International Journal of Engineering Research and General Science Volume 3, Issue 3, Part-2 , May-June, 2015
ISSN 2091-2730

FPGA BASED N-BIT LFSR TO GENERATE RANDOM SEQUENCE NUMBER

Babitha P K, Thushara T, Dechakka M P
Assistant Professors
Dept. of ECE, Coorg Institute of Technology
E-mail Id: dechakka.mp@gmail.com

Abstract— LFSR based PN Sequence Generator technique is used for various cryptography applications and for designing encoder, decoder in different communication channel. It is more important to test and verify by implementing on any hardware for getting better efficient result. 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. The total number of random state generated on LFSR depends on the feedback polynomial.

As it is simple counter so it can count maximum of 2^n -1 using maximum feedback polynomial. Here in this paper we implemented n -bit LFSR on FPGA by using VHDL to study the performance and analyze the behavior of randomness. The analysis is conceded out to find number of gates, memory and speed requirement in FPGA as the number of bits is increased. The comparative study of 8, 16 and n bit LFSR on FPGA is shown here to understand the on chip verification. Also the simulation problem for long bit LFSR on FPGA is presented.

Keywords— **LFSR, FPGA, PRNG, VHDL, FPGA, ASIC, RTL**

INTRODUCTION

For generating data encryption keys, random numbers are very much useful in the various applications such as communication channel, bank security, etc [1], [3]. it is used to design encoder and decoder for sending and receiving data in noisy communication channel. They have also been used aesthetically, for example in literature and music, and are of course ever popular for games and gambling [1]. When discussing single numbers, a random number is one that is drawn from a set of possible values, each of which is equally probable, i.e. a uniform distribution. Random number generator is a computational device to generate a sequence of numbers or that lack any pattern [2]-[5]. There are various methods for pseudo-random numbers are known [1], [3]. Most of them are based, on linear congruential equations [8], [9] and require a number of time consuming arithmetic operations. In contrast, the use of feedback shift registers permits very fast generation of binary sequences. Shift- register sequences of maximum length (m-sequences) are well suited to simulate truly random binary sequences [6], [7], [10]. With minimum length feedback polynomial 8, 16 and n-Bit LFSR based PRNG implemented on FPGA explain in [11], [12]. As we change the feedback polynomial the run-length as well randomness also changes. Here we have implemented n bit length sequences 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. The target device we have used Xilinx Spartan 3S 1000 FPGA and XSA 3S1000 Board of Xess Corporation [17] and performed simulation and synthesis using Xilinx ISE 10.1 [18].

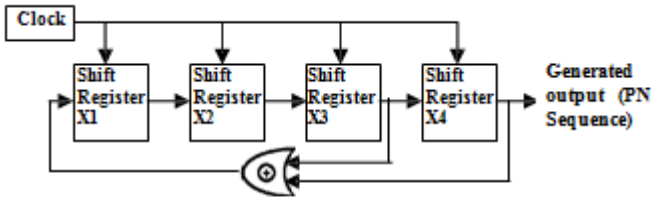

Figure 1. Basic block diagram of LFSR

FPGA is a predesigned reconfigurable IC [15]. It can be reconfigured any number of times according to the speciation of design. The FPGA configuration is generally defined using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) [14], [15]. The HDLs are VHDL and Verilog. We prefer VHDL for programming because of its widely in use [15], [16]. 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.

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. [1], [10]. The initial value of the LFSR is called the seed. 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. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common [11].

A. Implementation of LFSR based PRNG

Pseudo random number sequence generator is generated in VHDL according to the following circuit based on the concept of shift register.The bits in the LFSR state which influence the input are called taps. 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. The sequence of numbers generated by this method is random. The period of the sequence is (2^n - 1), where n is the number of shift registers used in the design. For 32 bit design the period is 4294967295. This is large enough for most of the practical application. 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 32^{nd}, 30^{th}, 11^{th} and 5^{th} bits, then the feedback polynomial is $X^{32} + X^{30} + X^{11} + X^5 + 1$

