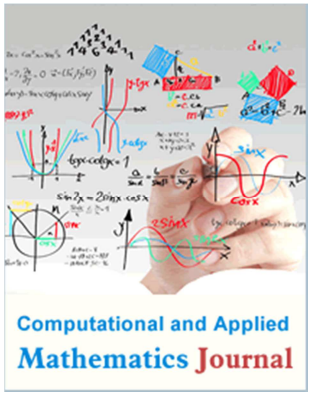
Computational and Applied Mathematics Journal
2015; 1(1): 12-15
Published online January 20, 2015 (http://www.aascit.org/journal/camj)

American Association for Science and Technology



Keywords
Tent Map,
Pseudo-Random Number Generator,
VHDL Implementation

Received: January 08, 2015
Revised: January 15, 2015
Accepted: January 16, 2015

VHDL implementation for a pseudo random number generator based on tent map

Ricardo F. Martinez-Gonzalez[1], J. Alejandro Diaz-Mendez[2], Ruben Vazquez-Medina[3]

[1]Electrics and Electronic Department, Veracruz Institute of Technology, Veracruz, Mexico
[2]Electronics Department, National Institute of Astrophysics, Optics and Electronics, Tonatzintla, Mexico
[3]Culhuacan ESIME, National Polytechnic Institute, Coyoacan, Mexico

Email address
imag@itver.edu.mx (R. F. Martinez-Gonzalez)

Citation
Ricardo F. Martinez-Gonzalez, J. Alejandro Diaz-Mendez, Ruben Vazquez-Medina. VHDL Implementation for a Pseudo Random Number Generator Based on Tent Map. *Computational and Applied Mathematics Journal.* Vol. 1, No. 1, 2015, pp. 12-15.

Abstract
Pseudo random numbers are used for several proposes where uniformity data is desired. For generating them, there are some useful methods, feedback shift registers and dynamical functions are most common methods. Among dynamical functions, one-dimensional chaotic maps are the simplest ones. The tent map is a piecewise-linear one-dimensional map that has been used to generate pseudo random numbers. Its mathematical description originally was delimited from zero to one, so an adjustment was needed for fixing to digital implementation. After adjusting has been made, some dynamical testes were applied to verify if adjusting did not affect its functioning. The next step was to implement the map using VHDL; such implementation was simulated and ratified using a Matlab script. Summarizing, it was achieved a method for easy generation of pseudo random numbers.

1. Introduction

The pseudo random numbers are used for different process. Cryptography strength relies on number randomness [1][2]. In addition, there are other processes where randomness is highly desired; testing and genetic algorithm are good example [3], as any process where data must be uniform either.

Pseudo random numbers can be generated by several methods, but two most widely used are feedback shift registers (FSR) [4] and dynamical system implementations [5]. The FSRs are implemented by one or a set of shifting registers. If they possess a linear condition feedback, then they are called linear FSRs (LFSR) [6]. Otherwise, if their feedback is non-linear, they will be non-linear FSRs (NFSR) [7]; however they are not limited to have only one kind of feedback, and compound structures appear [8].

The dynamical system implementations are more complex than FSRs for classification, because it must include every dynamical system existed or to-be. One part of dynamical systems is n-dimensional maps [9]. The maps describe a dynamical behavior, that can be chaotic or not. Maps have been extensively studied, owing to there are maps with just one variable, also known as one-dimensional map [10]; thus, they are an easy form to produce chaos, and as consequence, randomness.

Even though they are called one-dimensional map, they do not strictly depend on only one variable, they have a feedback control parameter normally designated as μ. This

parameter controls map behavior, transforming a relative stable map into a highly dynamical one.

The maps can be classified according to their mathematical description in polynomial and piecewise-linear. Usually, continuous function one-dimensional maps are classified as polynomial; on the other hand, when its function is divided in two or more linear functions, the map is called piece-wise linear. Probably the most famous polynomial one-dimensional map is the Logistic one [11], basically because it is one of the oldest maps [12] that proved to produce chaos in spite of its simplicity; however, there are several other maps. The Tent map is a good example for piecewise linear map and one of the first developed maps. The tent map will be this work focus.

