Survey on Hardware Implementation of Random Number Generators: Theories and Experiments Analysis

Mohammed Bakiri[1] and Christophe Guyeux[2]

[1] Center for Development of Advanced Technologies, Baba Hassan, Alger, Algeria
[2] FEMTO-ST Institute, UMR CNRS 6174, University of Franche-Comte, 25000, France.
mbakiri@cdta.dz,Christophe.Guyeux@femto-st.fr

Abstract. this paper introduce a survey on PRNG on FPGA.. . .

Keywords: Random Number Generator, PRNG, TRNG, Chaotic PRNG, Cryptography, Security, FPGA

1 Introduction

Accordingly, for cryptography applications, PRNGs must have the three characteristics. The first of these, the length of the seed used in the RNG should be large enough (say, 2200 or more), so that an adversary cannot simply run through all possibilities, the Random number generators based on linear recurrences modulo 2 are among the fastest long-period generators currently available but very fast RNGs with huge period length can indeed be constructed this way. The second, the output sequences of random numbers should be statistically indistinguishable from random sequences. The third, random numbers should be unpredictable to an adversary with limited system resources. Should not waste memory (the state should be represented in no more than roughly log2 bits of memory) and allow efficient jumping ahead in order to obtain multiple streams and sub-streams. But these properties do not suffice to imitate independent random numbers.
Chaos theory is well-studied in mathematics and nature phenomenal that widely exists in nonlinear systems [1, 2, 3]. It has many exceptional good properties such as the sensitivity to initial conditions and parameters, the pseudo-randomness, the topological transitivity, being irregular, non-periodicity, unpredictability, and ability to reciprocal synchronization [4]. However, in practice these solutions have also some important drawbacks and limitations, because the chaotic nature of generated series is non-ideal, due to the limited precision of arithmetic operations and quantization. As a result, instead of random number sequences we get pseudo-random or periodic series. In practice, the chaotic nature of the obtained series is non ideal, due to the finite precision of arithmetic and quantization. As a result we get pseudo-random or periodic series.

2 Terminologies and Basic Recall

Some definitions and Symbols resume

2.1 Definitions

Definition: A random bit generator is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits

3 Random Number Generator: Theories and Classification

We can find many implementation of RNG in Software and Hardware but all can be classifies generally as PRNG, TRNG and Hybrid, but new concept has been introduce this last year's defined by parallel and chaotic generator.

3.1 PRNG

Some of the algorithms that generate pseudo random numbers are Blum Blum Shub, Inversive congruential generator, ISAAC (cipher), Lagged Fibonacci generator, Linear congruential generator, Linear feedback shift register, Mersenne twister, generalized feedback shift register (GFSR), twisted GFSR (TGFSR), Multiply-with-carry, Well-Equidistributed-Long-period Linear, Xorshift and Cellular Automata. As well as separately implementation of these structures, with use one or several of them, hybrid PRNGs can be designed. An important property of all these generators is that they are special cases of a general class of generators whose state evolves according to a (matrix) linear recurrence modulo 2 and the bits that form the output are also determined by a linear transformation modulo 2 applied to the state.

LFSR The initial value of the LFSR is called the seed, and by changing the seed we can change the sequence 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. That feedback function is called a maximal length polynomial. The bits in the LFSR state which influence the input are called taps. This is called the feedback polynomial or characteristic polynomial. it is known that an LFSR with more taps produces a better sequence of random numbers. However, on an FPGA, adding more taps minimizes the number of LUT-based shift registers that can be utilized. For example, in 4 bit LFSR if the taps are at the 4th and 3rd bits (as shown), then the feedback polynomial is x4 +x3 +1.

A nonlinear feedback shift registers NFSRs must be included in a key stream generator design to remove the linearity in the encrypted

BBS 1.
In 2012, Khushboo Sewak and all. The main purpose of this paper is to study the FPGA implementation of two 16 bit PN sequence generator namely Linear Feedback Shift Register (LFSR) and Blum-Blum-Shub (BBS). The use of feedback shift register permits very fast generatio PN sequence whereas BBS method requires a number of time consuming arithmetic operation as it is based on quadratic Congruential equation.

Mastrine Tewister 2.
In 2013, Shengfei Wu and al In this paper, a hardware architecture for the generation of parallel long-period random numbers using MT19937 method was proposed. Most hardware implementations of MT19937 are straightforward non-parallelized implementations of the original C-code. The Mersenne Twister Method, which is a pseudorandom number algorithm based on a matrix linear recurrence over F2, is developed by Makoto Matsumoto in 1997. In order to get the long period and good equidistribution, the Mersenne Twister is cascaded with a tempering transform to compensate for the reduced dimensionality of equidistribution, the temper is defined in the case of Mersenne Twister. We use dual-port BRAMs in FPGA for the implementation and 3 degrees parallelization will be introduced as an example.

