Deepika Gurajapu, D.Nagesh / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 5, September- October 2012, pp.1067-1071

FPGA IMPLEMENTATION OF NON-LINEAR PRNG USING RESEEDING- MIXING METHOD

Deepika Gurajapu, D.Nagesh,
M.Tech (student), Pragati Engineering College, Surampalem
Assistant professor, Pragati engineering college, Surampalem

ABSTRACT

We present a new reseeding-mixing method to extend the system period length and to enhance the statistical properties of a chaos-based logistic map pseudo random number generator (PRNG). The reseeding method removes the short periods of the digitized logistic map and the mixing method extends the system period length to 2^{253} by "XOring" with a DX generator. Here we also present the method to increase the randomization by changing the initial pattern periodically.

1. INTRODUCTION

The understanding of random numbers and how a number is defined to be random is essential to the understanding of this paper. In a statistical sense, there is no such thing a single random number. This is because it is impossible to perform any statistical tests on a single number. Instead it is better to define a sequence of numbers to be random, or speak of a number chosen from a random sequence of numbers. *[Knuth]*

A random number can be defined as a number from an independent set of numbers which is the output of a natural event and has a very high degree of uncertainty. The next number generated is completely independent from all previous numbers generated and is selected purely by chance. The set of numbers produced will have a given distribution *[Knuth]*. Here, it is assumed that all random sequences have a uniform distribution between zero, and one, excluding which is written as [0,1) mathematically. All possible numbers in this range has an equal probability of being selected. Randomness is easier understood as the outcome of some natural process or event. A true random number generator has an infinite period and given the same input parameters, will never produce the same output.

Secondly, a pseudo random number can be defined as a number from a set of numbers, which is the output of a mathematical function, which tries to "mimic" a true random number. This therefore makes them completely deterministic *[Jansson]*.

Poor PRNG's have been an issue since the release of IBM's RANDU generator in the early 1960's.

II .Random Numbers

The need for true random numbers is needed in many applications, this has resulted in specialized hardware being built to produce a sequence of true random numbers instead of using a software method of producing random like number sequences.

There are many natural occurring events that exhibit true random behaviour. The world of quantum physics is characterized by purely random events as well as natural decay of radioactive material are just two examples of many, truly random events. These processes, although very good sources of random data, are not practical for many computing needs. There are however more practical ways of generating true random data sequences.

One such method is by allowing a current to flow thought a resistor. Thermal agitation of free electrons causes small voltage fluctuations. It is this random noise that circuit designers try to minimise as much as possible *[Ghausi]*. If the amplitude of the voltage fluctuations across the resistor are made large enough to use practically, a true random number generator could be constructed *[Connor]*. This can be used, together with a comparator and a microprocessor to feed a sequence of "random" bits into the PC via the COM port. This sequence of bits will of course have to be tested statistically to make sure that it does actually exhibit random properties. Another possible way to generate random data is to connect an antenna to an amplifier and measure the noise picked up. This is very similar to measuring the noise from a resistor, except that the sequence collected will have to be analyzed using Fourier analysis to check that there are no dominating frequencies in the data. This does however have its drawbacks

.

A. Pseudo Random Numbers

Obtaining random numbers from a physical source can often be impractical in many applications such as portable web applications. This led to the development of a mathematical method to create a sequence of numbers that could mimic true random numbers. Because a mathematical source is not a true source of random numbers, it is called a pseudo random number. This is due to the mathematical function being completely deterministic and hence, non-random. In the 1940's von Neuman developed the first mathematical algorithm to create random

numbers. This was known as the middle-square method, and while it could produce seemingly random number sequences, it quickly proved to be a very poor source of pseudo random numbers. These methods of producing pseudo random numbers are known as pseudo random number generators or PRNG for short.

B. Current Prngs

The first reliable PRNG algorithm was proposed by D. H. Lehmer in 1949, called the Linear Congruential Generator (or Linear Congruential Method, LCM). This method has been one of the most well known and widely used methods for generating random sequences. However, this method is not without flaws. It is well know that the sequence generated forms a lattice structure in 3-space. 3-space are three sets of coordinates (3-tuple) plotted against three sets of axis. This concept can be extended to 4,5… n-space.

III. Qualities of Prng Must Possess

A good PRNG can be designed and built following a few simple guide lines. This however does not make the actual task of designed such an algorithm a menial task.

These six characteristics are as follows:

--Since processing power is not limited to the extent as it was a few decades ago, PRNG algorithms must still be short and efficient. This will allow pseudo random numbers to be generated in only a few clock cycles to allow the processor to continue with the main calling function or program *[Jansson, Atreya]*.

