DESIGN OF MULTI BIT LFSR PNRG AND PERFORMANCE COMPARISON ON FPGA USING VHDL

Amit Kumar Panda, Praveena Rajput, Bhawna Shukla
Department of Electronics & Communication Engineering, Guru Ghasidas University
Bilaspur, Chattishgarh

ABSTRACT

The main purpose of this paper is to study the FPGA implementation and performance analysis of 8, 16, and32 bit LFSR pseudo random number generator system. We have used FPGA to explain how FPGA's ease the hardware implementation part of communication systems. 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 32 bit LFSR on FPGA is shown here to understand the on chip verification. Recently the field programmable gate arrays have enjoyed wide spread use due to several advantages related to relatively high gate density, short design cycle and low cost. The greatest advantage of FPGA's are flexibility that we reconfigured the design many times and check the results and verify it on-chip for comparing with others PN sequence generators. The logic of PN Sequence Generator presented here can be changed any time, if we want a PN generator of more length all we need to do is change the number of shift register and adjust the taps. In this paper we have used one XOR operation for taping. Also we can use XNOR operation for taping. By increasing the number of tapping we can generate more randomness in the sequence.

KEYWORDS: *LFSR, FPGA, PRNSG, VHDL*

I. INTRODUCTION

Random numbers are useful for a variety of purposes such as generating data encryption keys, simulating and modelling complex phenomena and for selecting random samples from larger data sets which are more explain in [1]-[4]. They have also been used aesthetically, for example in literature and music, and are of course ever popular for games and gambling. 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. When discussing a sequence of random numbers, each number drawn must be statistically independent of the other. Random number generator is a computational device to generate a sequence of numbers or that lack any pattern. Random number generator may be software based or Hardware-based systems [5], [6], [8]. Hardware-based systems for random number generation are widely used, but often fall short of this target, though they may meet some of the statistical tests for randomness to confirm that they do not have any easily decipherable patterns. Methods for generating random results have existed since ancient times, including dice, coin flipping, the shuffling of playing cards, the use of yarrow stalks, and many other techniques. Various methods to compute pseudo-random numbers are known [2], [7]. Most of them are based, on linear congruential equations [9], [10] 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 [11]. However, they cannot be used in applications which require a decimal random sequence. This paper presents a fast random generator which is based on shift-register sequences and which generate decimal representation of binary digits in a pseudo-random mode.

A **field-programmable gate array** (**FPGA**) is an integrated circuit created to be configured by the customer after manufacturing hence "field-programmable". The FPGA configuration is generally defined using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC can perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design offer advantages for many applications. The most common HDLs are VHDL and Verilog. Though these two languages are similar but we prefer VHDL for programming because of its widely in use ([15]-[18]). Here we implemented 8, 16 and 32 bit LFSR using double taping as in the feedback path on Spartan 3S1000 FPGA of Xilinx using Xilinx ISE 10.1 [19]. As Xilinx provides automation tools for designing and implementing any logical as well as hardware on a single chip to get faster prototype, so it is widely used to implement any digital logic on FPGA.
The structure of this paper is as follows. Section II contains a brief description of the Linear Feedback Shift Register and Rules for selecting feedback polynomial. Section III shows the Synthesis and timing simulation result and also performance comparison between 8, 16 and 32 bit LFSR using double tapping is presented.

II. LINEAR FEEDBACK SHIFT REGISTER

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is XOR. Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. Xor shift is a category of pseudorandom number generators designed by George Marsaglia. It repeatedly uses the transform of exclusive or on a number with a bit shifted version of it [11], [12]. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. 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.

2.1. Implementation of PRNSG

Pseudo random number sequence generator is generated in VHDL according to the following circuit based on the concept of shift register.

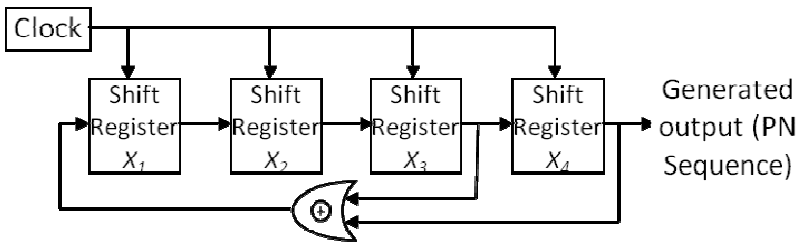

Figure 1. Single Taping LFSR

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 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is