Cellular Automata In 2004 [9] Guan et al. proposed a one dimensional CA where the rule in each cell changes dynamically based upon the states of the cells within a new neighborhood of three cells . Dubbed Self-Programmable Cellular Automata (SPCA), the rules are switched between 90 and 165 or 150 and 105. These rules were selected because they can be easily implemented with XOR gates. Certain combinations of rules and neighborhoods were shown to produce maximal length sequences with good quality random numbers.

In 2006 Leonidas Kotoulas and al, propose a one dimensional (1-d) Cellular Automaton (CA) for pseudorandom number generation. The proposed 1-d CA is based on the real time clock sequence and used for stream cipher. The authors show that by using rules based computer times sequence as year, months to seconds can generate initial state and the lengh of CA cells. the initial state configuration and simultaneously the length of CA cells the product of all the above numbers, namely day, month, year, hour, min and seconds was calculated. The execution time was decided to be t = x(60 - x). The first rule arises from the product of minutes by seconds. The second rule is the number of minutes divided by the number of seconds multiplied by a constant.

In 2009, Ding Jun and al, an inefficient PRNG based on the classical CA rule 30 is reported, because it is chaotic. Thus, we mainly use the rule 30 to achieve the most cell units, also provide an interface of the update rules to achieve other cells. The ca-prng is a CA with 32 cells, implemented as a 32 bit wide register. Each register has separate update logic that looks at the current state of the register and its two nearest neighbors (with wrap around). The total state update latency for all cells is thus one cycle. The actual update of the registers

is controlled by external control signals that allow a user to set the register initial pattern (state) and request generation of new pattern. Loading of initial pattern is accomplished by setting the input-pattern-data port to the desired initial pattern and then asserting the load-input-pattern port for one clock cycle. Requesting a new pattern is accomplished by asserting the next-pattern port. After reset the ca-prng will use rule30 as the update rule. Changing the rule is done by assigning the new rule to the update-rule port and then asserting the load-update-rule for one clock cycle. The generated pattern is available as a 32 bit value on the prng-data port. Figure 3 shows input and output ports of the Pseudorandom Sequence Generator.

In 2010 Ioana Dogaru and Radu Dogaru. This paper introduces a methodology based on algebraic normal form representations and software tools to rapidly generate the VHDL code description of elementary hybrid cellular automata with arbitrary parameters. They claim that One problem with a specified initial state xi (0) for all CA cells is X(t+ 1)= (mi) xor [Cell(Xi1(t), Xi(t), xi+1(t)), ID)] and the output y = Cell(u1, u2, u3, ID) is that for each particular ID one needs to determine a proper logic expression and rewrite the VHDL code accordingly. an alternative representation of Boolean functions exists and it is called an Algebraic Normal Form (ANF) representation [15]. For the case of any elementary CA (with cells defined as 3-inputs Boolean functions) the ANF representation is: y = [K0 xor K1(U1) xor K2(U2) xor K3(U3) xor K4(U1*U2) xor ... K1(U1*U2*U3)] xor mask if the number of cells is 8 and use the mask to maximize the period of the operation cycle. In 2010 [11] and [9] R. Drago, HCA101 was presented a comparison between chaotic map based on HCA and logistic map with finite precision implementation, from the perspective of using them as PN (pseudo-random) sequences in communication systems. It was shown that HCA-map is superior in any respect to the logistic map and in addition with the binary synchronization property allowing simplifying the acquisition circuits needed to reconstruct the phase of the PN sequence in the receiver.

In 2003 Tkacik proposed a hardware random number generator implemented on a custom IC which combines the outputs of a CA with an LFSR. A hybrid 90/150 rules an 37 bit CA was combined with a 43-bit LFSR. This maximal length configuration combined 32 bits from the CA and LFSR to produce a maximal length RNG. It was found that the LFSR and CA must be clocked at different frequencies to create a sequence of numbers that can pass all the DIEHARD tests.