B. The rules for selecting feedback polynomial

The rules for selecting feedback polynomial are given in [11], [12]. 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 and maximum length feedback polynomial [10], [13] for 8, 16 and 32 bit LFSR are 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	$X^{16} + X^{14} + X^{13} + X^{11} + 1$
n 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^n+..... $X^{32} + X^{22} + X^2 + X^1 + 1$

SYNTHESIS AND SIMULATION

In this design, we describe the RTL-level of the LFSR pseudo-random number generator for 8-bit, 16-bit and n bit using VHDL language, and use the Xilinx's chip XC3S 1000
Sparta3 as the target chip. Then we synthesize, place and route on the Xilinx ISE platform. Finally we use ISE Simulator to do a timing simulation

A. Timing Simulation

The simulation waveform for 8-bit, 16-bit and n-bit are under the simulation clock frequency 371.747 MHz with 20 ns simulation clock period.

B. Design of 8-Bit LFSR

8-bit LFSR with maximum length feedback polynomial $X^8 + X^6 + X^5 + X^4 + 1$ generates $2^8 - 1 = 255$ random outputs, which is verified from the simulation waveform.

The circuit diagram for 8-bit LFSR with maximum length polynomial is shown in Fig. 2. In The timing simulation from 40 ns to 5140 ns. After this clock time the random output is repeating again.

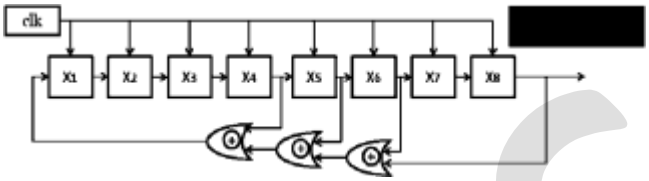

Figure 2. Circuit Diagram of 8- Bit LFSR with maximum length Feedback Polynomial $X^8 + X^6 + X^5 + X^4 + 1$

C. Design of 16-Bit LFSR

16-bit LFSR with maximum length feedback polynomial $X^{16} + X^{14} + X^{13} + X^{11} + 1$ generates $2^{16} - 1 = 65535$ random outputs, which is verified from the simulation waveform.

The circuit diagram for 16-bit LFSR with maximum length polynomial is shown in Fig. 3. In the timing simulation from 20 ns to 1310720 ns. It shows starting simulation and simulation at the end of cycle after which the sequence starts repeating again.

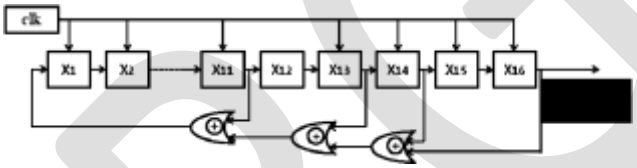

Figure 3. Circuit Diagram of 16- Bit LFSR with maximum length Feedback Polynomial $X^{16} + X^{14} + X^{13} + X^{11} + 1$

D. Design & Simulation of n-Bit LFSR

N-bit LFSR with maximum length feedback polynomial $X^n + X^{32} + X^{22} + X^1 + 1$ for which $2^n - 1 = 429,49,67,295$ random outputs, which is verified from the simulation waveform.

The circuit diagram for n-bit LFSR with maximum length polynomial is shown in Fig. 4. The timing simulation is shown in Fig. 7.a starting from 20 ns to 85899345920 ns (85.9 sec) and we can observe here the simulation is running for a long time to complete the sequence. In the Fig. 7.b a small zooming portion is shown and it can be observed the randomness behaviour for 32 bit LFSR from 30225 ns to 30500 ns. For 32 bit LFSR using Xilinx ISE 10.1 simulator, it is taking about 3 hour duration for simulating upto 1 sec time duration with 20 ns clock period. As the run length is very large which is 429,49,67,295 random states, so it is taking actual 85.9 sec to complete the sequence but practically simulate with ISE 10.1 is taking about 10-12 days.

