Efficient and Low Resource Uniform Random Number Generator Using LUT-SR

A. Jainie Janet Jenitta[1], A. Beno[2]
[1]PG Scholar, VLSI Design, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur.
[2]Assistant Professor (S.G), ECE Department, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur.

Abstract— The uniform random number generator plays a vital role in cryptography. The aim of the current work is to achieve the maximum period of $P=2^{2048} - 1$. The proposed uniform random number generator designed for efficient implementation in lookup tables (LUT) based architecture. It describes a type of uniform RNG called a LUT_SR RNG which takes advantage of bitwise XOR operations and the ability to turn the lookup tables into shift registers of varying lengths. The LUT-SR generator uses the absolute minimum resources of one LUT to generate the sequence of binary bits compared to previous high resource LUT_FIFO RNG and low quality LUT_OPT RNG, and also provides duty cycle of 50%.

Index Terms— Field programmable gate array, Lookup table, Lookuptable_Optimized Random number generator, Lookuptable_First in first out Random number generator, Linear feedback shift register, Shift register.

1 Introduction

MANY applications are reliant on random numbers, such as financial calculations in banking, space research organization, terrorism control and in military. The uniform random number generators are very efficient method in military to handle confidential informations, and also providing security for transmitting messages in the form of cryptography. In the existing Linear Feedback Shift Register (LFSR) random number generator, also called Tausworthe generators are based on linear recurrence using GF(2) arithmetic [1], [2]. The period length of this generator is low. There are two types of RNG: True random number generator (TRNG), and Pseudo random number generator (PRNG). TRNGs make use of physical phenomena, for example the jitter between oscillators to produce numbers that are fundamentally unpredictable, and are of great interest in cryptographic applications. However, such generators cannot produce high bit-rate streams of random bits, and it is impossible to repeat a simulation run with the same stimuli without storing the entire sequence of bits, so such generators are not appropriate for monte-carlo simulation [3], [4].

The mersenne twister is a PRNG that uses Generalized Feedback Shift Register which can be naturally implemented by chained registers [5]. It is based on a matrix linear recurrence over a finite binary field *F*2 and provides fast generation of very high quality pseudorandom numbers [6], [7].

In recent years some researchers have developed FPGA Optimized random number generator which is one of the family of uniform random number generator with a matrix *A* where each row and each column contains *t - 1* or *t* ones. In hardware terms this means that each row maps to a *t-1* or *t*-input XOR gate, and so can be implemented in a single t-input LUT. Thus if the current vector state is held in a register, the new vector state can be calculated in a single LUT, and an *r*-bit generator can be implemented in *r* fully utilized LUT-FF's [8], [9].

FPGA contains different types of storage elements, such as block RAM's, distributed RAM's, and shift-registers. All of these can be configured to act as fixed-length FIFO's, which delay data for some fixed number of cycles. To extend the period of a generator, it need to add state, and as well as the flipflops in logic elements. The uniform random numbers are designed by using the LUT-SR RNGs (Lookup table-Shift Register Random number generator) [9].

In hardware, there are two different methodologies such as interleaved parallelization (IP), chunked parallelization (CP) for parallelizing long period RNG's [10].

2 Proposed Work Descriptions

The Proposed LUT-SR random number generator provides a middle ground between the LUT-Opt generator and LUT-FIFO generator, by using cheap bit-wise shift-registers to provide long period and duty cycle without requiring expensive resources.

2.1 Architecture of LUT-SR RNG

The architecture of the proposed LUT-SR random number generator is shown in figure 1. The main goal of the proposed method is to achieve the maximum period of $P=2^{2048} - 1$ (i.e., $P=2^n - 1$).It can be defined as how much amount of time the sequence is repeated itself is called the period. The repeatability is one of the major requirements of the random number generator. Although this may seem to contradict randomness it is a virtue that, using the same initial conditions, the sequence exactly repeats itself.