In 2007, Petre Anghelescu and al, a hardware implementation in a FPGA circuit of an efficient encryption algorithm based on Cellular Automata. The proposed encryption system it is realized using a combination of two CA. We use a first CA as a key stream generator, a CA pseudorandom number generator (PSRG) that combines in some way two rules (the rules 90 and 150), to provide the key sequence. The block cipher algorithm presented in this paper is constructing using the second CA from class II, with rules 51, 153, 195. The operation of CA can be represented by a state-transition graph. Each node of the transition graph represents one of the possible states of the CA. In this encryp-

tion system we use as valid combination only the configurations that generate cycles of length 8. The switches are activated with the help of the first CA, so the first CA is designed to provide encryption rules for the encryption system. In the proposed 8-bit block cipher scheme, the first CA (pseudo-random number generator) must have 9 bites. First bit it is used to generate the signal So and the other 8 bits are used for SI signals (one bit for each cell of the second CA). In concordance with the CA theory, a single basic CA cell was designed. It consists of a D flip-flop and a logic combinational circuit (LCC). The LCC consists of multiplexers and XNORs to implement the rules of CA and control the loading of data and operation of the CA. When the load control signal (LoadData) is "logic 1", data is loaded into D flip-flop. When LoadData is "logic 0", data is run into the cell according with the rules applied to the rule control signals (S1, SO) and the states of neighbourhoods. After an established number of cycles (1 to 7), the data on the Q output of the flip-flop is sent out and new data is loaded in.

In 2012 Juan C.Cerda and al, This paper explores the implementation of an efficient hybrid configuration which combines the bit streams from an LFSR and a CA. The last bit of the LFSR and CA with rules 90/150 are combined in an XOR gate to produce a single random bit per clock cycle. There is a trade off between the degree of site spacing and the throughput of the PRNG. Configurations that can output 1 to 8 bits per clock cycle were explored in this study.

In 2013, Lakshman Raut and David H. K. Hoe, use the stream ciphers. Although they were reported as passing statistical test suites, they are difficult to scale-up for arbitrary numbers of cells and it is not very clear how they can ensure a large key space. This paper explores the combination of LFSRs and CA as the key components of an efficient stream cipher based on A2U2. The main component of the stream cipher is the key stream generator, which can be viewed as a pseudo-random number generator (PRNG). The use of an NFSR instead of an LFSR improves the resistance of the cipher from various forms of cryptanalysis, such as correlation attacks and algebraic attacks [8]. The proposed stream cipher is composed of five different blocks.

In 2014, Dogaru Ioana and Dogaru Radu. Introduce Two solutions based on hybrid cellular automata (HCA) for designing cryptographically secure PRNG and Both solutions are based on hybrid cellular automata and proved to be highly efficient in terms of resources. PRNG-A corresponds to the basic HCA model with a large number of bits (63 in this case, easily extended to any arbitrary number of bits). For such large number of cells masks can be arbitrary chosen still ensuring good cryptographic properties. PRNG-B implements a chain of two HCAs and one possible choice is to have 31 cells per each HCA.

3.2 TRNG

TRNG

3.3 Hybride RNG

Hybride

3.4 Parallel RNG

Parallel

3.5 Chaotic Rundom Number Generator

Logistic Based on earlier study in 2012, Pawel Dabal and all show us a study of HW implementation in FPGA of different chaos system as logistic and Hnon mapping and Frequency depended negative resistor FNDR. Many recent research illustrate an optimization using new methods and algorithms to have a better secure chaotic random and Area/Throutput results.

In 2014 Pawel Dabal and Ryszard Pelka, The authors propose a study of an fast Pipeline PRNG based on chaotic logistic map,and to solve the problem of the short cycle, distortion and correlation, the authors implement two version of PRNG based on simple logistic map equation using pipeline processing to have a a high operation frequency and the ability to generate sequence at different start point. The first is logLUT based on on LUT blocs and high-speed carry line of the FPGA and the second is Log DSP that use directly DSP of FPGA, however they add delays to ensure parallel sequence generation and a complex initial sequence used for a better NIST test results. As results, many configuration and tests based on delays and precision of PRNG applied to have the most combination of area, throutput and chaotic random outputs.

In 2013, Lahcene Merah and all, demonstrate by mixing to chaotic map they can increase the security against plaintex attacks, by coupling a chaotic encryption system (ENS) based on 2-D Hnon map used to generate the chaotic sequence, and control system (CRS) based on 1-D logistic map to control a multiplexer to choose the output of ENS according to the value generated by the logistic map by XORing the MSB of 32-bit of CSR with his it neighbor LSB. The results verified a good autocorrelation, sensitivity to initial parameters. However to increase NIST test and to resist more for the attacks, they process all the outputs sequence of the system in to a logic circuit that Xor the output CSR with the output system sequence following some rules.

Non-Linear Chaotic Dynamic In 2013 Hariprasad and NagaDeepa, the authors demonstrating using the reseeding technique to avoid non-lineair chaotic PRNG as short-period problems, and that by mixing Reseeding module with a non-linear chaotic logistic map (CLM) using a vector mixer module based on auxiliary linear generator ALG. For each fixed point condition by comparing the Xt and Xt+1 sequence of the CLM we increase the reseeding period until it reached, and then pass the result Xt+1 sequence to vector mixing to generate the final output bu XORing it with ALG (Yt+1) OUTt+1=Xt+1[1:31] XOR Yt+1[1:31].