--Since PRNG's are of the form of a mathematical function, it is noted that they will, at some stage, begin to repeat themselves *[Park]*. It is this period of a PRNG that must be as long as possible *[Jansson, Atreya]*.

---There are statistical tests in use that can test the possibility of randomness with high levels of accuracy. The sequence produced from a PRNG should be checked against these tests, and pass them *[Jansson]*.

--By analysing the outputs of a PRNG, it should not be possible to predict the next number that will be generated *[Atreya]*.

--A random number sequence, in its binary representation, must have, on average, an equal amount of 1's and 0's. Furthermore, there must be no noticeable patterns in the bit string. *[Atreya]*.

--A PRNG must be seeded with a value, and given the same value, the same sequence of numbers must be produced (this is especially important for Monte Carlo simulations discussed later). For systems where the PRNG must behave in a more random manner, the seed must not be known or must not be able to be calculated. In this aspect, it is important that the seed contain a high level on entropy.

A. Selecting A Seed

Selecting a seed number for a PRNG can be a very important process in the correct operation of a PRNG. In some applications it is acceptable to use a predefined seeding value or, for example the time of day. In security applications, this method of seeding a PRNG is not so simple. If the seeding value can be discovered, the entire security of an application can be broken .

This technique is known as delayed coordinates and can be explained as a comb being passed through a set of numbers to pick out, or identify, any patterns in the data. This can be extended to as many dimensions as one likes also known as n-dimensional space. Three dimensions are used so that it is easy to plot on set of three axes. An extra dimension is possible in the form of displaying time data. This will show how the sequence is being generated and whether there is some pattern in the generation process. Also, by modifying the lag it is possible to find dependencies in the data that is not immediately obvious. Another very similar method as the one above is to check how dependent a number in a sequence is from its predecessor

B. Construction And Testing Of A Hardware RNG

The use of a physical device to generate random numbers can possibly be one of the most secure methods of generating random numbers for computational needs. It is not, however, suited for all applications. Monte Carlo experiments, for example, should not be performed using a physical random number generator. This is due to the fact that the same sequence of numbers can never be obtained again which will result in an experiment never being replicated exactly. This will have major implications when results need to be verified by colleagues. Instead, a good PRNG should be used with very good statistical properties. But for purposes of security and cryptography, the use of a physical device is desirable. This is because a hardware device, if properly constructed, will have a high degree of entropy.

C. PRNGs for Simulations

The output sequence of random numbers is produced in the following manner.

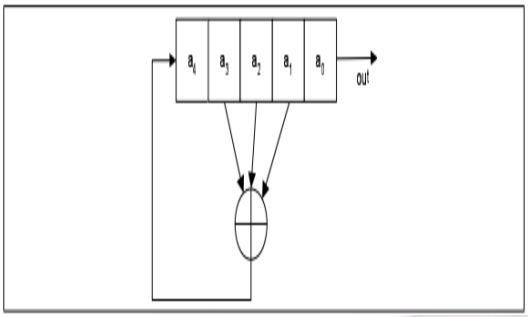
$$Xi+1 = aXi + c(\text{mod } n)$$

D. Linear Feedback Shift Register

Linear feedback shift register (LFSR) is composed of a register of n memory elements, each of them is capable of storing one bit, and having one input and one output; and a clock which controls the shift operation (the °ow of data between these elements) and the feedback operation. At each time unit the content of element 0 is output, the content of each element i is moved to element i+1, and the new

content of element n+1 is the output of the feedback function.

The feedback function is xor of the content of a fixed subset of the register elements. This subset (also called a tap sequence) is represented by a polynomial of the form



If element is used as input to the feedback function (i.e., c0 = 1), the LFSR is said to be non-singular. If the LFSR is non-singular and the polynomial is a primitive polynomial, and the initial content of the register is not all zero, then the LFSR produces output with the maximum possible period 2n+1. [21] (H16:2) listsof primitive polynomials modulo 2.

E. Generalized Feedback Shift Register

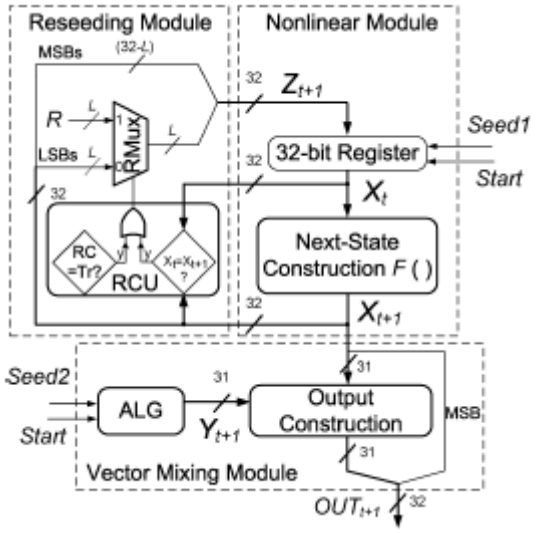
Generalized feedback shift register (GFSR) [15] is a refinement of LFSR. It is non-singular, the polynomial is primitive, and its degree is 3 (a trinomial). Among GFSR'sadvantages are: better distribution, long period, and speed.

