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, 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 length 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.

Where In 2009 Ding Jun and al, implement an efficient PRNG based on the classical CA with 32 cells using rule 30 is reported and prove a high PRNG performance.

In 2010 Ioana Dogaru and Radu Dogaru, create an automatic software tool based on Algebraic Normal Form (ANF) representation to generate an RTL code of an hybrid CA depending on ID rules. the results show by using ANF representation $y = [K_0 xor K_1(U_1) xor K_2(U_2) xor K_3(U_3) xor K_4(U_1 * U_2) xor ... K_7(U_1 * U_2 * U_3)] xormask$ with ID=101 and 3 neighbor are identical to Matlab results.

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. Then, in 2012 Juan C.Cerda and al, notice in Tkcide [2003] the combination must be clocked at different frequency to pass all NIST test, and present another combination PRNG using HCA using 90/150 and LFSR to solve this problem. The trick is XORing the last bit of HCA with the last bit of LSFR to generate 1-bit per clock cycle, and they found the best combination for a high quality of PRNG is 37-bit LFSR with 16-bit CA. then 2012, they compare they preview work LFSR/HCA with a SPCA 2004 that use 90/156 rules, and they find that even SPCA fail in one statistic test but give a better throughput than the LFSR/HCA

In 2007 Petre Anghelescu and al, propose a combination of two logic combinational circuit of Hybrid HCA where PRNG and block cipher for an encryption system, and the first HCA-1 use two rules 90/150 as a real-time key stream generator and the second HCA-1 use 51/153/195 rules. To select witch rules will be used by the block cipher HCA-2, the PRNG or HCA-1 generate an encrypting rules to switch the rules and that provide each cell has is own rules.

In 2013 Lakshman Raut and David H. K. Hoe, show us another stream cipher design and combination of CA and LFSR but they introduce NLFSR block based on A2U2 design to resist more for a various forms of cryptanalysis, such as correlation attacks and algebraic attacks. Where CA and NFSR are both has feedback for each other and use a LFSR counter as input random, then the key bit stream will pass to a mixer mechanism to increase the complexity for decryption.

In 2014, Dogaru Ioana and Dogaru Radu. introduce a comparison o two implementation of HCA as PRNG to maximize efficiency/throughput, where the first is a basic 63-bit HCA and the second is a chain of HCA(2 HCA) and they demonstrate a high ration of frequency/ares and cryptography by using a chain of HCA instead of single HCA.

3.2 TRNG

PLL: In 2008 Michal Varchola and al, To demonstrate advantages of this platform a TRNG based on internal PLL was implemented, but TRNGs based on other principles can be also tested. Great advantage of Actel Fusion FPGA is the possibility to create whole system with on-chip analog measurements. The basic principle behind the TRNG shown in Fig. 3 is to extract the randomness from the jitter of the clock signals synthesized in the embedded analog PLLs [2]. The jitter is detected by the sampling of a reference signal CLJ using a rationally related (clock) signal CLK synthesized in the on-chip analog PLLs with frequencies. It was observed that 16-bit data could be transmitted each 2 periods of CopreMP7 system clock by ModelSim simulation.Two configurations of TRNG have been

tested operating at following bit-rates - 40 kbps and 1 Mbps. Slower 40 kbps TRNG pass the test very well but faster 1 Mbps does not

In 2011 Martin Simka and al, In the paper we analyze behavior of the Phase-Locked Loop (PLL) based TRNG in changing working environment. The frequency of signals synthesized by PLL may be naturally influenced by chip temperature. We show what impact the temperature has on the quality of generated random sequence of the PLL-based TRNG. When considering PLL as a source of randomness, three parameters have to be taken into account: 1) the size of generated clock jitter; 2) available ranges of frequency dividers n, m and k together with the VCO maximum and minimum frequency; 3) bandwidth of the PLL loop filter. We can conclude that lower bandwidth of the feedback PLL loop in the configuration B causes higher number of the critical samples. This can be explained by the fact that lower bandwidth decreases PLL output jitter and thus increases the tracking jitter.

In 2004 Paul Kohlbrenner and Kris Gaj, In this paper we extend a technique that uses on-chip jitter and PLLs to a much larger class of FPGAs that do not contain PLLs. Our design uses only the Configurable Logic Blocks (CLBs) common to all FPGAs,The source of randomness for our TRNG is two ring oscillators. A Phase Locked Loop (PLL) present on Altera FPGAs was used to produce the two clock signals used in this technique. Our proposed design as shown in Figure 2 consists of two independent and identically configured ring oscillators, a sampling circuit, and a control circuit. By design this configuration exactly fit in one Virtex CLB slice. The sampler circuit extracts the jitter contained in the signals from the two ring oscillators. An important secondary benefit of the control circuit is that it prevents any output from the TRNG if the difference between the cycle lengths of the two ring oscillators is too great. it is important to place the two ring oscillators close to each other.

