8 Bit | $X^8+X^7+1$, $X^8+X^5+1$, $X^8+X^7+X^6+X^5+1$, $X^8+X^6+X^4+X^3+X^2+X^1+1$, etc | $X^{16}+X^{14}+X^{13}+X^{11}+1$ |
| 16 Bit | $X^{16}+X^{15}+1$, $X^{16}+X^{13}+X^{12}+X^9+1$, $X^{16}+X^{11}+X^{10}+X^7+X^3+X^1+1$, $X^{16}+X^{15}+X^{14}+X^{12}+X^7+X^6+ +X^3+X^2+1$, etc | $X^{16}+X^{14}+X^{13}+X^{11}+1$ |
| 32 Bit | $X^{32}+X^{31}+1$, $X^{32}+X^{28}+X^{27}+X^9+1$, $X^{32}+X^{21}+X^{15}+X^{13}+X^{12}+X^{10}+ +X^8+X^4+1$, $X^{32}+X^{31}+X^{27}+X^{24}+X^{19}+X^{18}+ +X^{17}+X^{14}+ X^{13}+X^{11}+X^5+X^4 +X^1$, etc | $X^{32}+X^{22}+X^2+X^1+1$ |

*B. Design of LFSRs*

The bits in the LFSR state which influence the input are called taps. A maximum-length LFSR produces an msequence (i.e. it cycles through all possible 2n -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. The sequence of numbers generated by this method is random. The period of the sequence is (2n - 1), where n is the number of shift registers used in the design.

- For 8 bit, example if the taps are at the 7th and 6th bits (as shown),the feedback polynomial is $x^7+x^6 +1$

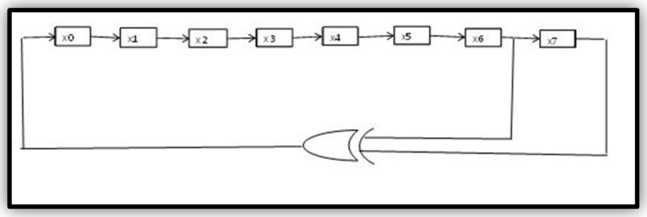

Fig 2: 8 Bit LFSRs

*C. Design of Fibonacci LFSRs*

The bit positions that affect the next state are called the taps. In the diagram the taps are [7,6,4 and 3]. 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 2n − 1 states) As an alternative to the XOR based feedback in an LFSR, one can also use XNOR.[1] 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. 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.

- For 8 bit, example if the taps are at the 7th, 6th, 4th and 3rd bits (as shown),the feedback polynomial is $x^7+x^6+x^4+x^3+1$

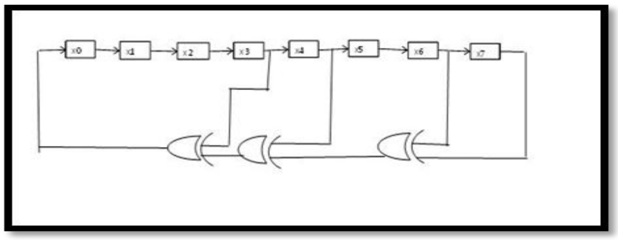

Fig 3: 8 Bit Fibonacci LFSRs

The LFSR is maximal-length if and only if the corresponding feedback polynomial is primitive. This means that the following conditions are necessary (but not sufficient): The number of taps should be even. 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 divisor other than 1 common to all taps.

*D. Design of Galois LFSRs*

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 (but offset in time).[2] In the Galois configuration, when the system is clocked, bits that are not taps are shifted one position to the right unchanged. 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), and then the entire register is shifted to the right and the input bit becomes 1. 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.

