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

A Review on Implementation of Random Number Generation based on FPGA

Vishakha V. Bonde[1], A. D. Kale[2]

[1]Student, Electronics and Telecommunication Engg, Sant Gadge Baba University Amravati, India
[2]Assistant Professor, Electronics and Telecommunication Engg, Sant Gadge Baba University Amravati, India

Abstract: ***Random number generator is required extensively by many applications like cryptography, simulation, numerical analysis, text-to-speech etc. Most C libraries have a pair of library routines for initializing, and then generating random numbers. For parametric speech synthesis application, a random number generator is required to produce noise samples. Therefore, a need has been felt for the design of a dedicated hardware for random number generator that generates one random number per cycle so that text-to speech conversion is done in real time. A random number generator (RNG) is a device designed to generate a sequence of numbers or symbols that don't have any pattern. Hardware-based systems for random number generation are widely used, but often fall short of this goal, albeit may meet some of the statistical tests for randomness for ensuring that do not have any "de-codable" patterns.***

Keywords: Random Number Generator, Cryptography, C, synthesis, text-to-speech, FPGA

1. Introduction

A pseudorandom number generator (PRNG), is an algorithm for generating sequence of numbers that approximates the properties of random numbers. The Sequence is not truly random. Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom numbers are important in practice for simulations (e.g., of physical systems with the Monte Carlo method), and are important in the practice of cryptography. A PRNG can be started from an arbitrary starting state using a seeds. It will always produce the same sequence thereafter when initialized with that state. The maximum length of the sequence before it begins to repeat is determined by the size of the state. However, since the length of the maximum period doubles with each bit of 'state' added, it is easy to build PRNGs with periods long enough for many practical applications. Most pseudorandom generator algorithms produce sequences which are uniformly distributed by any of several tests.

The security of most cryptographic algorithms and protocols using PRNGs is based on the assumption that it is infeasible to demarcate use of a suitable PRNG from the usage of a truly random sequence. The simplest examples of this dependency are stream ciphers, which work by exclusive or-ing the plaintext of a message with the output of a PRNG, producing cipher text. The design of cryptographically secure PRNGs is extremely difficult; because they must meet additional criteria .The size of its period is an important factor in the cryptographic suitability of a PRNG, but not the

Only one Algorithms for pseudorandom number generators are of many types such as Blum Blum Shub, ISAAC (Cipher), Inversive congruential generator, Lagged Fibonacci generator, Linear feedback shift register, Xorshift, Linear congruential generator etc.

LFSR is the traditional method for generating random numbers which uses shift registers. VHSIC HDL prefer because of its flexibility and writing commands. FPGA can implement any logical expression i.e. it is predefined reconfigurable IC. It can be reconfigured any number of time. Therefore FPGA is used for rapid prototype development as compared to ASIC.

The 8 and 16 bit length sequence using verilog HDL implemented on FPGA kit. Also the comparison between 8and 16 bit on the basis of synthesis and simulation result. FPGA can implement any logical expression i.e. it is predefined reconfigurable IC. It can be reconfigured any number of time. Therefore FPGA kit is used for rapid prototype development as compared to ASIC; hence FPGA is used to implement design.

There are two principal methods used to generate random numbers. One measures some physical phenomenon that is expected to be random and then compensates for possible biases in the measurement process. The other uses mathematical algorithms that produce long sequences of apparently random numbers, which are in fact completely determined by an initial value, known as a seed. The former one is known as True Random Number Generator (TRNG).

2. Literature Review

From the rigorous review of related work and published literature, it is observed that many researchers have designed random number generation by applying different techniques. Researchers have undertaken different systems, processes or phenomena with regard to design and analyze RNG content and attempted to find the unknown parameters. A pseudorandom number generator (PRNG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. These sequences are not truly random. Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom numbers are important in practice for simulations (e.g., of physical systems with the Monte Carlo method), and are important in the practice of cryptography.

Ray C. C. Cheung, Dong-U Lee, John D. Villasenor [1], presented an automated methodology for producing hardware-based random number generator (RNG) designs for

arbitrary distributions using the inverse cumulative distribution function (ICDF). The ICDF is evaluated via piecewise polynomial approximation with a hierarchical segmentation scheme that involves uniform segments and segments with size varying by powers of two which can adapt to local function nonlinearities. Analytical error analysis is used to guarantee accuracy to one unit in the last place (ulp). Compact and efficient RNGs that can reach arbitrary multiples of the standard deviation can be generated. For instance, a Gaussian RNG based on our approach for a Xilinx Virtex-4 XC4VLX100-12 field-programmable gate array produces 16-bit random samples up to 8.2delta. It occupies 487 slices, 2 block-RAMs, and 2 DSP-blocks. The design is capable of running at 371 MHz and generates one sample every clock cycle. The designs are capable of generating random numbers from arbitrary distributions provided that the ICDFs is known.