$$X^{16} + X^{14} + X^{13} + X^{11} + 1$$

2.2. The rules for selecting Feedback Polynomial

The rules for selecting feedback polynomial [11] are as follows

- The 'one' in the polynomial does not correspond to a tap it corresponds to the input to the first bit (i.e. x_0 which is equivalent to 1).
- The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and output tap respectively.
- The LFSR will only be maximum-length if the number of taps is even; just 2 or 4 taps can suffice even for extremely long sequences.
- The set of taps taken all together, *not* pairwise (i.e. as pairs of elements) must be relatively prime. In other words, there must be no common divisor to all taps.
- Once one maximum-length tap sequence has been found, another automatically follows. If the tap sequence, in an n-bit LFSR, is [n, A, B, C, 0], where the 0 corresponds to the $x0 = 1$ term, then the corresponding 'mirror' sequence is [n, $n - C$, $n - B$, $n - A$, 0]. So the tap sequence [32, 7, 3, 2, 0] has as its counterpart [32, 30, 29, 25, 0]. Both give a maximum-length sequence.

III. SYNTHESIS AND SIMULATION

In this design, we describe the RTL-level of the LFSR pseudo-random number generator for 8-bit, 16-bit and 32 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 [19].

3.1. Timing Simulation

Input the netlist file generated from synthesizing, placing and routing on the Xilinx ISE 10.1 software and the standard delay file, the simulation waveform is shown in Fig. 2, Fig. 3, Fig. 4 under the simulation clock is 371.747MHz. Although there are a certain delay and glitch phenomena in the waveform, but they don't affect the normal operation of the whole module. So the test results are correct on the function and timing**.** Here the simulation clock period is taken as 20 ns. Simulation waveform for 8-bit, 16-bit and 32-bit are shown in Fig. 2, Fig. 3, Fig. 4.

3.1.1 Simulation Result of 8 bit LFSR

The actual random output for 8-bit LFSR should come $2^8 = 256$. But from the waveform simulation analysis it is found 63 random states is coming in 8-bit LFSR using single taping which is shown in Fig. 2. The timing simulation is shown from 20 ns to 1280 ns. After this time clock the random output is repeating again.

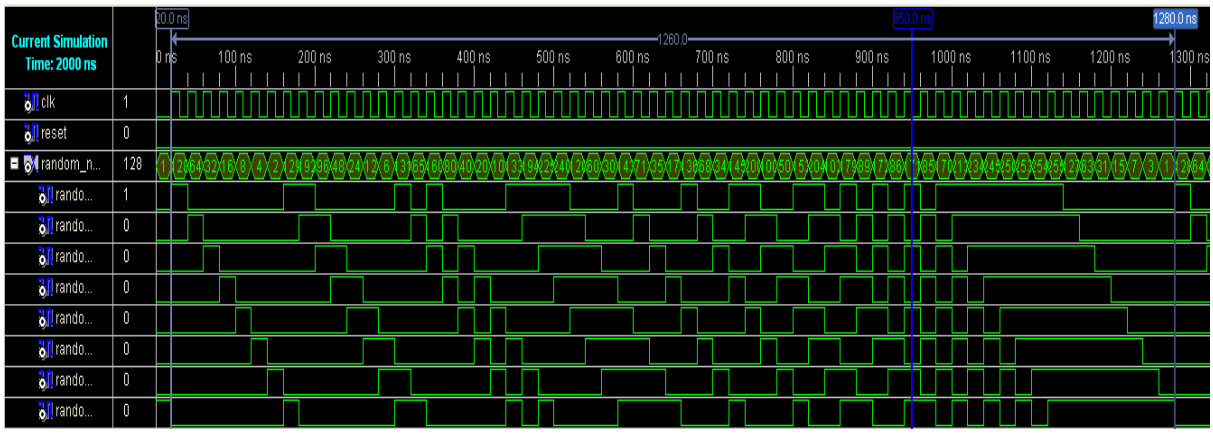

Figure 2. Timing Simulation waveform of LFSR-8 Bit (20 ns to 1280 ns): Total 63 periods

3.1.2. Simulation Result of LFSR 16-Bit

The actual random output for 16-bit LFSR should come 2^{16}-1 = 65535. But from the waveform simulation analysis it is found 255 random states is generating in 16-bit LFSR using single taping which is shown in Fig. 2. The timing simulation is shown from 20 ns to 5120 ns. After this time clock the random output is repeating again.

