

# Non-Linear RM-PRNG for the Design and Implementation of an Encryption and Decryption

D.Siva Jyothi\*, Nitin Meena\*\*

\* P.G. Student (Embedded Systems and VLSI Design), IES College of Technology, Bhopal, India.

\*\* Assistant Professor, ECE Department, IES College of Technology, Bhopal, India.

**Abstract-** Pseudo Random Number Generator (PRNG) is an algorithm for generating a sequence of numbers. Due to speed in number generation pseudorandom numbers are very important. The increasing application of cryptographic algorithms to ensure secure communications is widely used. So, Here we propose a new reseeding mixing method to extend the system period length and to enhance the statistical properties of pseudo random number generator (PRNG). The reseeding method removes the short periods which are occurred by CB-PRNG and the mixing method extends the system period length by Xoring with ALG. The output sequence of RM-PRNG is used as a key to the encryption and decryption modules. The simulation results are obtained by using Modelsim and Synthesis is observed by Xilinx ISE 10.1.

**Index Terms-** Encryption, Decryption, Reseeding, Mixing, PRNG, Period Extension.

## I. INTRODUCTION

It is hard to imagine a well-designed cryptographic application that doesn't use random numbers. Session keys, initialization vectors, salts to be hashed with passwords, unique parameters in digital signature operations, and nonce's in protocols are all assumed to be random by system designers. Unfortunately, many cryptographic applications don't have a reliable source of real random bits, such as thermal noise in electrical circuits or precise timing of Geiger counter click [Agn88, Ric92]. Instead, they use a cryptographic mechanism, called a Pseudo-Random Number Generator (PRNG) to generate these values. The PRNG collects randomness from various low-entropy input streams, and tries to generate outputs that are in practice indistinguishable from truly random streams. Typical PRNG consists of unpredictable input called "seed" value and a secret state "S". Software approaches use machine state information like movement of the mouse, keystrokes, contents of memory registers, and hardware latency to create a seed value. Prngs operate by repeatedly scrambling the seed to generate random output. Typically, the seed is a short, random number that the PRNG expands into a longer, random-looking bit stream.

A PRNG often starts in a random state and must process many seeds to reach a secure state **S**. Upon request, it must generate outputs that are indistinguishable from random numbers to an attacker who doesn't know and cannot guess **S**. In this, it is very similar to a stream cipher. Additionally, however, a PRNG must be able to alter its secret state by repeatedly processing

input values (seed). See Figure 1 for a high-level view of a PRNG.

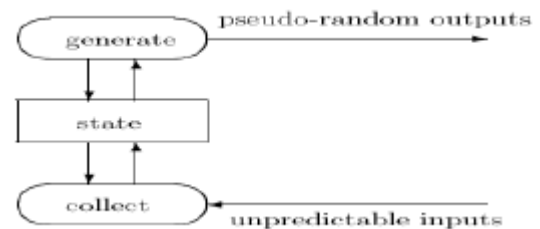


Figure.1. Model of PRNG's

Prngs are produce long period random number sequence linear prngs are very useful, some of the linear prngs are linear feedback shift registers (lfsrs), linear congruential generators (lcs), and multiple recursive generators (mrsgs). These linear prngs are good in hardware cost and throughput rate. But due to their linear structure output random numbers of these generators are easily predictable. To overcome the predictability problem nonlinear chaos-based prngs (CB-prngs) [8] were proposed, it is efficient in hardware cost, but due to quantization error there exists short periods in such nonlinear prngs. They produce only one bit per iteration hence throughput rate is low. And then to produce long periods and high throughput rate reseeding-mixing PRNG (RM-PRNG) were proposed. The RM-PRNG consists of a CB-PRNG and MRG [1], [8]. The reseeding method removes the short periods in the CB-PRNG and by mixing MRG with CB-PRNG the overall system period length increases.

In this paper, we propose a encryption and decryption technology; in this technology has an encryption scheme the message or information is encrypted by using an encryption method, the plain text is converted into unreadable cipher text. An adversary that can see the cipher text should not able to determined anything about the original message. However the cipher text is converted to plain text (original message) by using decryption method. In this encryption and decryption methods are implementing an operation of the plain text and RM-PRNG key with "Xoring" operation.

## II. RM-PRNG

Fig. 2 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 state

register stores the state value ( $X_t$ ) which is set to seed1 by using the start command. The next state construction is used to produce the next state value ( $X_{t+1}$ ) by using recursive formula  $X_{t+1}=F(X_t)$  [1].