- For 8 bit, example if the taps are at the 7th, 6th, 4th and 3rd bits (as shown),the feedback polynomial is
  $x^7+x^6+x^4+x^3+1$

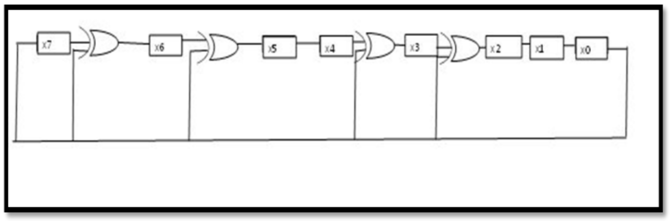

Fig 4: 8 Bit Galois LFSRs

The Galois register shown has the same output stream as the Fibonacci register in the first section. A time offset exists between the streams, so a different start point will be needed to get the same output each cycle. 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.

## III. Synthesis And Simulation

To design and simulation for 8,16 and 32 bit LFSR, Fibonacci LFSRs and Galois LFSRs with maximum length feedback polynomial In this design, we describe the RTL-level of the LFSR pseudo-random number generator for 8,16 or 32-bit using VHDL language, and use the Xilinx's chip XC3S 1000 Sparta3 as the target chip.

### *A. Simulation Result of 8,16 and 32 bit LFSR, Fibonacci LFSRs and Galois LFSRs*

The simulation waveform for 8, 16 and 32 bit LFSR, Fibonacci LFSRs and Galois LFSRs are shown in Fig. 5,6,7,8,9,10,11,and 12.

### *B. Synthesis Result and comparison between 8,16 and 32 bit LFSR, Fibonacci LFSRs and Galois LFSRs*

The synthesis and simulation report for Fibonacci LFSRs and Galois LFSRs by using feedback polynomial are given in Table 2. Form the table we can find the total logic elements, total registers for Fibonacci LFSRs and Galois LFSRs.

TABLE II. SIMULATION AND SYNTHESIS RESULT

| **Performance** | **8 Bit LFSRs** | **8 Bit Fibonacci LFSRs** | **8 Bit Galois LFSRs** |
|---|---|---|---|
| Total Logic Elements | 16 | 17 | 11 |
| Dedicated Logic Registers | 16 | 16 | 11 |
| Total Registers | 16 | 16 | 11 |
| Total Pins | 11 | 11 | 11 |
| Total Virtual Pins | 0 | 0 | 0 |
| Total power dissipation | 67.96mW | 67.96mW | 67.96mW |
| Clockto destination | 9.253 ns | 9.207 ns | 9.844 ns |
| Throughput | | | |
| Clock Setup: 'clk' | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) |

| **Performance** | **16 Bit LFSRs** | **16 Bit Fibonacci LFSRs** | **16 Bit Galois LFSRs** |
|---|---|---|---|
| Total Logic Elements | 32 | 34 | 26 |
| Dedicated Logic Registers | 32 | 32 | 26 |
| Total Registers | 32 | 32 | 26 |
| Total Pins | 19 | 19 | 19 |
| Total Virtual Pins | 0 | 0 | 0 |
| Total power dissipation | 68.83mW | 68.83mW | 68.85mW |
| Clock to destination | 8.247 ns | 9.831 ns | 9.294 ns |
| Throughput | | | |
| Clock Setup: 'clk' | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) |

| **Performance** | **32 Bit LFSRs** | **32 Bit Fibonacci LFSRs** | **32 Bit Galois LFSRs** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Total Logic Elements | 64 | 64 | 57 |
| Dedicated Logic Registers | 64 | 64 | 57 |
| Total Registers | 64 | 64 | 57 |
| Total Pins | 35 | 35 | 35 |
| Total Virtual Pins | 0 | 0 | 0 |
| Total power dissipation | 70.68mW | 70.68mW | 70.70mW |
| Clock to destination | 10.150 ns | 9.207 ns | 7.949ns |
| Throughput | | | |
| Clock Setup: 'clk' | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) | Restricted to 380.08 MHz ( period = 2.631 ns ) |

Synthesis report from Modelsim SE 6.3f and Quartus II

## IV. CONCLUSION

It is clearly found from the synthesis and simulation result that 8, 16 and 32 bit LFSR, Fibonacci LFSRs and Galois LFSRs with feedback polynomial can generate the maximum random output. The LFSR and Fibonacci LFSR generating the random output but Galois LFSRs generate the alternate output. Therefore, for design LFSR, the Galois LFSR is more secure than Fibonacci LFSR. As well as it required less number of Total logic element, Total combinational functions, and dedicated logic register and Total registers.

