Pseudo Random Number Generator

Swapnil Gutte, Chetan Shinde, Sharvin Tanwade, prof. Swati Shilaskar, prof. Abhay Chopde

**Abstract**—***Random number generation plays a crucial role in various fields, including cryptography, simulations, and statistical analysis. Pseudo-Random Number Generators (PRNGs) are widely utilized to generate sequences of numbers that exhibit statistical properties similar to true random numbers. This research paper proposes a novel approach for constructing a pseudo-random number generator using a Shift Register. The study presents a detailed analysis of the shift register-based architecture and its ability to produce pseudo-random numbers with desirable statistical properties. The proposed PRNG utilizes a linear feedback shift register (LFSR) to generate a sequence of bits that are subsequently converted into a pseudo-random number stream. The simulation is done on the software named Xilinx using Verilog.***

***Keywords—Pseudo-random number generator (PRNG), Linear feedback Shift resistor (LFSR), Verilog***

1. INTRODUCTION

A group of numbers called "random numbers" lacks a predetermined sequence. The fact that random numbers are independent, or that there is no association between consecutive numbers, is one of their most crucial qualities. Lotteries, intricate mathematical modelling, gambling, encryption, and security all depend on random numbers. The difficulty in producing random numbers is that computers, which offer algorithm-based mathematical solutions, are fundamentally predictable. Pseudo-Random Number Generators (PRNGs) and True Random Number Generators are the two most popular ways for computers to produce random numbers (TRNGs). An algorithm and a seed number are used by a computer to create PRNGs, which appear random but are actually predictable. Although the outputs of PRNGs may produce patterns that are very complicated, the numbers they generate are not truly random since they follow a preset process. Contrary to PRNGs, TRNGs use hardware to assess randomness from physical occurrences and incorporate it into a computer. Computers generate really random numbers by using unexpected phenomena, such as air noise or radioactive decay, rather than human-defined patterns. Here, the surrounding processes of the computer's randomness are utilized to generate a true random number generator.

If you wish to play music while randomly shuffled songs are played, you will need numbers to enter into the program so that the samples are more or less evenly dispersed. In these situations, a pseudo-random number is perfectly acceptable. Since the level of unpredictability has no quantifiable advantages in this situation, the usage of pseudo-random numbers is enough. It's important to recognize the value of pure unpredictability. A program that anticipates crucial cryptic information may be written if the attacker knows the technique used in cryptography, for example, which uses integers that an attacker cannot guess. Such reverse engineering is unlikely when using true random numbers. In order to create a series of numbers that are difficult to anticipate, random number generators (RNGs) are utilized. Random number generators have recently been discovered to be helpful in fields including encryption, sampling, simulations (Monte Carlo simulation), one-time password creation, and hardware security. The random number generator must meet the following criteria in order to function properly: the sequences produced must be statistically significant, the random number generator's future state cannot be readily estimated, and the series of random numbers must not be predictable. In order to boost the random number generator's unpredictable nature, an RNG based on a linear feedback shift register (LFSR) is presented in this research. The produced random numbers should ideally meet any statistical test for randomness and be uncorrelated. There are two types of generators: "really random" and "pseudo-random." The value of the next number is uncertain whereas the former displays actual randomness. The latter just seems illogical. The pattern is repeating and predictable since the sequence is based on particular mathematical procedures. On the other hand, if the cycle time is too high, the sequence seems random and non-repetitive. True Random number generators can be implemented in hardware, but doing so is slow and costly. In this research, we introduce an 8-bit shift register with linear feedback random number generator. The entire design was written down in VHDL and synthesized for a particular XILINX Software.