F. Reseeding Technique

Reseeding technique is widely used in LFSR for test pattern generation [26], [27] and in CB-PRNG for period extension. Cernák [13] presented a reseeding method either to perturb the state value or the system parameter of digitized logistic map (LGM) for removing the short periods of a CB-PRNG. In 1998, Sang *et al.* applied a different reseeding method in perturbing a CB-PRNG to extend its period length up to 3.362×10^{29} and the lower bound of the reseeded system can be calculated. Li *et al.* [16] discovered that the reseeding technique not only removes the short periods but also improves the statistical properties of CB-PRNG. Recently, these merits of reseeding were confirmed by exhaustive simulation [28] in a 32-b implementation of a CB-PRNG.

IV. Proposed RM-PRNG

Fig. 1 shows the schematic diagram of the RM-PRNG, which is composed of three modules: Nonlinear Module, Reseeding Module, and Vector Mixing Module. In a 32-b implementation, the Nonlinear Module has a controlled 32-b state register and a Next-State construction circuitry. The controlled register stores the state value which can be set to Seed1 by the Start command. The Next-State construction circuitry produces the next state value according to the recursive formula. For each generated state value, the reseeding control unit (RCU) in the Reseeding Module compares the values for checking the fixed point condition,



and increases the reseeding counter (RC) at the same time. The RC will be reset and the reseeding operation will be activated when either the fixed point condition is detected or the RC reaches the reseeding period □□. When reseeding is activated, the state register will be loaded through the reseeding multiplexer (RMux) with a value described in Section III-B. Otherwise, the value of __□_ is directly loaded into the state register. The output of the proposed RM-PRNG is obtained by mixing Xt+1 with the output Yt+1 from an auxiliary linear generator
(ALG) in the Vector Mixing Module according to the rule given in Section III-C.

A. Nonlinear Module

We use the LGM as the next-state construction function in the Nonlinear Module so that

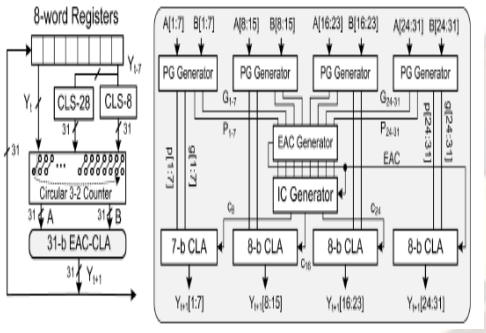
$$X_{t+1} = F(X_t) = \gamma X_t(1 - X_t), \quad t \geq 0 \tag{1}$$

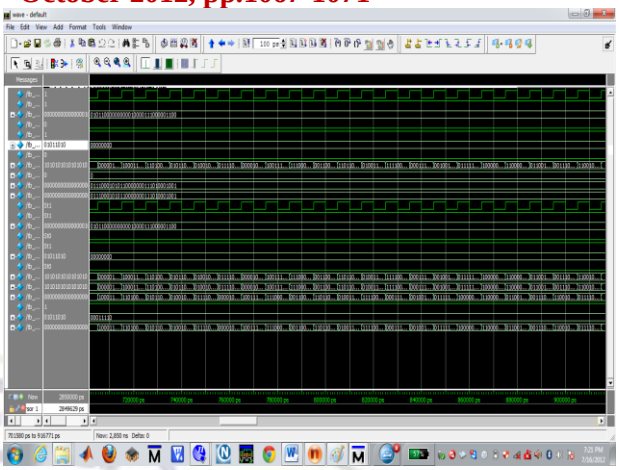
. Choosing a value 4 for not only makes the LG Mchaotic but also simplifies the implementation of (1) to merely left-shifting the product by 2 b. However, the state size decreases from 32 to 31 b, because the dynamics (1) are the same. This is equivalent to a degradation of resolution by 1 b. In addition, fixed as well as short periods exist when the LGM is digitized. From exhaustive runs for all of the seeds, we obtain all other periods for the 32-b LGM without reseeding. They are given in Table I with the longest period (18 675) and the set of short listed separately along with their total occurrences. Clearly, the performance of a CB-PRNG using only

the Nonlinear Module is unsatisfactory. To solve the fixed points and short-period problem, a Reseeding Module is in order.