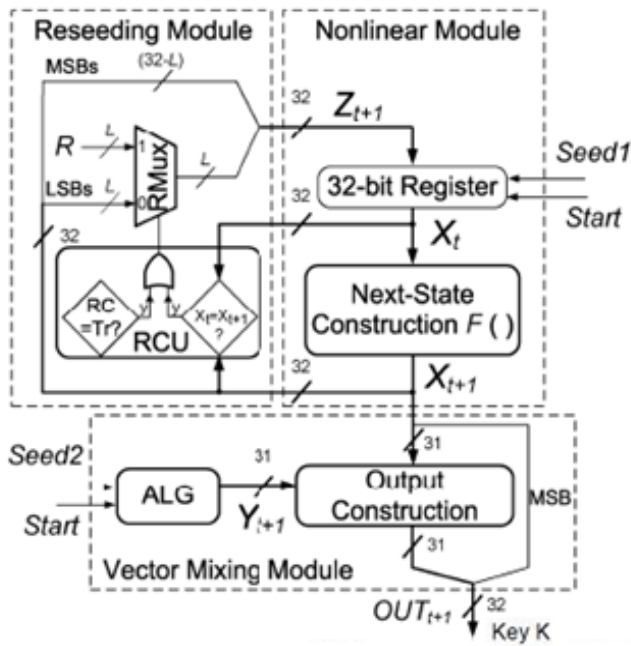


Figure.2. Structure of RM-PRNG

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 reseed 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 reseed period. When RC reaches the reseed period  $T_r$  or the fixed point condition is detected then RC will be reset and the reseeding operation will be activated. The state register will be loaded through the rmux, when reseeding is activated [1]. The value of  $X_{t+1}$  is directly loaded into the state register if the reseeding is not activated. Vector Mixing Module is implemented by an auxiliary linear generator (ALG) and output construction. By mixing  $X_{t+1}$  with the output  $Y_{t+1}$  from ALG in Vector Mixing Module, we obtain the output of the RM-PRNG (32-bit implementation).

**A. Nonlinear Module:** We use the LGM as the next-state construction function in the Nonlinear Module so that  $X_{t+1} = F(X_t) = \gamma x_t (1 - X_t)$ ,  $t \geq 0, \dots$  (1) Choosing a value 4 for not only makes the LGM Chaotic 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

$$Z_{t+1} = \begin{cases} X_{t+1}[j], & 1 \leq j \leq 32-L; \\ R[i], & 33-L \leq j \leq 32, i=j+L-32 \end{cases}$$

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  $(25-1)/232$  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. 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 lsb is 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.

**C. Vector Mixing Module:** The Vector Mixing Module is constructed by using ALG and output construction. In this module an efficient MRG which is called as DX generator acts as the ALG. By using the following recurrence formula

$$Y_{t+1} = Y_t + B_{DX} \cdot Y_{t-1} \text{ mod } M, t \geq 7$$

In output construction unit, to obtain the lsb of the output The lsb of  $Y_{t+1}$  and that of  $X_{t+1}$  are mixed by using XOR operation according to the following equation

$$OUT_{t+1}[1:31] = X_{t+1}[1:31] \oplus Y_{t+1}[1:31]$$

To form the full 32-b output vector  $out_{t+1}$  the MSB of  $X_{t+1}$  is added to  $out_{t+1}[1:31]$ .

The DX generator is implemented described below the implementation [of the DX generator is (the ALG) done by using 8-word registers, circular-left-shift (CLS), circular 3-2 counter and End Around Carry- carry look ahead adder (EAC-CLA). By using flip-flops the eight-word register was implemented. For generating two partial products signal  $Y_{t-7}$  is circular-left-shifted 28 and 8 b using the modules CLS-28 and CLS-8 respectively. To combine these three 31-b operands into two 31-b operands a circular 3-2 counter is used, which Consumes 247 gates. To

evaluate  $Y_{t+1}$  31-b EAC-CLA is used with 348 gates. The schematic design of the 31-b EAC-CLA [4], [9] is shown in Figure 2(b). The schematic design of the 31-b EAC-CLA includes four modules they are propagation and generation (PG) generators, end-around-carry (EAC) generator, internal carry (IC) generator, and clas [5]. When EAC is generated by group of pgs, EAC is then fed to the IC generator and then to least-significant 8-b CLA. On clas, the final addition was performed.