In 2006 Martin Simka and al, This paper summarizes possible TRNG configurations and relation between PLL and TRNG parameters. based on extraction of the random tracking jitter using two rationally related clocking signals. The basic principle of the true random number generation by an extraction of the tracking jitter included in clock signal CLJ using clock signal CLK is illustrated. There are three options how the PLLs can be configured in the TRNG in dependency on chosen FPGA: with one PLL, with two parallel PLLs and with two (or more) cascaded PLLs. Each configuration permits to achieve different characteristics (defined in [5]) depending on parameters of PLLs, namely maximum input, output and voltage-controlled oscillator (VCO) frequency, multiplication and division factors, etc. and in this way the needed frequency can be synthesized. use of two PLLs in either parallel or serial (cascaded) configuration can increase significantly sensitivity on the jitter and the output bit-rate of the generator.

RO: In 2006 Drie Schellekens and al, a review on Ring oxialltor definition [6][7]

In 2009 Cristian Klein and al, This paper focuses on the design and implementation of a high-quality and high-throughput true-random number generator (TRNG) in FPGA. Our first attempt was to create a TRNG based on [3], which

uses one RO to sample the output of the other RO. We appreciated this approach due to the fact that the whole stream before the post-processing phase is random (although it might be biased a little bit). A post-processing phase is required which consists in a resilience function. A single inverter allows us to create ROs which both even and odd number of latches. To make sure that the inverter does not add more delay the inverter and the first latch are mapped to the same CLB. we chose as the resilience function a simple XOR of 2r-bits. increases the output period of the ring oscillators, it also increases the amount of jitter.

In 2003 K.H.Tsoi and al , Two FPGA based implementations of random number generators intended for embedded cryptographic applications are presented. The first is a true random number generator (TRNG) which employs oscillator phase noise, and the second is a bit serial implementation of a Blum Blum Shub (BBS) pseudorandom number generator (PRNG). the TRNG, oscillator phase noise was used. our implementation uses a very high frequency clock (up to 400 MHz) and does not require a scrambler to achieve good random output. There are several factors which affect the randomness of the output [22]. The first situation is that the duty cycle of Fh may not be 50performed once only during initialization, and after that, a squaring and modulo operation are performed each iteration to produce Xi, In total, each iteration of the PRNG requires $(4.5 * n^2 + n)$ clock cycles, where n is the size of the modules in bits.

In 2009 KnutWold and Chik How Tan, In this paper, we analyze the TRNG designed by Sunar et al. (2007) based on XOR of the outputs of several oscillator rings. We propose an enhanced TRNG with better randomness characteristics that does not require postprocessing and passes the statistical tests. We have shown by experiment that the frequencies of the equal length oscillator rings in the TRNG are not identical. The difference is due to the placement of the inverters in the FPGA and the resulting routing between the inverters. In this paper, we examine more closely the TRNG based on oscillator rings proposed by Sunar et al. [2]. We show that the TRNG described in [2] is not random without postprocessing our TRNG has no bias and, therefore, no need for complicated postprocessing. The TRNG consists of several equal length oscillator rings connected to an XOR tree. The output from the XOR tree is sampled by a D flip-flop, and the output signal of the D flip-flop is then postprocessed in order to increase the entropy and remove bias from the random signal. The entropy source of the TRNG is the jitter created by each oscillator ring. The main concern of the authors of [10] is that the XOR-tree and the sampling D flip-flop cannot handle the high number of transitions from the oscillator rings. we suggest an enhancement of the TRNG based on the oscillator rings in [2] by adding an extra D flip-flop after each ring. The frequency will increase with the decreasing number of inverters. The tendency is that the bias increases with the increasing number of rings. however equal length oscillator rings will have the same frequency is not true when the dispersion is decreasing with increasing number of inverters. for restart experiment, the first data bits should be omitted in order to have good quality of the random sequence.