B. Reseeding Module

The removal of the fixed points by the reseeding mechanism is obvious. When the fixed point condition is detected or the reseeding period is reached, the value loaded to the state register will be perturbed away from in the RCU by the fixed pattern according to the formula where subscripts ,i ,j are the bit-index, L is integer. In order to minimize the degradation of the statistical properties of chaos dynamics, the magnitude of the perturbation of the fixed pattern should be small compared Here, we set L=5 so that the maximum relative perturbation is only $(2^5-1)/2^{32}$ and the degradation can be ignored [15]. Clearly, the effectiveness of removing short-periods depends on the reseeding period as well as the reseeding pattern . However, choosing the optimal reseeding period and the reseeding pattern is nontrivial. Nevertheless, several guidelines to choose a suitable combination had been proposed and discussed in our previous work [28]. First, the reseeding period should avoid being the values or the multiples of the short periods of the unperturbed digitized LGM. Otherwise, if the 5 LSBs of equal to when the reseeding procedure is activated. Then no effective reseeding will be realized and the system will be trapped in the short-period cycle. Hence, prime numbers should be used as the reseeding period candidates. Although the average period of the reseeded PRNG has increased more than 100 times relative to that of the non reseeded counterpart, the period can in fact be extended tremendously in the Vector Mixing Module described below.





Simulated output.

Area Utilization Report

Flow Summary

Flow Status	Successful - Mon Jul 16 19:09:43 2012
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	PRNG
Top-level Entity Name	TOP_MODULE
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	77 / 33,216 (< 1 %)
Total combinational functions	76 / 33,216 (< 1 %)
Dedicated logic registers	33 / 33,216 (< 1 %)
Total registers	33
Total pins	93 / 475 (20 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Flow summary report

CONCLUSIONS

The proposed one is a hardware implementation of RM-PRNG to offer long periods and high throughput rate while adhering to established statistical standards for PRNGs. The reseeding mechanism solves the short-period problem originated from the digitization of the chaotic map, while mixing a CB-PRNG with a long-period DX generator extends the period length to the theoretically calculated value greater than 2^{253}. Replacing a hardware-demanding CB-PRNG with a hardware-efficient MRG, the hardware cost is reduced and the hardware efficiency achieves 0.538 Mb/s-gate. In addition, the high throughput rate (> 6.4 Gb/s) is attained because RM-PRNG can generate multiple random bits in an iteration

REFERANCES:

[1] J. E. Gentle, *Random Number Generation and Monte Carlo Methods*, 2nd ed. New York: Springer-Verlag, 2003.

[2] M. P. Kennedy, R. Rovatti, and G. Setti, *Chaotic Electronics in Telecommunications*. Boca Raton, FL: CRC, 2000.

[3] D. Knuth, *The Art of Computer Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1981.

[4] A. Klapper and M. Goresky, "Feedback shift registers, 2-adic span, and combiners with memory," *J. Cryptology*, vol. 10, pp. 111–147, 1997.

[5] D. H. Lehmer, "Mathematical methods in large-scale computing units," in *Proc. 2nd Symp. Large Scale Digital Comput. Machinery*, Cambridge, MA, 1951, pp. 141–146, Harvard Univ. Press.

[6] P. C. Wu, "Multiplicative, congruential random-number generators with multiplier □□ □□ and modulus □ __," *ACM Trans. Math. Software*, vol. 23, pp. 255–265, 1997.

[7] L. Y. Deng and H. Xu, "A system of high-dimensional, efficient, longcycle and portable uniform random number generators," *ACM Trans. Model Comput. Simul.*, vol. 13, no. 4, pp. 299–309, Oct. 1, 2003.

[8] L. Y. Deng, "Efficient and portable multiple recursive generators of large order," *ACM Trans. Modeling Comput. Simul.*, vol. 15, no. 1, pp. 1–13, Jan. 2005.

[9] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudorandom number generator," *SIAM J. Comput.*, vol. 15, pp. 364–383, 1986.

[10] B. M. Gammel, R. Goettfert, and O. Kniffler, "An NLFSR-based stream cipher," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 2917–2920.

[11] D. Mukhopadhyay,D. R. Chowdhury, and C. Rebeiro, "Theory of composing non-linear machines with predictable cyclic structures," in *Proc. 8th Int. Conf. Cellular Autom. Res. Ind.*, 2008, pp. 210–219, Springer.

[12] D. Mukhopadhyay, "Group properties of non-linear cellular automata," *J. Cellular Autom.*, vol. 5, no. 1, pp. 139–155, Oct. 2009.

[13] J. Cermak, "Digital generators of chaos," *Phys Lett. A*, vol. 214, no.3–4, pp. 151–160, 1996. [14] T. Sang, R.Wang, and Y.Yan, "Perturbance-based algorithm to expand cycle length of chaotic