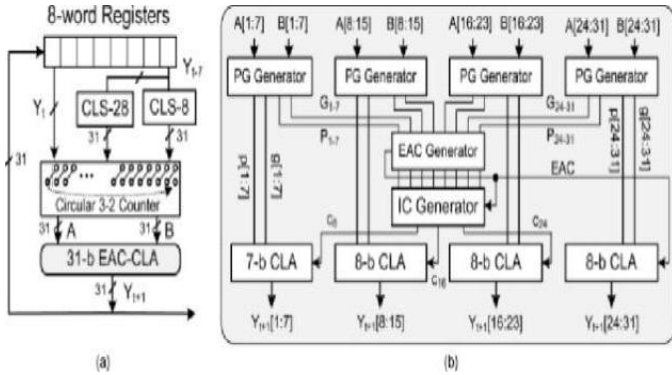


Figure.3: (a) Structure of the DX generator  
 (b) Structure of the 31-b EAC-CLA.

III. PROPOSED ENCRYPTION AND DECRYPTION

In this proposed Encryption and decryption technology is where security engineering meets mathematics. It provides us with the tools that underlie most modern security protocols. It is probably the key enabling technology for protecting distributed systems, yet it is surprisingly hard to do right. Encryption and decryption technology is the practice and study of techniques for secure INFORMATION SHARING in the presence of ADVERSARYIES. In encryption and decryption technology, encryption is the process of encoding messages or information in such a way that hackers cannot read it. Encryption and decryption technology is designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. Encryption is the process of converting plain text or information into unintelligible cipher text. Any adversary that can see the cipher text should not know anything about the original message. Decryption is the reverse, in other words, moving from the unintelligible cipher text back to plaintext. The statistical properties of cryptographic methods are the reason for the excellent pseudorandom testability of encryption and decryption technology processor cores. and finally the RM-PRNG key using an Encryption and decryption in as shown fig4.

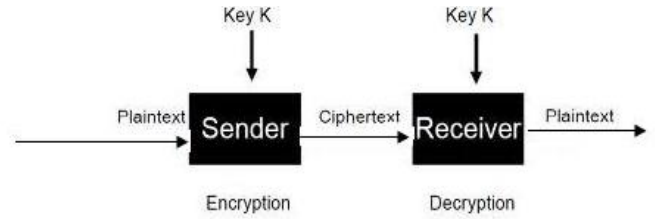


Figure.4: Encryption and Decryption technology

IV. RESULTS AND SIMULATION

Pseudo Random Number Generator, Encryption and Decryption were designed using Verilog language in modelsim 6.4. All the simulations are performed using modelsim 6.4 simulator. The simulated output of Pseudo Random Number Generator, Encryption and Decryption are shown in Figure 5&6 And also FPGA implementation of synthesis using Xilinx10.1 in fig7 and also FPGA result in fig8.