IN 2007 Markus Dichtl and Jovan Dj.Golic, The main objective of this work is to evaluate and analyze the amount of true randomness produced by these oscillators. This is achieved by using the restart approach, which consists in repeating the experiments from identical starting conditions. Fibonacci and Galois ring oscillators [8] (FIRO and GARO, respectively) are both defined as generalizations of a ring oscillator (RO). They consist of a number, r, of inverters connected in a cascade together with a number of XOR logic gates forming a feedback in an analogous way. the feedback polynomial should be chosen to have a form $f(x) = (1+x)h(x)$. To increase randomness and robustness, it is also proposed to use an XOR combination of a FIRO and a GARO (FIGARO). compare them with a classical RO based inverters, show very clearly that the classical ROs need more than 5 s until they reach an approximately stable value and the FIROs and GAROs achieve a more or less stable standard deviation of their output voltages already after about 50 ns.

STR:

In 2013 Abdelkarim Cherkaoui and al, The proposed true random number generator (TRNG) exploits the jitter of events propagating in a self-timed ring (STR) to generate random bit sequences at a very high bit rate. the stochastic models are not feasible or at least not plausible, because they combine intrinsically pseudo randomness with true randomness. In [9], we showed for the first time that self-timed rings (STR) are a highly suitable source of entropy. Based on these observations, in [10] we proposed the first TRNG principle based on STRs. Self-timed rings (STR) are oscillators that can provide events which are evenly spaced in time and distributed over half an oscillation period of one ring stage. Contrary to inverter ring oscillators, several events can propagate simultaneously in STRs thanks to the asynchronous hand- shake protocol. The signals resulting from the STR outputs are synchronized and their mutual position does not change over time. In contrast, the ring oscillator output signals from [7] drift in time and generate pseudo-randomness.

metastability:

3.3 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.

In 2011 Pawel Dabal and Ryszard Pelka implement two version of generator using XSG tool based on logistic mapping equation but distribute the equation $X_{t+1} = r * X_n * (1 - X_n)$.

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 X_t and X_{t+1} sequence of the CLM we increase the reseeding period until it reached, and then pass the result X_{t+1} sequence to vector mixing to generate the final output bu XORing it with the outputs Y_{t+1} of ALG , $OUT_{t+1} = (X_{t+1}[1:31])XOR(Y_{t+1}[1:31])$.

Du to degeneration phenomenon and for a high quality of chaotic of PRNG, In 2010 Ziqi Zhu and Hanping Hu present a new high efficiency dynamic nonlinear transform arithmetic DNT. A chaos system based on three DNT process in a parallel structure that transform input 3 times and improve a high cycle-length and distribution output sequence, and that of each DNT module initiate his 2x256 code book and obtain the binary input sequence, then transformed and look up it using inputs as parameters $B_{i+1} = Bi(C(w)R(q))$.

Spationtemporal Chaos In 2009 Yaobin Mao and all, implement a new parallel PRNG based on digitized spatio-temporal chaos map using coupled mapped lattice (CML) model. To achieve a high operation speed, they first deal with continuous domain with digitazed all operand to be suited for HW implementation by using and modifying a bi-directional coupled chaotic map lattice to a finite integer set. then second and to avoid finite precision chaotic map problem, they compute only the significant bit is subject to output and randomly select initial value for each lattice.

Fibonacci post-processing In 2013 Abhinav S. Mansingka and al, present a post processing based on Fibonacci series based on LFSR for non-autonmous signum hyperchaotic PRNG. This paper presents a hardware implementation of a robust non-autonomous 4-D hyperchaotic-based PRNG driven by a 256-bit LFSR. The post processing is based on two loop feedback using as first a

fixed 1-bit static rotation to suppress the short-term predictability, then the second is based on a variable rotation controlled using Fibonacci series of K-bit to enhances differential sensitivity if there is a change at any bit when the other bit propagate during M Cycle, $M = (i : K_i < 64, K_{i+1}, iEZ^+)$.

4 Rundom Number Generator:Statistic Test

Statistic Test

RNG	**Algorithme**	**DieHard**	**NIST**	**FIPS**	**Test01**
Pseudo Randum Number Generator **PRNG**	LFSR	[A][E]	[B]	[C]	[D]
	BBS	[A]	[B]	-	-
	MT				
	CA				
	Custom PRNG				
Other Randum Number Generator	TRNG	[A]	[B]	[C]	[D]
Chaotic Randum Number Generator	Logistic Map	[A]	[B]	[C]	[D]
	Henon				
	CI				
	Others CRNG				

5 Rundom Number Generator: Cryptography Secure

Cryptography Secure

6 Rundom Number Generator: Hardware Implementation

Hardware Implementation

7 Conclusion

Conclusion