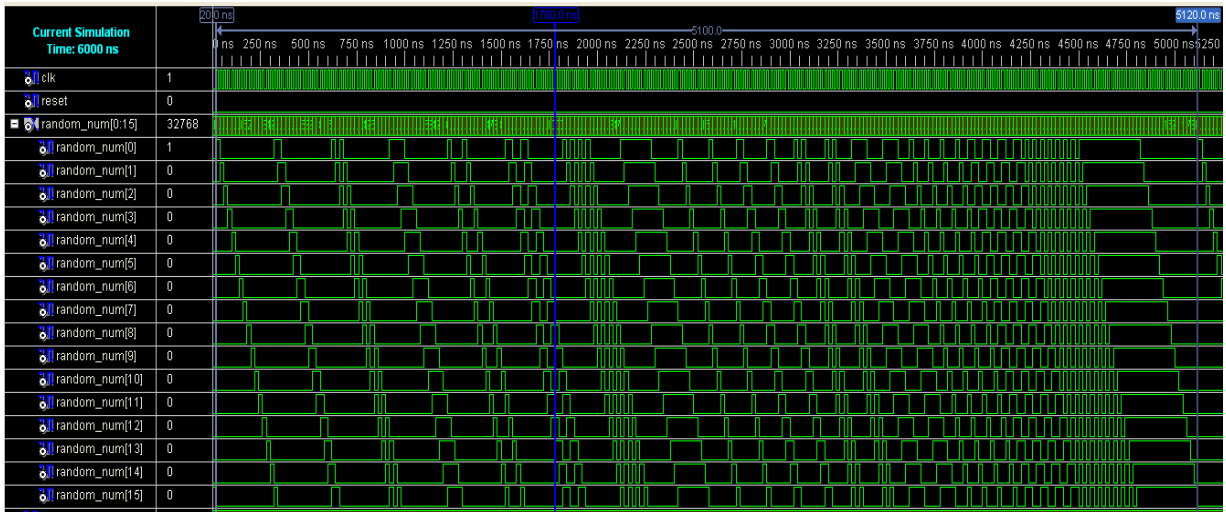

Figure 3. Timing Simulation waveform of LFSR-16 Bit (20 ns to 5120 ns): Total 255 periods

3.1.3. Simulation Result of LFSR 32-Bit

The actual random output for 32-bit LFSR should come 2^{32}-1= 4,29,49,67,295. But from the waveform simulation analysis it is found 2046 random states is generating in 32-bit LFSR using single taping which is shown in Fig. 4. The timing simulation is shown from 20 ns to 40940 ns. After this time clock the random output is repeating again.