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. LFSR is sufficient for different cryptographic applications, as well as use in counters, circuit testing, digital broadcasting and communications etc.

## REFERENCES

[1] F. James, "A Review of Pseudo-random Number Generators," *Computer Physics Communications* 60, 1990.

[2] Jiang Hao, Li Zheying, "On the Production of Pseudo-random Numbers in Cryptography" in *Journal Of Changzhou Teachers College of Technology*, Vo1. 7 , No. 4, Dec. 2001.

[3] C. Li and B. Sun, "Using linear congruential generators for cryptographic purposes", *In Proceedings of the ISCA 20th International Conference on Computers and Their Applications*, pp. 13-18, March 2005.

[4] Madhusudan Dey, Abhishek Singh, "Design and IP core based implementation of a programmable 8-bits random sequence generator, "*In Proceedings of the International Symposium on Nuclear Physics*, 2009, pp.678-679.

[5] Katti, R.S. Srinivasan, S.K., "Efficient hardware implementation of a new pseudo-random bit sequence generator" *IEEE International Symposium on Circuits and Systems, 2009. ISCAS 2009*.

[6] Ding Jun, Li Na, Guo Yixiong, "A high-performance pseudo random number generator based on FPGA" *2009 International Conference on Wireless Networks and Information Systems*.

[7] Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and David H. K. Hoe, " Random Number Generators using Cellular Automata Implemented on FPGAs" *44th IEEE Southeastern Symposium on System Theory University of North Florida*, Jacksonville, FL March 11-13, 2012.

[8] Je-Hoon Lee1 and Seong Kun Kim2, "Segmented Leap-Ahead LFSR Architecture for Uniform Random Number Generator" *International Journal of Software Engineering and Its Applications* Vol.7, No.5 (2013), pp.233-242.

[9] Sarmad Fakhrulddin Ismael and Dr. Basil Shukr Mahmood, "Architectural Design of Random Number Generators and Their Hardware Implementations" *Al-Rafidain Engineering* Vol.22 No. 2 March 2014.

[10] Efficient Shift Registers, LFSR Counters, and Long Pseudo Random Sequence Generators, *Application Note*, Xilinx Inc.

[11] K. Chandra Sekhar, K. Saritha Raj, "An Efficient Pseudo Random Number Generator for Cryptographic Applications" *International Journal of Engineering and Advanced Technology* (IJEAT) ISSN: 2249 – 8958, Volume-4 Issue 1, October 2014.

[12] Shruti Hathwalia, Meenakshi Yadav, " Design and Analysis of a 32 Bit Linear Feedback Shift Register Using VHDL" *Shruti Hathwalia Int. Journal of Engineering Research and Applications* ISSN : 2248-9622, Vol. 4, Issue 6( Version 6), June 2014, pp.99-102.

[13] Brown S., Vranesic Z "Fundamental of Digital Logic Design with VHDL" McGraw Hill, 2nd Edition.

[14] Xilinx, Inc. Xilinx Libraries Guide, 2011.

[15] Xess Corp.. XSA-3S1000 Board V1.1 User Manual. Available: http://xess.com/manuals/xsa-3S-manual-v1_1.pdf. Sept 2007.