GU Xiao-chen, ZHANG Min-xuan [2] presented "Uniform Random Number Generator using Leap-Ahead LFSR Architecture". Introducing a new kind of URNG using Leap-Ahead LFSR Architecture which could generate an m-bits random number per cycle using only one LFSR. A normal LFSR could only generate one random bit per cycle. As multi-bits is required to form a random number in most applications, Multi-LFSRs architecture is used to implement a URNG. This means 32 different LFSRs are needed in a 32-bit output URNG. But Leap-Ahead architecture could avoid this and generate one multi-bits random number per cycle using only one LFSR. The Leap-Ahead architecture consumes less than 10% of slices which the Multi-LFSR architecture consumes. One of the reasons for this is that the Leap-Ahead architecture has only 1LFSR in the URNG hardware, while the Multi-LFSR architecture has 18. The other reason is that every register in the URNG has to be initialed separately when the circuit is restarted, and the logic for this is complicated. As the Multi-LFSR architecture has 18×18registers, while the Leap-Ahead architecture has only 23registers, it needs more slices for the initializing function.

By implementing the Leap-Ahead LFSR architecture and Multi-LFSR architecture of both Galois type and Fibonacci type on Xilinx Vertex 4 FPGA, we acquire the conclusion that, with only very little lost in speed, Leap-Ahead LFSR architecture consumes only 10% slices of what the Multi-LFSR architecture does to generate the random numbers that have the same period. By comparison with other URNGs, Leap-Ahead LFSR architecture has very good Area Time performance and Throughput performance that are 2.18×10-9 slices×sec per bit and 17.87×109 bits per sec.

Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and David H. K. Hoe [3] introduced new architecture using Cellular Automata. Cellular Automata (CA) have been found to make good pseudo-random number generators (PRNGs), and these CA-based PRNGs are well suited for implementation on Field Programmable Gate Arrays (FPGAs). To improve the quality of the random numbers that are generated, the basic CA structure is enhanced in two ways. First, the addition of a super-rule to each CA cell is considered. The overviews of the design of linear feedback shift register (LFSRs) and cellular automata (CA), followed by a review of related works that have utilized LFSR and CA for generating random numbers. Therefore, evaluated the performance of CA-based PRNGs suitable for implementation on FPGAs. Synthesis results for the Xilinx Spartan 3E FPGA give a good idea of the relative resources required for each configuration.

Pawel Dabal, Ryszard Pelka [4] presented "FPGA Implementation of Chaotic Pseudo-Random Bit Generators"
Modern communication systems (including mobile systems) require the use of advanced methods of information protection against unauthorized access. Therefore, one of the essential problems of modern cryptography is the generation of keys having relevant statistical properties. In recent years, the cryptographers pay an increasing attention to digital systems based on chaos theory. The use of chaotic signals to carry information .An idea of using a nonlinear chaotic dynamic system for design of cryptographic secure pseudo-random number or bit generator (PRNG or PRBG) seems to be interesting from practical reasons.

Carlos Arturo Gayoso, C. González, L. Arnone, M. Rabini, Jorge Castiñeira Moreira, [5] presented "Pseudorandom Number Generator Based on the Residue Number System and its FPGA Implementation"
Residue Number System (RNS), which allows us to design a very fast circuit that has a very different way of operating with respect to other generators. A set of classic tests, the Diehard test, the statistic complexity test and the Hurst exponent test are used to provide a measure of the quality of the randomness of the proposed pseudorandom number generator.

David B. Thomas, Wayne Luk, [6] presented "The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures".

A type of FPGA RNG called a LUT-SR RNG, which takes advantage of bitwise XOR operations and the ability to turn lookup tables (LUTs) into shift registers of varying lengths. This provides a good resource–quality balance compared to previous FPGA-optimized generators, between the previous high-resource high-period LUT-FIFO RNGs and low-resource low-quality LUTOPT RNGs, with quality comparable to the best software generators. The LUT-SR generators can also be expressed using a simple C++ algorithm contained within this paper, allowing 60 fully-specified LUT-SR RNGs with different characteristics to be embedded in this paper, backed up by an online set of very high speed integrated circuit hardware description language (VHDL) generators and test benches**.**

Ravi Saini, Sanjay Singh, Anil K Saini, A S Mandal, Chandra Shekhar [7] presented "Design of a Fast and Efficient Hardware Implementation of a Random Number Generator in FPGA" presents a fast and efficient hardware implementation of a pseudo-random number generator based on Lehmer linear congruential method. Demonstrated in this paper that how the introduction of application specificity in the architecture can deliver huge performance in terms of area and speed. The design has been specified in VHDL and is implemented on Xilinx FPGA device XC5VFX130T-3ff1738 and takes up only 23 slice LUTS.

