International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Survey on Implementation of Random Number Generator in FPGA

Pallavi Bhaskar[1], Prof. P. D. Gawande[2]

Department of Electronics and Telecommunication, Sipna College of Engineering and Technology
Sant Gadge Baba Amravati University, India

Abstract: ***A pseudo random number generator (PRNG), also known as a deterministic random bit generator(DRBG), is an algorithm for generating a sequence of random numbers. This paper presents an implementation of pseudo random number generator. The design has been specified in VHDL and is implemented on altera FPGA device. It is based on the Residue Number System (RNS),which gives us the way to design a very fast circuit. This paper presents design and implementation of a pseudo-random number generator based on Blum Blum Shub, XOR Shift, Fibonacci series and Galois LFSR methods. We will demonstrate that how the introduction of application specificity in the architecture can deliver huge performance in terms of area and speed. The design will specify in VHDL and will analyze on altera FPGA parameter. Which will give us higher throughput and also the parameter like area, propagation delay and power requirement.***

Keywords: Blum Blum Shub method, Fibonacci series method, Galois LFSR method, VHDL, XOR Shift

1. Introduction

A Random Number Generator (RNG) is a source of unpredictable numbers, which means that it is impossible to predict its outcome with an accuracy greater than the one given by the pure luck; a classic example is the coin toss, assuming a fair (i.e., unbiased) coin. In many practical applications such as cryptography, model simulation, sampling, games of chance, among others, there is a need of the generation of series of random numbers. This is achieved, for example, by means of tables, specific algorithms or electronic circuits. Unlike the natural sources of noise, these generators possess a finite period, so that they are called pseudorandom number generators (PRNGs). Most of the work on random number generator has been done on the algorithm and the development of software for them and not much work has been done on the hardware implementation of the random number generator. Yuan Li et. Al has done the hardware implementation of the random number generator based on Mersenne Twister (MT) algorithm and their random number generator produces 450 million samples per second.

2. Literature Review

GU Xiao-chen, ZHANG Min-xuan introduced a new kind of URNG using Leap-Ahead LFSR Architecture which could generate an m-bits random number per cycle using only one LFSR. They analyzed its architecture, present the expression of the period and point out how to choose the taps of the LFSR. Finally, a 18-bits URNG was implemented on Xilinx Vertex Ⅳ FPGA.. By comparison, the Leap-Ahead LFSR Architecture URNG consumes less than 40 slices which was only 10% of what the Multi-LFSRs architecture consumes and acquires very good Area Time performance and Throughput performance that were 2.18×10-9 slices×sec per bit and 17.87×109 bits per sec.[1]

Fabio Pareschi, Gianluca Setti, and Riccardo Rovatti given that the architecture used for common pipeline ADCs can be reused for designing a chaotic circuit, which is very appealing for the generation of random numbers. Following this approach they have de- signed two prototypes of a true-random number generator in 0.35- m and 0.18- m CMOS technology, capable of generating random bits with a throughput, respectively, of 40 Mbit/s and 100 Mbit/s[2].

Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and David H. K. Hoe Evaluated the performance of CA-based PRNGs suitable for implementation on FPGAs. Their results for the Xilinx Spartan 3E FPGA given a good idea of the relative resources required for each configuration. The DIEHARD suite of statistical tests was used to evaluate the quality of the random numbers produced from each configuration. It was found that the 37 bit LFSR + 16 bit CA and the 52 bit LFSR + 8 bit CA produced the best results. The LFSR + CA uses less overhead than the SPCA and produces higher quality random numbers, but the SPCA gives much greater throughput with similar randomness but greater overhead.[3]

Pawel Dabal Ryszard Pelka In this the author described implementation of three versions of chaotic pseudo-random bit generators in five selected FPGA devices offered by Xilinx. A comparative study of required number of FPGA resources and maximum operating frequencies had been presented. They had shown that the proposed PRBG architectures can be relatively easy implemented in FPGA devices and used in embedded SoC systems. It is also possible to implement dedicated interfaces for different bus standards.[4]

Carlos Arturo Gayoso, C. González, L. Arnone, M. Rabini, Jorge Castiñeira Moreira presented a new PRNG based on the RNS. The proposed circuit had a dynamic, due to the use of RNS, different than known generators. It also had a superior performance in terms of speed and resource usage with respect to its realization in 2's complement. Rigorous tests like those of the pack Diehard for testing randomness of numeric series have been applied to the generated sequences, to verify that the proposed generator satisfactorily passes the different tests applied.[5]

3. Methods Used

We are going using Modelsim and Quartus 2 software. Here are the following methods which will produce the pseudorandom number generator mentioned below.

1. Design of a Random Number Generator using XOR Shift method XOR shift is a category of pseudorandom number generators. It repeatedly uses the transform of exclusive or on a number with a bit shifted version of it.

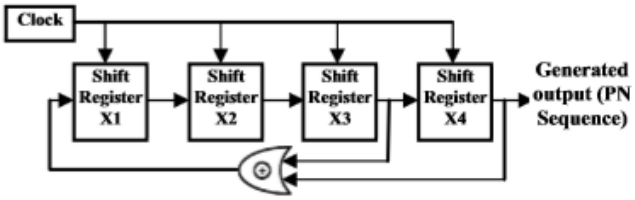

Figure 1. Basic block diagram of LFSR

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 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 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. [8][13]

2. Design of a Random Number Generator using Fibonacci series method Fibonacci series method works on the principal of Sn = Sn-1 + Sn -2 . It performs binary addition like addition, subtraction, multiplication or Ex - Or operation. 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.