2. Mathematical Description and Its Digital Adaption for VHDL Usage

According to Hauptmann [13], Tent map mathematical description is delimited from 1 to 0, and it has a feedback factor (μ), what is in charge of control its dynamical behavior. The complete mathematical description appears on next in Eq. 1.

$$x_n = \begin{cases} \mu x_n & 0 < x_n \leq 0.5 \\ \mu(1 - x_n) & 0.5 < x_n < 1 \end{cases} \quad \text{Eq. 1}$$

Eq. 1 needs some modifications to be implemented for a fixed-point digital system; therefore, Eq. 2 was elaborated.

$$x_{n+1} = \begin{cases} \mu x_n & 0 < x_n \leq 2^{bits-1} \\ \mu(2^{bits} - x_n) & 2^{bits-1} < x_n < 2^{bits} \end{cases} \quad \text{Eq. 2}$$

Eq. 2 is the mathematical description used for proposed implementation and every test present from here onward.

In figure 1 appears how digital Tent map behaves, several μ values were taken and it was implemented by a 32 bit system. In comparison with regular Tent map, modified one is quite similar except for its limits. Original map limits are (0, 1) and modified one are (0, 4294967296).

2.1. Dynamical Behavior of Digitalized Tent Map

In order to verify its dynamical behavior, two traditional tools were used, Bifurcation and Entropy diagrams. Bifurcation diagram measures range where possible outputs can be located [14]. And Entropy diagram determinates actual randomness into a certain system for a specific μ [15], its calculation was based on Eq. 3.

$$S = -\sum_{i=1}^{N} P(i) log P(i) \quad \text{Eq. 3}$$

Where, *S* is system Entropy, and *P(i)* is probability of occurrence for certain *i* section.

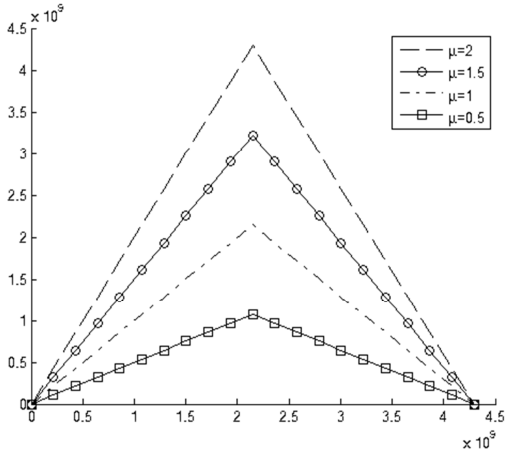


Figure 1. *Behavior of Tent map for different feedback factors using 32 bits.*

Using both diagrams shown at figure 2, it is possible to obtain a good landscape of how random a system is. There are concordances between both diagrams. First one, both diagrams show that Tent map starts its chaotic behavior from μ=1 and tops on μ=2. Other concordance, the higher μ is the spreader output appears to be.

The Entropy diagram was calculated using 256 regions. If every section is equiprobable, best scenario, maximum entropy for system is S=5.5412; and the map obtains its maximum at μ=2, where it has S=5.1982.