In 2014, Purushottam Y. Chawle and R.V. Kshirsagar [8] , presented a simple algorithm to generate pseudo random number using Linear Feedback Shift register(LFSR).The generated pseudo sequence is mainly used for communication process such as cryptographic, encoder and decoder application in coded format.
In LFSR operation, the linear operation of single bit is exclusive-or (X-OR). The 8 and 16 bit LFSR is designed using verilog HDL language to study their performance and randomness. LFSR is a shift register whose output random state is depend on feedback polynomial.

3. Proposed Work

Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena and for selecting random samples from larger data sets. They have also been used aesthetically, for example in literature and music, and are of course ever popular for games and gambling. The earliest methods for generating random numbers are dice, coin flipping, roulette wheels are still used today, mainly in games and gambling as they tend to be too slow for most applications in statistics and cryptography.

Blum Blum Shub, ISAAC (cipher), Inversive congruential generator, Lagged Fibonacci generator, Linear feedback shift register, Multiply-with-carry, Xorshift, Linear congruential generator, Mersenne twister.In our case we will design Blum Blum Shub, LFSR, XORshift.

Linear Feedback Shift Register (LFSR)

A linear feedback shift register (LFSR) is a shift register whose input bit isa linear function of its previous state. The only linear function of single bits is XOR, thus it is a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. 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 generates 1 random number per cycle with a clock frequency of 502 MHz (502 million samples per second).

The random numbers generated by our design are extensively verified against the C-code generated outputs for functional correctness.

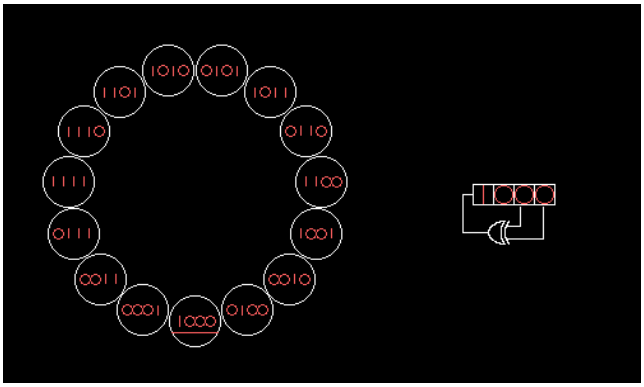

Figure 1: A 4 bit Fibonacci LFSR with its state diagram

Fibonacci LFSRs

A 16-bit Fibonacci LFSR. The feedback tap numbers in white correspond to a primitive polynomial in the table so the register cycles through the maximum number of 65535 states excluding the all-zeroes state. The state shown, 0xACE1 (hexadecimal) will be followed by 0x5670.The bit positions that affect the next state are called the taps.

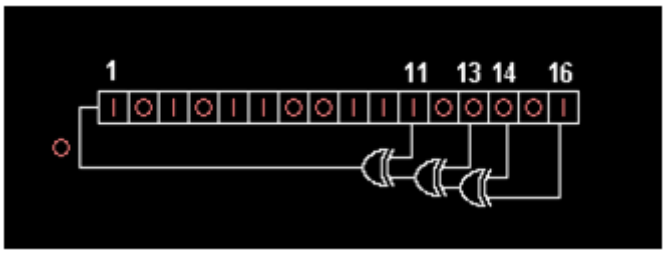

Figure 2: A 16 bit Fibonacci LFSR

In the diagram the taps are [16,14,13,11]. 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 $2^n - 1$ 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. As an alternative to the XOR based feedback in an LFSR, one can also use XNOR. 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. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state.

Galois LFSRs

Named after the French mathematician Évariste Galois, an LFSR in Galois configuration. 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 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.