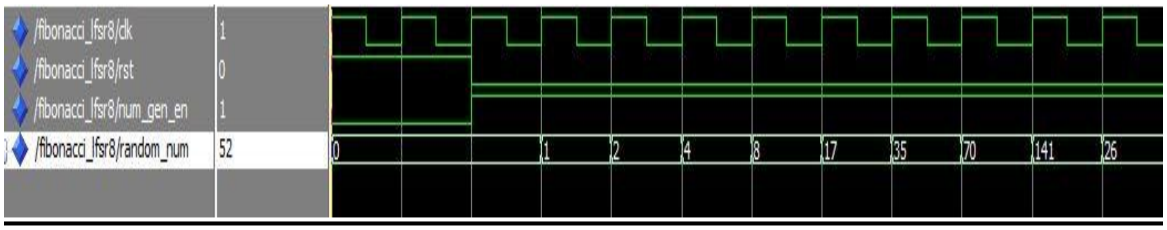

Fig 5: Simulation Result of 8 bit Fibonacci LFSR

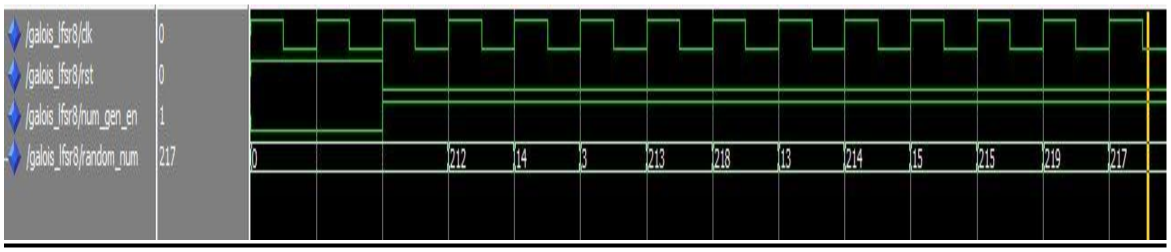

Fig 6: Simulation Result of 8 bit Galois LFSR

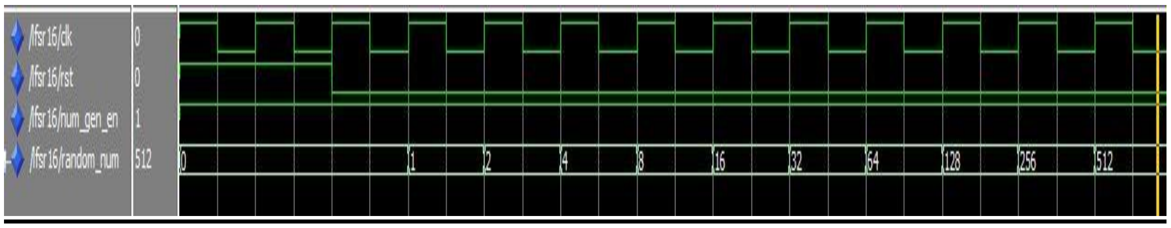

Fig 7: Simulation Result of 16 bit LFSR

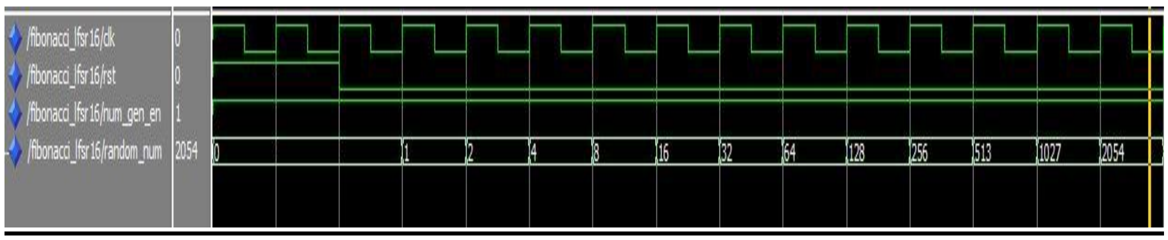

Fig 8: Simulation Result of 16 bit Fibonacci LFSR