The lookup table is a memory with a one bit output that essentially implements a truth table where each input combination generates a certain logic output. The input combination is referred to as an address. The HDL synthesizer implements an AND gate by programming the stored elements in a LUT.

Consid[illegible] depth of the FIFO [illegible] t register increases [illegible] eriod of $P=2^{2048}-1$ [illegible] pared to a previou[illegible] number generator [illegible]

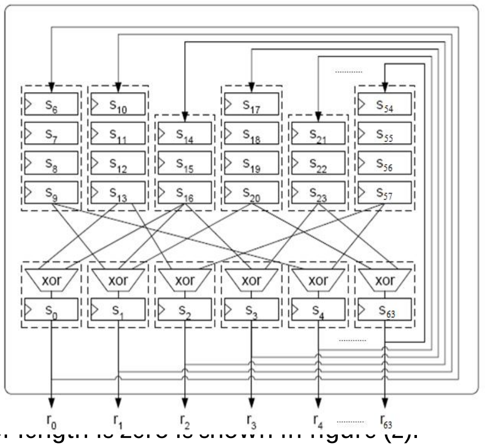

Fig 1: Architecture of proposed LUT-SR RNG

The four [illegible] number generator [illegible]

A. Create [illegible] s created throug[illegible] At this stage t[illegible] ere are *r* FIFOs [illegible] shown in figure (2).

B. FIFO Extension: The cycle is randomly extended until a total cycle length of *n* is reached, by randomly selecting a FIFO and increasing its length by 1, while maintaining the known cycle is shown in figure (3).

C. Add XOR Connections: The cycle provides one input for each of the XOR gates, so now the additional *t – 1* random inputs are added over *t–1* rounds. Each round is constructed from a permutation of the FIFO outputs, which ensures that at the end each FIFO output is used at most *t* times is shown in figure (4).

D. Output permutation: The simple dependency between adjacent bits is masked using a final output permutation is shown in figure (5).

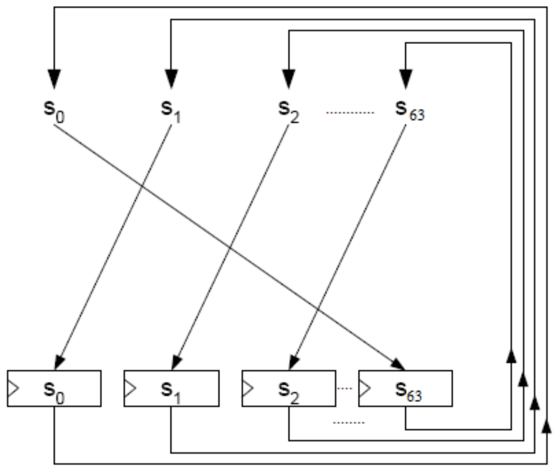

Fig 2: Create seed cycle.