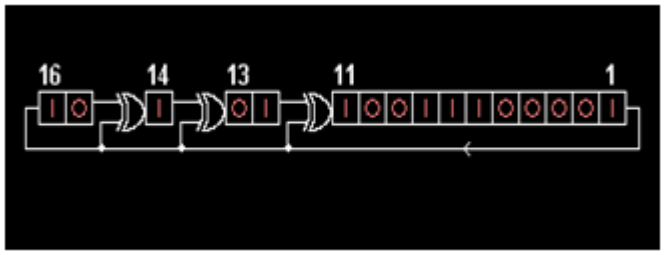

Figure 3: A 4 bit Galois LFSR

XORShift

Xorshift 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. This makes them extremely fast on modern computer architectures. The xor shift primitive is invertible if the number of combinations is odd, but not otherwise (most textbook xorshift implementations have 3 combinations, since this is the minimum number for a maximum period generator, given correct parameters).

Blum Blum Shub

Blum Blum Shub (B.B.S.) is a pseudorandom number generator proposed in 1986by Lenore Blum, Manuel Blum and Michael Shub (Blum et al., 1986).Blum Blum Shub takes the form:

$$\mathbf{Xn+1 = Xn^2 mod\ n}$$

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) .

There are many applications where random number generation is very essential. Some of the applications are listed below.

In Cryptography

A ubiquitous use of unpredictable random numbers is in cryptography which underlies most of the schemes which attempt to provide security in modern communications (e.g., confidentiality, authentication, electronic commerce, etc.).

For Encryption Algorithm

If a user wants to use an encryption algorithm, it is best that they select a random number as the key. These numbers must have high entropy for any attacker, thus increasing attack difficulty.

For Parametric Speech Synthesis

For parametric speech synthesis application, a random number generator is required to produce noise samples. Therefore, a need has been felt for the design of a dedicated hardware for random number generator that generates one random number per cycle.

In Simulation and Modeling

Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena and for selecting random samples from larger data sets.

Literature, Music and Art

Random numbers are useful for a variety of purposes. They have also been used aesthetically, for example in literature and music, and are of course ever popular for games and gambling. Some aesthetic theories claim to be based on randomness in one way or another.

Uses in circuit testing

PRNG are used in circuit testing, for test pattern generation (for exhaustive testing, pseudo random testing or pseudo exhaustive testing) and signature analysis.

For Signature analysis

In Built In Self Testing (BIST) techniques, storing all the circuit outputs on chip is not possible, but the circuit output can be compressed to form a signature which later will be compared to the golden signature (of the good circuit) to detect the faults.

4. Conclusion

In this review paper, a literature review on different designing methodologies used for generating random numbers has been presented. Several different ways have been already examined to increase randomness of random number generator. For a single bit random number generator, LFSR is most effective method. When multiple bits are required, LFSR can be extended by utilizing extra time and extra circuitry. Cryptographic algorithms and communication protocol are based on random number generation. By implementing, multi LFSR Architecture of both Fibonacci and Galois type on FPGA, we acquire conclusion that with only very little loss in speed, multi LFSR generate random numbers.

References

[1] Ray C. C. Cheung, John D. Villasenor, Wayne Luk, "Hardware Generation of Arbitrary Random Number

Distributions From Uniform Distribution Via the Inversion Method" vol.15, no. 8, August 2007.
[2] GU Xiao-chen, ZHANG Min-xuan "Uniform Random Number Generator using Leap- Ahead LFSR Architecture"2009 International Conference on Computer and Communications Security.
[3] Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, and David H. K. Hoe 44th IEEE Southeastern Symposium on System Theory University of North Florida, Jacksonville, FL March 11-13, 2012.
[4] Pawel Dabal, Ryszard Pelka "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] David B. Thomas, Wayne Luk, "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
[7] Ravi Saini, Sanjay Singh, Anil K Saini, A S Mandal, Chandra Shekhar "Design of a Fast and Efficient Hardware Implementation of a Random Number Generator in FPGA" CSIR- Central Electronics Engineering Research Institute (CSIR-CEERI) Pilani-333031, Rajasthan, India 2013 International Conference on Advanced Electronic Systems (ICAES).
[8] Purushottam Y. Chawle and R.V. Kshirsagar, "Design of 8 and 16 bit LFSR with maximum length feedback polynomial using verilog HDL".13th IRF international conference 20thjuly 2014, Pune, India

Author Profile

Vishakha V. Bonde received her BE degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University in 2012. Currently pursuing ME degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University, India.

A. D. Kale received his BE degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University in 2009 and M. Tech. in Electronic System and Communication from Sant Gadge Baba Amravati University in 2012. Currently working as Assistant Professor in P. R. Patil COE, Amravati, India