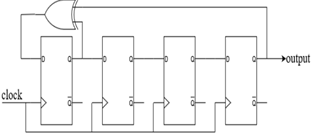


Fig. 1. LFSR Structure

1. LITERATURE SURVEY

The Paper [1] presents a random number generator composed entirely of digital circuits, which utilises electronic noise. The design of a new type of true random number generator that is based solely on digital components consumes little power, provides high throughput, and passes the DIEHARD suite of tests for randomness. Also, it discusses the methodology for the design, implementation and testing of a true random number generator based on digital artifacts. Lastly, this paper concludes by providing results and outlining the necessary steps to create practical versions of these promising designs. The [2] paper says a new improvement is proposed to ring oscillator-based random number generators. The implemented circuit utilizes 1531 LUTs and 14 FFs on a Xilinx Artix-7 FPGA. The proposed method does not require manual placement of the ring oscillators to be as far as possible from each other. RNG can be implemented on any kind of FPGA chips easily using auto-placement techniques supplied by the design software. The generated random bit streams passed full NIST 800-22 tests without any need for post-processing. An attack circuit is also implemented to see if it is possible to predict the random bit outputs of the proposed RNG. This [3] paper evaluates the hardware-based Intel Random Number Generator (RNG) for cryptographic applications. Along with that it briefly explains the working of the Block diagram of the Intel Random Number Generator. Also, the various parameters of Random Number Generators based on architecture were discussed. In producing the Random Number Generator, Intel applied conservative design, implementation, and testing approaches concepts were discussed. The [4] paper highlights the design of a True Random Number Generator based on the XOR of the outputs of several oscillator rings. They propose an enhanced TRNG with better randomness characteristics that do not require postprocessing and pass the statistical tests. The experiment shows that the frequencies of the equal-length oscillator rings in the TRNG are not identical. A True Random Number Generator (TRNG) on a Xilinx FPGA provides high throughput and security against physical attacks. The proposed architecture, implemented in a Virtex-5 XC5VLX50T, uses 4% of the available resources and generates random bits at a 20 Mbps rate. The randomness quality of this TRNG has been validated using AIS-31 and NIST statistical tests [5]. LFSR is a primarily based random number generator with novel on-the-fly seed change in the usage of the System display of FPGA. The proposed scheme is carried out on the Xilinx Virtex-5 ML505 board. Experimental outcomes exhibit that on-chip assets take much less location and electricity consumption [6].

This [7] paper aims to design a digital true random number generator (RNG). A test board was designed to collect random numbers from the fabricated IC. The software was developed in C language to operate the fabricated RNG chip, perform functional testing, generate random numbers and upload generated random numbers to a PC. Random number generation (RNG) is the most important factor in a security system. We propose an RNG which can generate unpredictable random numbers by utilizing bio-related signals embedded in wearable devices. The proposed structure has been verified for excellent security through the NIST test suite [8].

Random numbers are used in numerous cryptographic applications such as key generation, encryption, masking protocols, and online gambling. Many of these security procedures may be implemented in hardware using field programmable gate arrays (FPGAs). The proposed TRNG is based on the beat frequency detection approach for Xilinx-FPGA-based applications [9]. This paper [10] presents a highly efficient and tunable TRNG based on the principle of Beat Frequency Detection (BFD), specifically for Xilinx FPGA-based applications. The main advantages of the proposed TRNG are its on-the-fly tunability through Dynamic Partial Reconfiguration (DPR) to improve randomness qualities. The True

Random Number Generation process became more detailed and was elaborated in international standards (for example the NIST). The Whole logic is implemented in FPGA logic [11]. 10-Bit random number generation can be achieved using an LFSR (Linear Feedback Shift Register). An LFSR is simply a shift register with some of its bits (known as taps) XOR'd with themselves to create a feedback term. When implementing an LFSR, its width and repeatability must be considered. An N-bit LFSR will be able to generate (2^N) - 1 random bit before it starts repeating. We used 10 LFSR for the generation of 10-bit Random numbers. In an LFSR the MSB will always be the feedback point. The main thing of an LFSR is to know which bits are the taps (to be selected for XOR). The taps are different for different size registers. An LFSR is good for generating a stream of random bits. It does not generate a random number by itself; only the feedback bit is random. Hence random numbers are generated successfully.

1. METHODOLGY

The shift register is one of an LFSR's two primary components (the other being the feedback function). The distinguishing feature of a shift register is that it moves its contents into nearby places inside the register or, in the case of the position on the end, out of the register. Unless fresh material is added to the register, the slot on the other end is left vacant. A shift register has two functions: it can transfer data between parallel and serial streams and delay a serial bit stream. The conversion function can either shift the contents into the register bit by bit or load the shift register slots in parallel, shift them out in serial (serial).

A shift register whose input bit is a linear function of its prior state is known as a linear feedback shift register (LFSR). It is a shift register whose input bit is controlled by the exclusive-or (xor) of certain bits of the overall shift register value since the only linear functions of single bits are xor and inverse-xor. Because the functioning of the register is deterministic, the sequence of values produced by the register is entirely governed by its present (or prior) state. In the proposed system, we have used an external feedback shift register. Here we have used an external feedback shift register instead of an internal one because it takes less time to process. The beginning value of the LFSR is referred to as the seed. Similarly, the register must ultimately enter a repeating cycle since there are only a finite number of potential states. Pseudorandom outputs are produced through an experiment using various tap locations applied to an 8-bit set of registers. The output values vary as the clock pulse goes from low to high when the load is in the high state.

This is the proposed model of generating 10-BIT random numbers using “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.

1. NOVELTY

There were many parameters where our system has the upper hand on other systems like-

1] High Randomness

2] It gives output in Decimal format

3] Easy to implement

4] Multi-purpose application

5] Time taken is less

8-bit Random Number Generator implemented using Verilog. In our system, high randomness is created using random bits of LFSR. and our proposed system gives the output in binary form and we worked on decimal form. This system is easy to implement and has a multi-purpose application. The processing time of the system is less than compared to existing systems.