In 2011, Pawel Dabal, Ryszard Pelka. This paper presents results of studies on the implementation of pseudo-random bit generators based on a nonlinear dynamic chaotic system. Design of chaotic PRNG: The logistic function generator was designed using MATLAB/Simulink with System Generator tool which offers ready to use library of fixed-point arithmetic blocks, that can be directly implemented into the FPGA device. In both cases we need a multiplier and a subtraction block. The multiplier unit was built using DSP48E blocks that can perform 18x25 multiplication.

In 2010, Ziqi Zhu and Hanping Hu. a high efficiency dynamic nonlinear transform arithmetic, which is used to improving the properties of chaosbased PRNG, is designed. Therefore, the DNT module 1 is processing next element of the sequence while the second one is processing the output number of DNT module 1. So the whole system's efficiency is determined by the efficiency of each DNT module. And for each one, the parallel structure guarantees the high efficiency of DNT module.

Spationtemporal Chaos In 2009, Yaobin Mao and all. To meet these needs, a spatiotemporal chaotic map is digitized to develop a highly paralleled PRBS generator that accommodates to FPGA (Field Programmable Gate Array) implementation in present paper. in this paper, a spatio-temporal chaos based PRBS generator suitable for FPGA chip implementation is suggested. Compared with traditional chaos based PRBS generators that usually realized serially by software, the proposed generator can simultaneously generate multiple pieces of bit sequences by hardware. The intrinsic operational parallelism of the PRBS generator makes the hardware implementation fairly easy and efficient. Elementary test results show that the throughput of the designed chip can reach high up to 512 Mbits/s under a running condition of 50 MHz clock frequency. Four kinds of mathematical models are usually used to represent spatiotemporal chaotic systems. They are partial differential equation (PDE), coupled ordinary differential equation, coupled mapped lattice (CML) and cellular automata respectively, among which the CML is most widely used due to its appropriate tradeoff between the calculational complexity and the representative of original system. Therefore, a pseudo-random number generator employing CML would achieve high operation speed. Since in above mentioned PRBS generating scheme only integers and some simple arithmetic and logic operations are used, it is easy to be implemented on a chip.

Fibonacci post-processing In 2013, Abhinav S. Mansingka. This paper presents a hardware implementation of a robust non-autonomous hyperchaotic-based PRNG driven by a 256-bit LFSR. The original chaotic output is post-processed using a novel technique based on the Fibonacci series, bitwise XOR, rotation, and feedback. This paper considers a new approach to hardware post processing using XOR operation and variable rotation based on Fibonacci series through two feedback loops. chaotic output passing successfully all NIST SP. 800-22 tests. The system is verified on a Xilinx Virtex 4 FPGA achieving throughput up to 13.165 Gbits/s for 16-bit implementations surpassing previously reported CB-PRNGs. Digital Realization, The proposed 4-D hyperchaotic non-autonomous

system is described by the following state space matrix representation of four first order ordinary differential equations. This particular system is digitally implemented in hardware by realizing the numerical solution of the ODE. In this section, a new post processing technique is proposed that uses two rotations and XOR feedback loops: the first loop suppresses short-term predictability using a fixed rotation while the second enhances differential sensitivity using a variable rotation controlled by the Fibonacci series. Feedback Loop 1: Static Rotation Loop 1 implements a rotation and subsequent XOR in feedback. The rotation amount is fixed at 1-bit, axiomatically guaranteeing that the rotation amount and the total input width (64-bits) are relative primes. If loop 2 is neglected, the resulting output after C cycles at the m-th bit position is. Feedback Loop 2: Variable Rotation with Fibonacci Series, The rotation amount is specified by a Fibonacci series. The native chaotic system suffers from short-term predictability and thus fails the tests. Post processing guarantees full output bus width passes the NIST tests. Introducing two registers in the post processor segments the combinational logic, the XOR and the variable rotation using Fibonacci series, which reduces the delay introduced by the post processor and preserves the output to be pipelined. While the LFSR and the proposed post processing have added a significant hardware cost.

FPGA Implementation and Evaluation of Discrete-time Chaotic Generators Circuits

In 2012, Pascal Giard and all. In this paper, implementation of discrete-time chaotic generators widely used in digital communications is studied and evaluated. To our best knowledge, there is no paper which helps us select the optimal chaotic generator based on engineering priorities for a given FPGA. The chosen discrete-time chaotic generators are: 1) Bernoulli map, 2) Chebychev map, 3) Tent map.

4 Rundom Number Generator:Statistic Test

Statistic Test

5 Rundom Number Generator: Cryptography Secure

Cryptography Secure

6 Rundom Number Generator: Hardware Implementation

Hardware Implementation

7 Conclusion

Conclusion