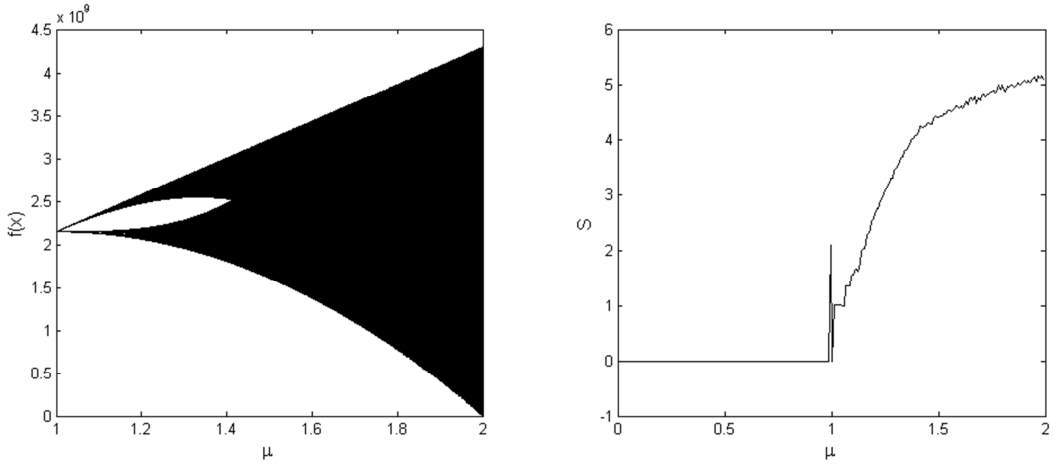


Figure 2. *Bifurcation (left) and Entropy (right) diagram for Tent map.*

2.2. VHDL Implementation

After dynamical verification, the map was implemented using a hardware description language as VHDL. For its implementation, the system was divided in order to obtain a clearer view of its function. The figure 3 depicts blocks that take part of proposed system.

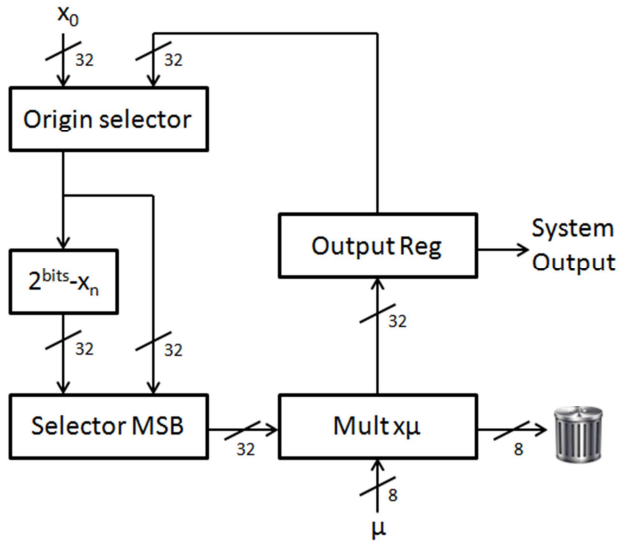


Figure 3. *Block diagram for VHDL implementation*

- Origin selector. It selects what data is introduced to the system, it is important in order to use the initial value for very first time; and thereon, last calculated value.

- 2^{bits}-x_n and Selector MSB. Tent map has two possible outputs according to conditions presented at Eq. 2. One of possible output is $\mu(2^{bits}-x_n)$, and other is μx_n. Both possibilities have as common factor μ; in consequence, it is possible to abstract it and make multiplication later, to reduce complexity. The condition is evaluated using most significant bit (MSB) from x_n, so Selector only let pass correct output.

- Mult xμ and Output register. Here the Selector MSB output is multiplied to complete mathematical description. When two numbers of m and n bits are multiplied, their result is an m*n bits number; thus, eight least significant bits are thrown away and 32 most significant are collected and stored by output register. Output register does data available to be used for another system.

Summarizing, proposed implementation has two inputs, one of 32 bits as initial value, and another one of 8 bits, which works as μ value. Therefore, system output is a 32 bit word.

3. Simulation Results

The VHDL code was simulated using University program VWF from Quartus II Suite. The first simulation was using a 4 bits representation, essentially for quick debugging. 4 bit system had an initial value of 6 and a μ=1.375. The obtained results were satisfied, owing to concurrency between simulation results and calculations; so the next step was taken, simulate using 32 bits for representation. The initial was set on AAAABBBB on hexadecimal what correspond to 2863315899 in decimal; and μ=1.5, whose hexadecimal representation is C0. Figure 4 shown simulations for 4 and 32 bit systems.