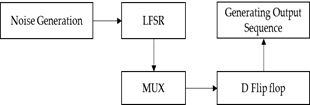


Fig. 4. Block diagram of proposed system

In the above Fig.4, Noise is generated using an LFSR circuit where randomness is generated using a random seed. LFSR and MUX are used in combination where multiple inputs from LFSR are given to MUX which creates a single output. This single output is given to the D-Flip flop. Finally, we generate an 8-bit random number generator output sequence.

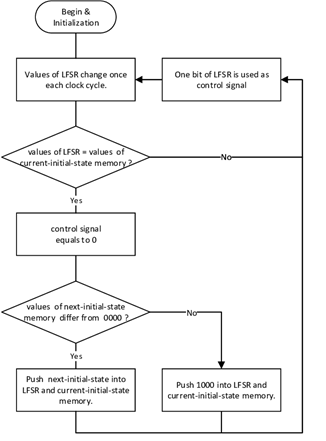


Fig. 5 Working flow of project

1. **RESULTS**

Pseudorandom outputs are produced through an experiment using various tap locations applied to an 10-bit set of registers. The output values vary as the clock pulse goes from low to high when the load is in the high state. Additionally, we discovered that the output values repeat themselves after a specific clock pulse. This is mostly beneficial for security reasons. This is quite safe. Circuit testing, cryptography, and data encryption are the key uses. These are quite helpful for security purposes since they produce good unpredictability. We used the first seed in the picture above to get various random integers. It begins to produce various patterns to obtain randomness just from the original seed, which is highly helpful for security purposes. Counters and dividers may be created with the LFSR.