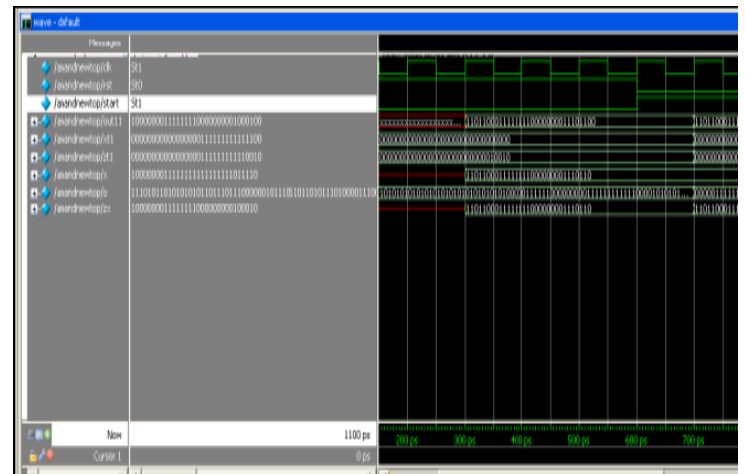


Figure.5: simulation results for RM-PRNG key

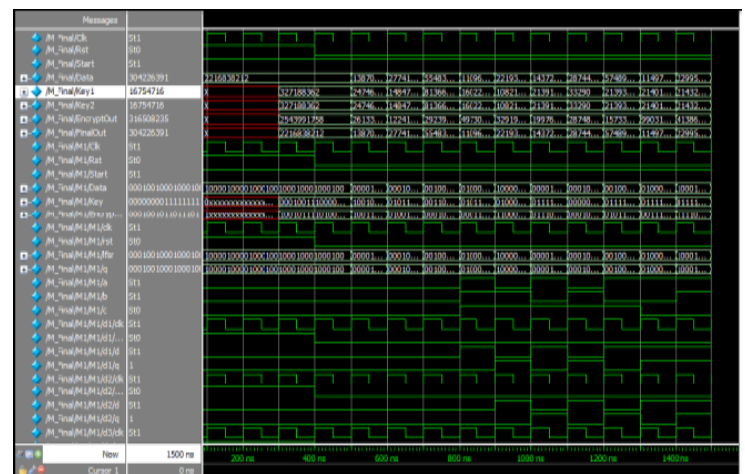


Figure.6: Simulation results of encryption and decryption

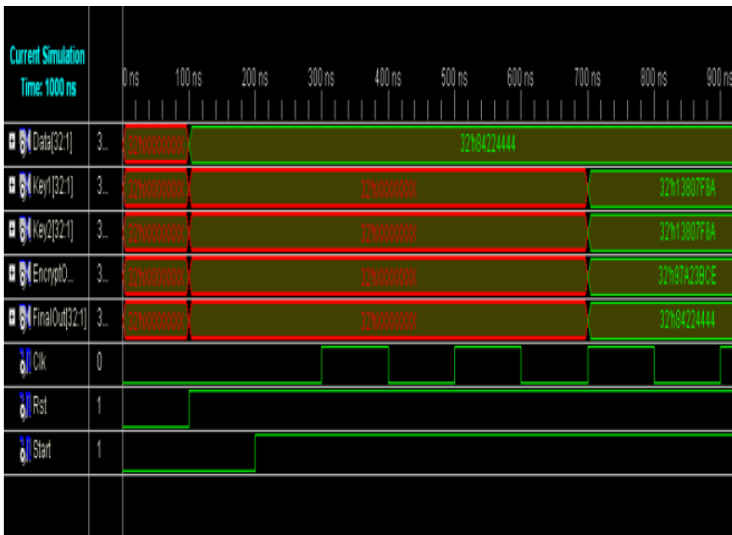


Figure.7: synthesis result

ACKNOWLEDGMENTS

I would like to thank Mr. Nitin Meena, who had been guiding through out to complete the work successfully, and would also like to thank the HOD Mrs. Shwetha Singh, ECE Department and other Professors for extending their help & support in giving technical ideas about the paper and motivating to complete the work effectively & successfully.

REFERENCES

- [1] Chung-Yi Li, Yuan-Ho Chen, Tsin-Yuan Chang, Lih-Yuan Deng and Kiwing To, "Period Extension and Randomness Enhancement Using High-Throughput Reseeding-Mixing PRNG" transactions on (vlsi) systems, vol. 20, no. 2, february 2
- [2] T. Sang, R. Wang, and Y. Yan, "Clock-controlled chaotic keystream generators," Electron. Lett., vol. 34, no. 20, pp. 1932–1934, Oct. 1998.
- [3] D. Mukhopadhyay, "Group properties of non-linear cellular automata," J. Cellular Autom., vol. 5, no. 1, pp. 139–155, Oct. 2009.
- [4] 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.
- [5] J. E. Gentle, Random Number Generation and Monte Carlo Methods, 2nd ed. New York: Springer-Verlag, 2003.
- [6] D. Knuth, The Art of Computer Programming, 2nd ed. Reading, MA: Addison-Wesley, 1981.
- [7] Klapper and M. Goresky, "Feedback shift registers, 2-adic span, and combiners with memory," J. Cryptology, vol. 10, pp. 111–147, 1997.
- [8] 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.
- [9] S. Li, X. Mou, and Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its application in stream-ciphers cryptography," in Progr. Cryptol.-INDOCRYPT, 2001, vol. 2247, pp. 316–329, Lecture Notes Comput. Sci.
- [10] L. Y. Deng and H. Xu, "A system of high-dimensional, efficient, long cycle and portable uniform random number generators," ACM Trans. Model Comput. Simul., vol. 13, no. 4, pp. 299–309, Oct. 1, 2003.
- [11] J. Cermak, "Digital generators of chaos," Phys Lett. A, vol. 214, no.3–4, pp. 151–160, 1996.
- [12] T. Sang, R. Wang, and Y. Yan, "Perturbance-based algorithm to expand cycle length of chaotic key stream," Electron. Lett., vol. 34, no. 9, pp. 873–874, Apr. 1998.
- [13] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudorandom number generator," SIAM J. Comput., vol. 15, pp. 364–383, 1986.
- [14] N. Masuda and K. Aihara, "Cryptosystems with discrete chaotic map," IEEE Trans. Circuits Syst.-I 29, pp. 28–40, 2002.
- [15] J. Fridrich, "Image encryption based on chaotic maps," IEEE Int. Conf. Systems, Man and Cybernetics, vol. 2, pp. 1105–1110, 1997.
- [16] A. Dornbusch, and J. P. Gyvez, "Chaotic generation of PN sequences: AVLSI implementation," IEEE ISCAS 5, pp. 454–457, 1999.
- [17] H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods. Philadelphia, PA: SIAM, 1992.
- [18] S. M. Mathyas and C. H. Meyer, "Generation, distribution and installation of cryptographic keys," IBM Syst. J., vol. 17, no. 2, pp. 126–137, 1978.
- [19] T. Kuusela, "Random number generation using a chaotic circuit," J. NonlinearSci., vol. 3, no. 4, pp. 445–458, 1993.
- [20] A statistical test suite for random and pseudorandom number generators for cryptographic applications, <http://csrc.nist.gov/rng>