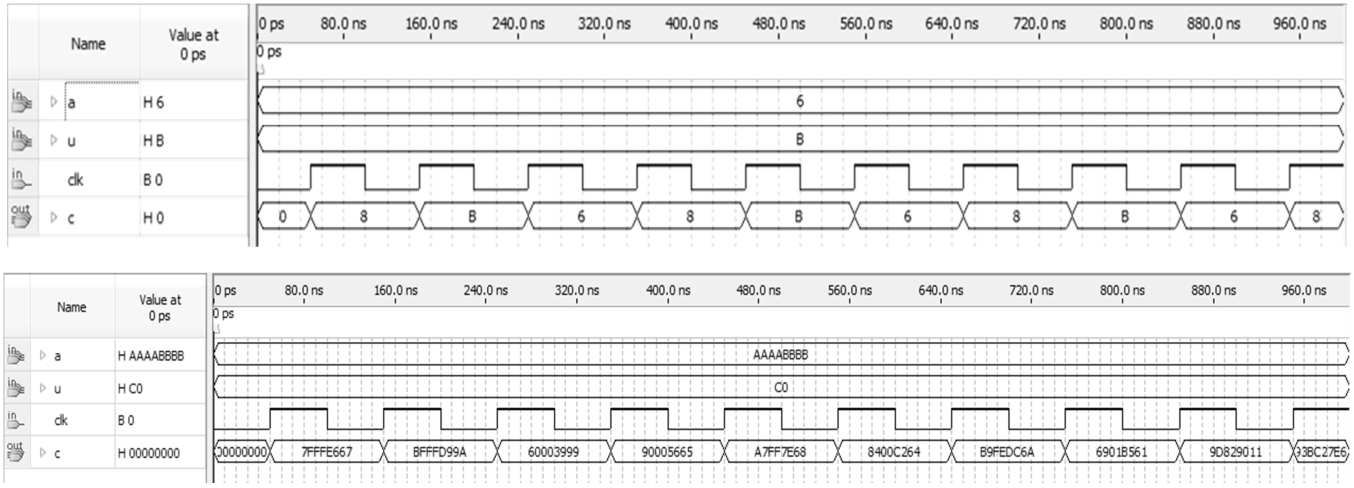


Figure 4. *Simulation of Tent map implemented by a 4 (up) and 32 (down) bit representation system*

Basically because it had simple operations, 4 bit system calculations were handmade; nevertheless, for 32 bits things change and operations were more complex. In consequence, a Matlab script was elaborate. In Table 1 appears a comparison between results obtained from Matlab and simulation.

At both simulations of Figure 4, initial values are signal "a", signal "u" corresponds to μ value and signal "c" is system output. First both system output is 0; in other words, they do not have a valid output. Such behavior prevents malfunctions or possible misunderstandings in other systems that could be connected after the designed pseudo random number generator.

Table 1. *Comparison between Matlab and University program VWF*

Iteration	Matlab (hexadecimal)	Simulation (hexadecimal)
1	7FFFE667	7FFFE667
2	BFFFD99A	BFFFD99A
3	60003999	60003999
4	90005665	90005665
5	A7FF7E68	A7FF7E68
6	8400C264	8400C264
7	B9FEDC6A	B9FEDC6A
8	6901B561	6901B561
9	9D829011	9D829011

4. Conclusions

Tent map proved to be useful to generate pseudo random numbers. Its mathematical description was modified and adjusted in order to simplify digital implementation. Once it was obtained, its description was verified through traditional tools for dynamical systems, as Bifurcation and Entropy diagrams.

The system was designed, analyzed and divided, producing blocks that were used for its function description. Three blocks are part of system: "Origin selector", "2^{bits}-x_n and Selector MSB" and "Mult xμ and Output register". Every block was implemented using a hardware description language (VHDL) and simulated.