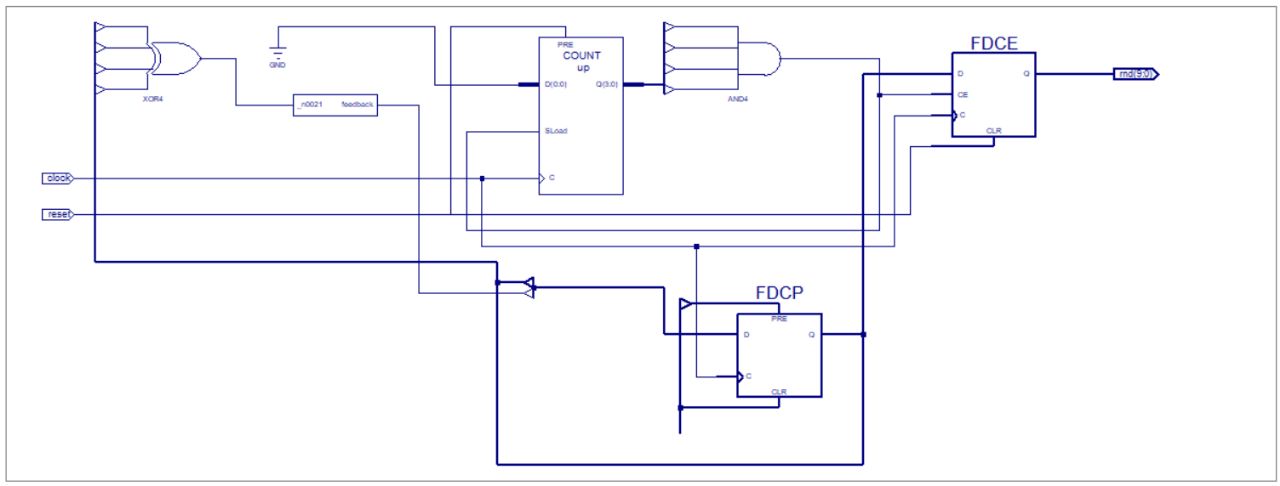


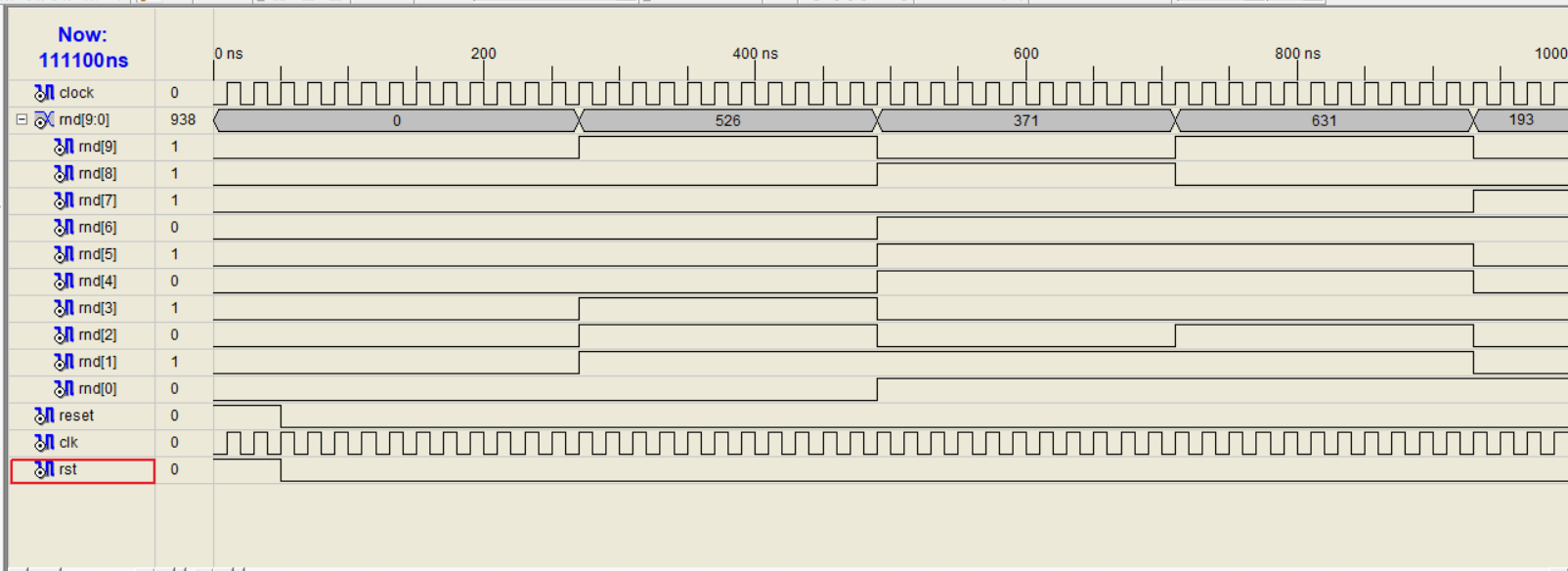
Fig. 5. Output Screenshots for RNG

Fig. 6. Waveform Output

1. **CONCLUSION**

Based on our research, we can conclude that LFSR serves multiple important purposes in real-life applications. LFSR makes it easier for cybersecurity, CRC, cryptography and pseudorandom generators. Hardware and software can be used to work on random number generators. For future work, we would like to work on software skills and will use Verilog code. We would deeply work on CRC and use different approaches like one-to-many implementation and more than two taps alternative using XOR to get more random values.

**REFERENCES**

[1] Michael Epstein, Laszlo Hars, Raymond Krasinski, Martin Rosner, and Hao Zheng, “Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts”, C.D. Walter et al. (Eds.): CHES 2003, LNCS 2779, pp. 152–165, 2003. © Springer-Verlag Berlin Heidelberg 2003.

[2] Burak Acar and Salih Ergun, “A Digital Random Number Generator Based on Irregular Sampling of Regular Waveform”, 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS).

[3] Benjamin Jun and Paul Kocher, “The Intel Random Number Generator”, Shannon, C.E., “A Mathematical Theory of Communication,” The Bell System Technical Journal, vol. 27, p. 379-423, July 1999.

[4] Knut Wold and Chik How Tan, “Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings”, Hindawi Publishing Corporation International Journal of Reconfigurable Computing Volume 2009, Article ID 501672, 8 pages.

[5] F. Lozach, M. Ben-Romdhane, T. Graba and J. -L. Danger, "FPGA Design of an Open-Loop True Random Number Generator," *2013 Euromicro Conference on Digital System Design*, 2013, pp. 615-622.

[6] P. Zode, P. Zode and R. Deshmukh, "FPGA Based Novel True Random Number Generator using LFSR with Dynamic Seed," *2019 IEEE 16th India Council International Conference (INDICON)*, 2019, pp. 1-3

[7] Wang, J., Cao, L., & Yang, Y. (2020). "Design of a true random number generator". *2020 International Conference on Artificial Intelligence and Communication Technology (AICT 2020)* (pp. 202-208). China: CSP.

[8] H. Yu and Y. Kim, "True Random Number Generator Using Bio-related Signals in Wearable Devices," *2018 International SoC Design Conference (ISOCC)*, 2018, pp. 231-232

[9] Narole, Dr. Narendra G. and Kshanada B. Choudhary. “Implementation of True Random Number Generator Using FPGA Based System.” (2018).

[10] A. P. Johnson, R. S. Chakraborty and D. Mukhopadyay, "An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 64, no. 4, pp. 452-456, April 2017, doi: 10.1109/TCSII.2016.2566262.

[11] Knut Wold and Chik How Tan, “Analysis and Enhancement of RNG in FPGA Based on Oscillator Rings”, Hindawi Publishing Corporation International Journal of Reconfigurable Computing Volume 2009.