AUTHORS

**First Author** – Ms. D. Siva Jyothi pursuing her M. Tech in Embedded Systems and VLSI Design from IES College of Technology, Bhopal, Completed her B. Tech (ECE) from BIT

Project File: lvel Project Status (06/26/2013 17:20:20)					
Project Name:	lvel_Final	Current State:	Programming File Generated		
Target Device:	xc3s500e-5fg320	Errors:	No Errors		
Product Version:	ISE 10.1 - Foundation Simulator	Warnings:	17 Warnings		
Design Goal:	Balanced	Routing Results:	All Signals Completely Routed		
Design Strategy:	Place Default (auto-hack)	Timing Constraints:	All Constraints Met		
		Final Timing Score:	0 (Success Report)		
lvel Partition Summary					
No partition information was found.					
Device Utilization Summary					
Logic Utilization	Used	Available	Utilization		
Number of Slice Flip Flops	733	9,312	7%		
Number of 4 input LUTs	963	9,312	10%		
Logic Distribution					
Number of unoccupied Slices	1,046	4,886	22%		
Number of Slices containing only related logic	1,046	1,046	100%		
Number of Slices containing unrelated logic	0	1,046	0%		
<b>Total Number of 4 input LUTs</b>	<b>1,099</b>	<b>9,312</b>	<b>11%</b>		
Number used as logic	939				
Number used as route-thru	126				
Number used as Shift registers	34				
Number used as Shift registers	24				
Number of bonded I/Os	162	232	69%		
Number of BUFMUXs	1	24	4%		
Performance Summary					
Final Timing Score:	0	Pinout Data:	Pinout Report		
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report		
Timing Constraints:	All Constraints Met				
Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Tue Jun 25 17:13:31 2013	0	17 Warnings	2 Infos
Translation Report	Current	Tue Jun 25 17:19:19 2013	0	0	0
Map Report	Current	Tue Jun 25 17:19:37 2013	0	2 Warnings	2 Infos
Place and Route Report	Current	Tue Jun 25 17:20:03 2013	0	0	2 Infos
Static Timing Report	Current	Tue Jun 25 17:20:12 2013	0	0	3 Infos
Bitgen Report	Current	Tue Jun 25 17:20:26 2013	0	0	0

Figure.8: FPGA Result

V. CONCLUSION

We proposed a Encryption and decryption using RM-PRNG to ensure secure communication. A hardware implementation of RM-PRNG is to offer long periods and high throughput rate to established statistical standards for PRNGs.

The reseeding mechanism solves the short-period problem, while mixing a CB-PRNG with a long-period DX generator extends the period length. Hence, Simulation and Synthesis is observed by ModelSim 6. 4b and Xilinx ISE 10.1

Institute of Technology, Hindupur. Her interested areas of Research include Embedded Systems and VLSI.

**Second Author** – Mr. Nitin Meena working as Assistant Professor in Department of ECE, IES College of Technology,

Bhopal, Completed his M. Tech from Maulana Azad National Institute of Technology, Bhopal. Areas of research interest include VLSI and Embedded Systems.