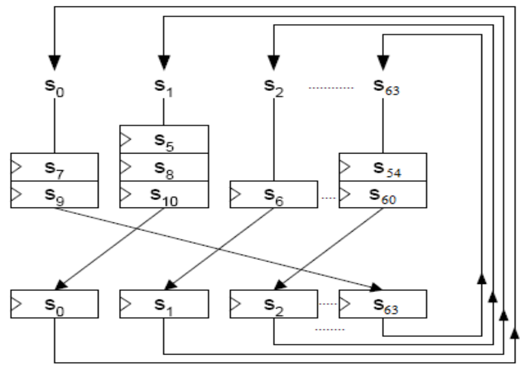

Fig 3: Randomly extended FIFO

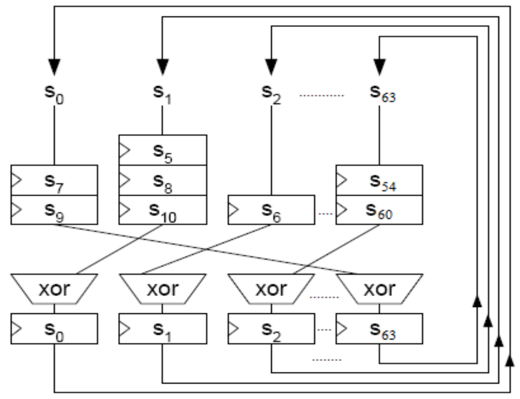

Fig 4: Add XOR Connections

2.2 Generation of Random Binary Sequences of Maximum Period

The random binary sequences of maximum length can be generated by using shift register which is composed of 16 flip-flops and appropriate feedback connections, is shown in figure (6). The order of binary zeros and binary ones depends on feedback configuration.

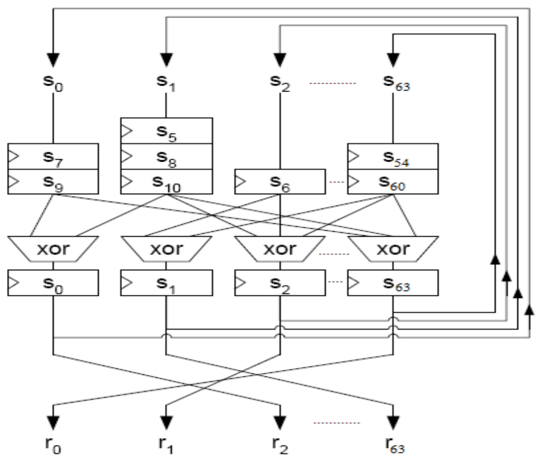

Fig 5: Output Permutation

With a proper selection of feedback, a random binary sequence of maximum length $P=2^{2048}-1$ is generated, where *n* is the number of stages in the shift register.

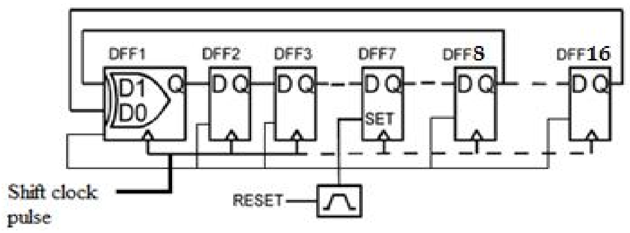

Fig 6: Generation of Random Binary Sequence.

In the configuration of random sequence generator using exclusive-OR (XOR) gates is not allowed to appear the state where all outputs of shift register are zeros, because $0\oplus 0 = 0$. A properly selected feedback provides a generation of random sequences of maximum length, $P=2^{2048}-1$. The generated sequences are deterministic.

The below figure (7) shows the LUT as an addressable SR, here the clock and the reset is given as a input to the random number generator. Then the initial seed value is given to the generator is 64 bit binary value to achieve the maximum period $P=2^{2048}-1$. If the clock is 1, and also the reset is 1 means we are not getting the output of the random number. Here the generator is designed to generate the random numbers in hexadecimal form as well as binary form per generated random sequence, when the clock is 0, and also the reset is 0.

In this work, we also introduce duty cycle for power calculation. The term duty cycle (D) is defined as the ratio between the pulse duration (τ) and the period (T) of a rectangular waveform is given by,

$$\text{Duty cycle} = PW/T \qquad (1)$$

Note that the symbol τ represents pulse width (PW). In the case of a square wave the duty cycle is 0.5 (50%) since the pulses are present 1/2 the time, of a square wave.The pulse width is 1 unit of time and the period is 10 units. In this case the duty cycle is:

$$PW/T = 1/10 = 0.1\ (10\%).$$

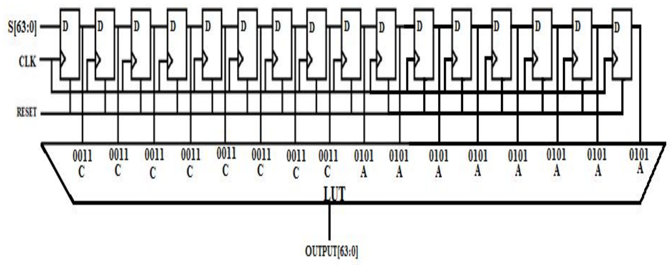

Fig 7: LUT as an addressable SR

3 Results and Discussions

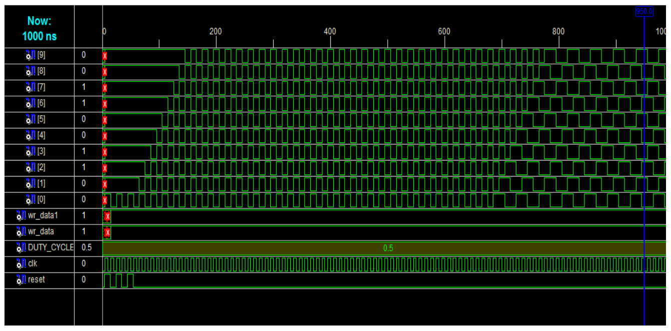

Fig 8: Random number generated using Xilinx

The VHDL simulations was carried out on the XILINX ISE 9.1i.The output simulation waveform of the proposed random number generator for the period of $2^{2048}-1$ is given in figure (8). Here the number of flip flops used in the simulation is 10 and the utilization of LUT is used in the simulation is 1 LUT.

Here the simulation shows the clock and the reset is given as an input to the random number generator. The generated random numbers are shown in hexadecimal form per generated sequence. The output waveform for the random number generator is obtained when the clock is 1 and reset is 0 (i.e 1, 0). This is the output waveform of the proposed random number generator for the period of $2^{2048}-1$.

4 Conclusion

A family of uniform RNG is called LUT-SR RNGs. These RNGs take advantage of the ability to configure LUTs as independent shift registers, allowing high-quality long-period generators to be simulated using only a small amount of logic for the period of $2^{2048}-1$ and also achieve the duty cycle of 50%.

5 References

[1] B. Thomas and W. Luk, *"High quality uniform random number generation using LUT optimised state-transition matrices," J. VLSI Signal Process.*, vol. 47, no. 1, pp. 77–92, 2007.

[2] P. L'Ecuyer, "Tables of maximally equidistributed combined LFSR generators," Math. Comput., vol. 68, no. 225, pp. 261–269, 1999.

[3] F. Panneton, P. L'Ecuyer, and M. Matsumoto, "Improved long-period generators based on linear recurrences modulo 2," *ACM Trans. Math.Software*, vol. 32, no. 1, pp. 1–16, 2006.

[4] D. B. Thomas and W. Luk, "FPGA-optimised high-quality uniform random number generators," in Proc. Field Program. Logic Appl. Int.Conf., 2008, pp. 235–244.

[5] M. Saito and M. Matsumoto, "SIMD-oriented fast mersenne twister: A 128-bit pseudorandom number generator," in Monte-Carlo and

Quasi-Monte Carlo Methods. New York: Springer-Verlag, 2006, pp. 607–622.

[6] M. Matsumoto and Y. Kurita, “Twisted GFSR generators II,” ACM Trans.Modeling Comput. Simulat., vol. 4, no. 3, pp. 254–266, 1994.

[7] Shrutisagar Chandrasekaran “High Performance FPGA implementation of the Mersenne Twister” *IEEE International Symposium on Electronic Design, Test & Application*,2008.

[8] David B. Thomas”High Quality Uniform Random Number Generation Through LUT Optimised Linear Recurrences” IEEE transaction, 2005.

[9] D. B. Thomas and W. Luk, “FPGA-optimised uniform random numbergenerators using luts and shift registers,” in *Proc. Int. Conf. Field Program. Logic Appl.*, 2010, pp. 77–82.

[10] M. Zhang “A Hardware framework for the fast generation of multiple long period random number streams”, ACM transaction, 2008.