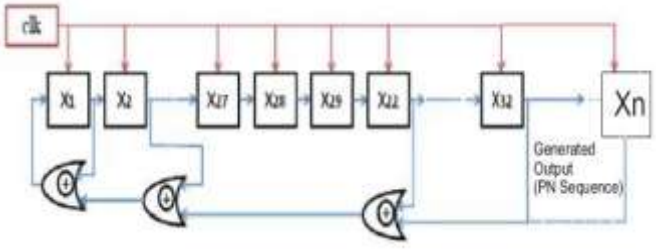

Figure 4. Circuit Diagram of n- Bit LFSR for maximum length Feedback Polynomial **X ^ n + … . X ^ 3 2 + X ^ 2 2 + X ^ 2 + X ^ 1 + 1**

E. Synthesis Result and comparison between 8, 16, n bit LFSR

The synthesis and simulation report for 8, 16 and 32 bit LFSR by using maximum length feedback polynomial are given in Table 2. Form the table we can find the total memory usage and simulation time of different length LFSR.

TABLE II. SIMULATION AND SYNTHESIS RESULT

Performance	8 Bit	16 Bit	n Bit
Time to complete the total states	40 ns to 5140 ns =5100 ns	20 ns to 1310720 ns = 1310.7 us	20 ns to 85899345920 ns = 85.9 sec
Total no. of Random States generating	255	65535	429,49,67,295
Clock	20 ns	20 ns	20 ns
Shift Register	08	16	32
Xor gate	01	01	01
Number of Slices	04	09	18
No. of Slice Flip Flops	08	16	32
No. of 4 i/p LUT	01	01	01
Total memory usage	185904 kb	185904 kb	185904 kb
GCLK	01	01	01
(Gate + Net) Delay	7.271 ns	7.271ns	7.271ns
Total pin	10	18	34

Synthesis report for the targeted device Xilinx Spartan 3S1000 FPGA

CONCLUSION

It is clearly found from the synthesis and simulation result that 8 bit 16 bit and n bit LFSR with maximum feedback polynomial can generate maximum random output. The n bit LFSR takes a lot of simulation time 85.9 sec with 20 ns clock period for generating 429,49,67,295 random output but practically it takes 10-12 days to complete the sequence by using Xlinx ISE 10.1 Simulator. So there is a simulation problem for long bit LFSR when it is targeting to FPGA for rapid prototyping development. Also we can find the memory utilization is same for all three LFSR. Definitely n bit LFSR with maximum length feedback polynomial will generate large sequence which is more secure than other but because of simulation difficulties modification in long bit LFSR is needed. In the practical use 8-bit and 16-bit LFSR is sufficient for different cryptographic applications.

REFERENCES:

[1] M. Luby, Pseudorandomness and Cryptographic Applications, Princeton University Press, 1996.

[2] 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*.

[3] 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.

[4] D. E. Knuth, "The Art of Computer Programming", Vol. 2: Seminumerical Algorithms. Reading, MA: Addison-Wesley, 1969.

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

[6] P. L'Ecuyer, "Random Numbers for Simulation," *Comm. ACM*, 33:10, 1990.

[7] 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*.

[8] 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.

[9] L'Ecuyer, Pierre, "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure," *Mathematics of Computation*, Vol. 68, No. 225, 1999, Pages 249-260.

[10] Goresky, M. and Klapper, A.M. Fibonacci and Galois representations of feedback-with-carry shift registers, *IEEE Transactions on Information Theory*, Nov 2002, Volume**:** 48, On page(s): 2826 – 2836.

[11] Panda Amit K, Rajput P, Shukla B, "Design of Multi Bit LFSR PNRG and Performance comparison on FPGA using VHDL", *International Journal of Advances in Engineering & Technology (IJAET), Mar 2012*, Vol. 3, Issue 1, pp. 566-571.

[12] Sewak K, Rajput P, Panda Amit K, "FPGA Implementation of 16 bit BBS and LFSR PN Sequence Generator: *A Comparative Study*", In Proce. of the *IEEE Student Conference on Electrical, Electronics and Computer Sciences 2012*, 1-2 Mar 2012, NIT Bhopal, India.

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

[14] Jiang Hao, Li Zheying, "FPGA design flow based on a variety of EDA tools" in *Micro-computer information*,

2007(23)11-2:201-203.

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

[16] Bhasker J, "*A VHDL Primer*", P T R Prentice Hall, Pages 1-2, 4-13, 28-30