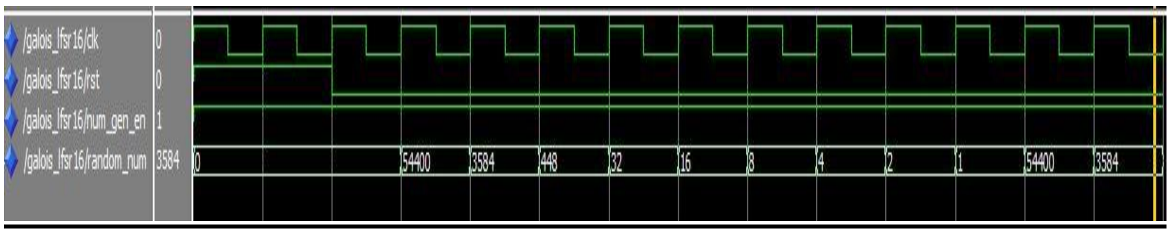

Fig 9: Simulation Result of 16 bit Galois LFSR

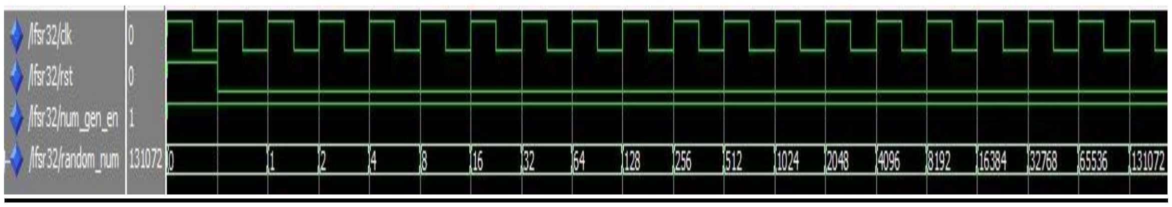

Fig 10: Simulation Result of 32 bit LFSR

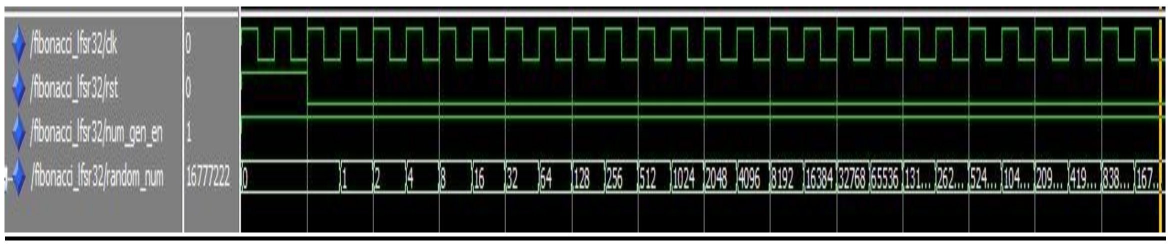

Fig 11: Simulation Result of 32 bit Fibonacci LFSR

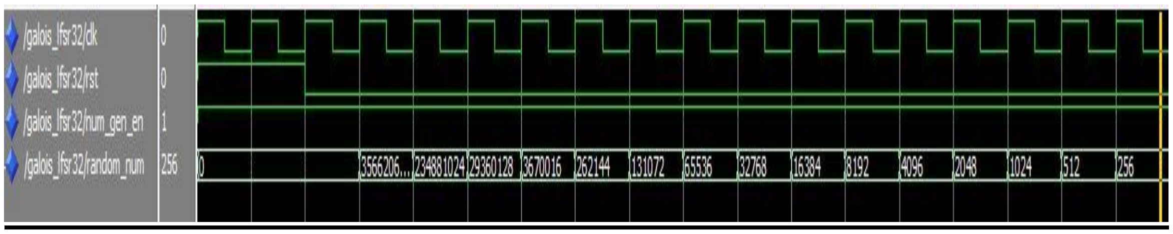

Fig 12: Simulation Result of 32 bit Galois LFSR