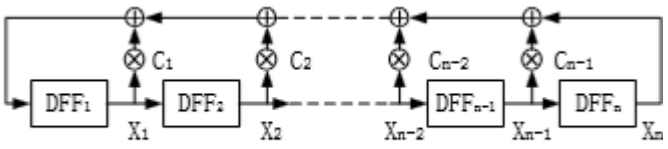

Fig Fibonacci Type

There can be more than one maximum-length tap sequence for a given LFSR length. Also, 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]. As well as Fibonacci, this LFSR configuration is also known as standard, many-to-one or external XOR gates. LFSR has an alternative configuration.[1]

3. Design of a Random Number Generator using Galois LFSR method.

In Galois method, when the system is clocked, bits that are taps are shifted one position to the right unchanged. The taps on the other hand are XOR with the output bit before they are stored in next position. 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. 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).[1][9]

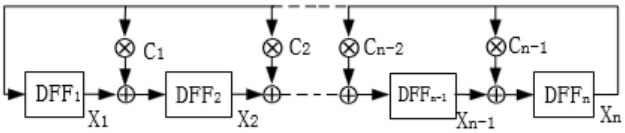

Fig Galoise Type

4. Design of a Random Number Generator using Blum Blum Shub method Blum Blum Shub (B.B.S.) is a pseudorandom number generator works on the principal of Xn+1 = Xn mod

Where n=p x q is the product of two large primes p and q. At each step of the algorithm, some output is derived from xn+1; the output is commonly the bit parity of Xn+1 or one or more of the least significant bits of Xn+1. The two primes, p and q, should both be congruent to 3 (mod 4) (this guarantees that each quadratic residue has one square root which is also a quadratic residue) and GCD (φ(p-1), φ(q-1)) should be small (this makes the cycle length large).[11][12]

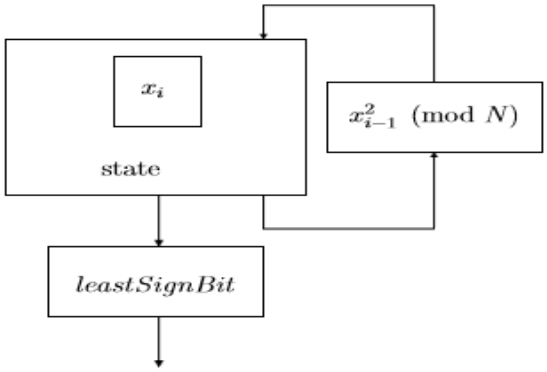

Fig:- General structure of BBS

4. Conclusion

In this paper, we are going to generate random numbers to get higher throughput means in how much time maximum random numbers will be generated. In our proposed work, we are going to implement the basic to other methods which will give us higher throughput and also the parameter like area, propagation delay and power requirement will be compared with previous authors implementation.

References

[1] GU Xiao-chen, ZHANG Min-xuan School of Computer National University of Defense Technology Changsha, China *"Uniform Random Number Generator using Leap-Ahead LFSR Architecture"* 2009 International Conference on Computer and Communications Security

[2] Fabio Pareschi, Member, IEEE, Gianluca Setti, Fellow, IEEE, and Riccardo Rovatti, Senior Member, IEEE *"Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems"* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 57, NO. 12, DECEMBER 2010

[3] Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and David H. K. Hoe Department of Electrical Engineering The University of Texas at Tyler *"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

[4] Pawel Dabal Faculty of Electronics Military UniverRyszard Pelka Faculty of Electronics Military University of Technology Warsaw, Poland sity of Technology Warsaw, Poland *"FPGA Implementation of Chaotic Pseudo-Random Bit Generators"* MIXDES 2012, 19th International Conference "Mixed Design of Integrated Circuits and Systems" , May 24-26, 2012, Warsaw, Poland

[5] Carlos Arturo Gayoso, C. González, L. Arnone, M. Rabini, Jorge Castiñeira Moreira *"Pseudorandom Number Generator Based on the Residue Number System and its FPGA Implementation"* 2013 Argentine School of Micro-Nanoelectronics, Technology and Applications

[6] Ravi Saini, Sanjay Singh, Anil K Saini, AS Mandal, Chandra Shekhar CSIR- Central Electronics Engineering Research Institute (CSIR-CEERI) Pilani-333031, Rajasthan, India *"Design of a Fast and Efficient Hardware Implementation of a Random Number Generator in FPGA"* 2013 International Conference on Advanced Electronic Systems (ICAES)

[7] David B. Thomas, Member, IEEE, and Wayne Luk, Fellow, IEEE *"The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures"* IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 21, NO. 4, APRIL 2013

[8] Amit Kumar Panda*, Praveena Rajput, Bhawna Shukla Deptt. of ECE, IT Guru Ghasidas Vishwavidyalaya Bilaspur, India FPGA *"Implementation of 8, 16 and 32 Bit LFSR with Maximum Length Feedback Polynomial using VHDL"* 2012 International Conference on Communication Systems and Network Technologies

[9] Beker, Henry; Piper, Fred (1982). *Cipher Systems: The Protection of Communications.* Wiley-Interscience. p. 212

[10] Paplinski, A.P. and Bhattacharjee, N, *"Hardware implementation of the Lehmer random number generator,"* IEE Proceedings of Computers and Digital Techniques, vol. 143, no. 1, pp. 93 –95, 1996.

[11] Lenore Blum, Manuel Blum, and Michael Shub. Comparison of two pseudo-random number generators. In R. L. Rivest, A. Sherman, and D. Chaum, editors, Proc. CRYPTO 82, pages 61–78, New York, 1983. Plenum Press.

[12] Cryptographic Secure Pseudo-Random Bits Generation : The Blum-Blum-Shub Generator Pascal Junod August 1999

[13] 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.