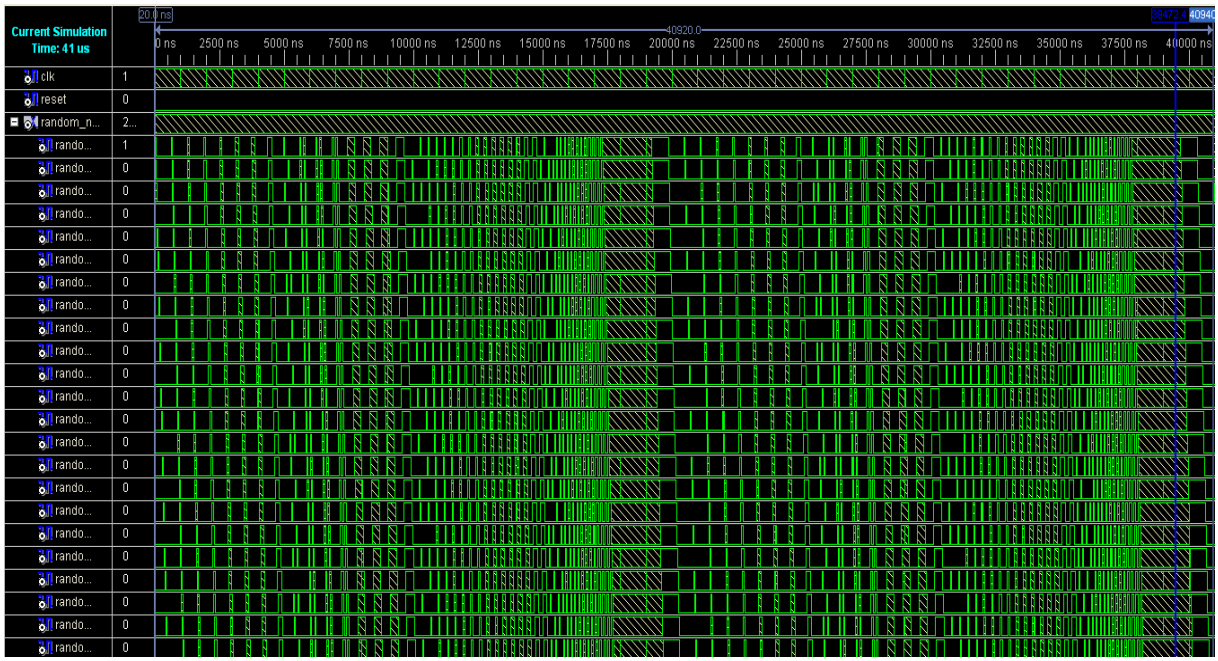

Figure 4. Timing Simulation waveform of LFSR-32 Bit (20 ns to 40940 ns): Total 2046 periods

3.2. Performance Analysis

The performance based on simulation and synthesis report for 8 bit, 16 bit and 32 bit LFSR using single tapping are given in Table 1.

Table 1. Simulation and Synthesis Result.

Performance	8 Bit	16 Bit	32 Bit
Time to complete the total states	20 ns to 1280 ns = 1260 ns	20 ns to 5120 ns = 5100 ns	20 ns to 40940 ns = 40920 ns
Total no. of Random States generating	63	255	2046
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
GCLK	01	01	01
(Gate + Net) Delay	2.690 ns	2.690 ns	2.690 ns
Total pin	10	18	34

IV. CONCLUSIONS

It is clearly found from the synthesis and simulation result that 8 bit 16 bit and 32 bit LFSR with single tapping can only generate 63, 255 and 2046 random output rather than 255, 65535 and 4,29,49,67,295. Also the performance is better in 32 bit LFSR as compared to other. In future enhancement by increasing the number of tapping the degree of randomness as well as the run length can be increased and again we can compared for maximum length sequence between different length LFSR by implementing on FPGA.

REFERENCES

[1]. M. Luby, Pseudorandomness and Cryptographic Applications, Princeton University Press, 1996.
[2]. Random Number Generator (2011). Wikipedia website [Online]. Available:http://en.wikipedia.org/wiki/Hardware_random_number_generator.
[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]. Pseudo-Random-Generator. Wikipedia website [Online] Available: http://en.wikipedia.org/wiki/Pseudorandom_number_generator
[8]. 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.
[9]. 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.
[10]. 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.
[11]. Linear Feedback Shift Register (2011), Wikipedia website. [Online]. Available: http://en.wikipedia.org/wiki/Linear_feedback_shift_register.
[12]. 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
[13]. 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.
[14]. K.H. Tsoi, K.H. Leung and P.H.W. Leong, Compact FPGA-based True and Pseudo Random Number Generators, 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03), 2003.
[15]. Jiang Hao, Li Zheying, "FPGA design flow based on a variety of EDA tools" in Micro-computer information, 2007(23)11-2:201-203.

[16]. Chan K., Samiha Mourad "Digital Design using Field Programmable Gate Arrays" Prentice Hall, Inc 1994
[17]. Field programmable gate array (2011). Wikipedia website. [Online]. Available:http://en.wikipedia.org/wiki/Field-programmable_gate_array.
[18]. Bhasker J, "A VHDL Primer", P T R Prentice Hall, Pages 1-2, 4-13, 28-30
[19]. Xilinx, Inc. Xilinx Libraries Guide, 2011.

Authors

Amit Kumar Panda was born on 08 Jul 1982. He has done M. Tech (ELDT) in 2009. His teaching Experience is 2.5 year in Guru Ghasidas Vishwavidyalaya (A Central University), Dept. of ECE, Bilaspur with the designation of Asst. Professor.

Praveena Rajput is working as an Assistant Professor in the Department of Electronics & Communication Engg., Guru Ghasidas Vishwavidyalaya, Bilaspur (C.G). She has done her B.E from I.E.T.E. She is also life member of IETE Institute. She has done his M.E from SSCET Bhilai with Hons.

Bhawna Shukla is working as an Asst. Professor & HOD in the Department of ECE, IT, GGV, Bilaspur (CG). She has done M. Tech from SSCET Bhilai. Her teaching experience is 13 year.