Simulation results were compared with calculations, for 4 bit system calculations were handmade, and for 32 bit system calculations were obtained from a Matlab script, exclusively made for such purpose.

Both system simulations were congruent with their respective calculations; such validation gave trust in a well-done implementation. Finally, it is possible to assume that Tent map using a fixed-point representation not only is possible but simple, after making correct adjustments.

References

[1] Susan Hohenberger, Venkata Koppula and Brent Waters, Adaptively Secure Puncturable Pseudorandom Functions in the Standard Model, (2014) IACR Cryptology ePrint Archive, Volume 2014, p 521.

[2] Dennis Hofheinz, Akshay Kamath Venkata Koppula and Brent Waters, Adaptively Secure Constrained Pseudorandom Functions , (2014) IACR Cryptology ePrint Archive, Volume 2014, p720.

[3] Suvajit Dutta, Tanumay Das, Sharad Jash, Debasish Patra and Pranam Paul, A Cryptography Algorithm Using the Operations of Genetic Algorithm & Pseudo Random Sequence Generating Functions, (2014) International Journal of Advances in Computer Science and Technology, Volume 3, No 5, pp 325-330.

[4] Ming Li and Dongdai Lin, A Class of FSRs and Their Adjacency Graphs, (2014) IACR Cryptology ePrint Archive, Volume 2014, p 658.

[5] Ricardo Francisco Martinez-Gonzalez and Jose Alejandro Diaz-Mendez, Implementation of a Stream Cipher Based on Bernoulli's Map, (2014) International Journal of Computer Science & Information Technology, Vol 6 No 6, pp. 113-121.

[6] Guang Zeng, Wenbao Han and Kaicheng He, High Efficiency Feedback Shift Register: sigma-LFSR, (2007) IACR Cryptology ePrint Archive, Volume 2007, p. 114.

[7] Navin Rajpal, Anil Kumar, Sureka Dudhani and Pravesh Raja Jindal, Copyright Protection Using Non-Linear Forward Feedback Shift Register and Error-correction Technique, (2004) 7th Annual International Conference Map India, New Delhi, India.

[8] Yongbin Zhao, Yupu Hu and Shunbo Li, A New Analysis Method for Nonlinear Component of Stream Cipher, (2013) Journal of Information & Computational Science, Vol 10 No 16, pp. 5313-5321.

[9] S. Ramahrishnan, B. Elakkiya, R. Geetha and P Vasuki, Image Encryption Using Chaotic Maps in Hybrid Domain, (2014) International Journal of Communication and Computer Technologies, Volume 2 No 13 Issue 5, pp 44-48.

[10] Chai Wah Wu and Nikolai F. Rulkov, Studying Chaos via 1-D Maps – A Tutorial, (1993) IEEE Transactions on Circuits and Systems: Fundamental, Theory and Applications, Vol 40 No 10, pp 707-721.

[11] Vinod Patinar ans K.K. Sud, A Pseudo Random Bit Generator Based on Chaotic Logistic Map and its Statistical Testing, (2009) Informatica 33, pp. 441-452.

[12] Robert M. May, Simple Mathematical Models with very Complicated Dynamics, (1976) Nature Vol. 261, pp. 459-467.

[13] Hauptmann C., Touchette H. & Mackey M.C., Information Capacity and Pattern Formation in a Tent Map Network Featuring Statistical Periodicity, (2003) Physical Review E 67.

[14] A.Bedri Ozer and Erhan Akin, Tools for detecting chaos. (2005) SAÜ Fen Bilimleri Enstitü sü Dergisi 9.Cilt, 1.Sayõ, pp. 60–66

[15] Robert Lyda and James Hamrock, Using Entropy Analysis to Find Encrypted and Packed Malware. (2007) IEEE Security & Privacy, published by the IEEE Computer Society, pp. 40-45.