Research and Analysis of the Pseudorandom Number Generators Implemented on Cellular Automata

STEPAN BILAN
Department Telecommunication technologies and automatics
State Economy and Technology University of Transport
Lukashevicha str., 19, Kiev, 03049, Ukraine.
UKRAINE
bstepan@ukr.net
MYKOLA BILAN
The municipal educational institution Mayakskaya Secondary School,
Mayak, Moldova of
REPUBLIC OF MOLDOVA
nickni@mail.ru
RUSLAN MOTORNYUK
Data Processing Center of Southwestern Railways, Kyiv, Ukraine
UKRAINE
xehap0@gmail.com
ANDRII BILAN
Business Soft Ltd,
Tiraspol, Moldova of
REPUBLIC OF MOLDOVA
hl181582@gmail.com
SERGII BILAN
Win-Interactive LLC
Vinnytsia, Ukraine.
UKRAINE
belan@svitonline.com

Abstract: - In this paper three pseudorandom number generators are considered which are built on cellular automata. The paper presents the hardware implementation of the generator and it the software simulation. The behavior of the all three random number generators and behavior of the cellular automata were investigated. The first and second generators on aperiodic cellular automata are built, and the third generator on the classic synchronous cellular automaton with inhomogeneous cells is built. There is the analysis of the proposed pseudorandom number generators uses NIST and graphical tests and also their main characteristics described. The most efficient initial settings of the generators have been determined.

Key-Words: - Cellular automata, pseudorandom number generator, tests, cell, neighborhood of cells.

1 Introduction

Formation and obtaining pseudorandom numbers is a necessary operation that is widely used in different fields. Today, the need for the pseudorandom number generators (PRNG) has significantly increased. At the same time there is a need in the PRNG with the certain properties for each problem. These properties satisfy its positive solution. PRNG is widely used in such fields of human activity as: cryptography, game theory, testing of telecommunication systems, protection of telephone lines, simulation etc.

Today there are many random number generators [1–14]. They are implemented on the basis of the different mathematical, hardware and software approaches. The PRNG got the widespread popularity and development, which are implemented on the basis of cellular automata (CA). The first generator that was implemented on the CA has been proposed by S. Wolfram [15-16]. This generator uses a one-dimensional CA. PRNGs, which are implemented on the hybrid CAs were considered in later works [18-22]. The combination of rules for different CA cells are used in such generators, and this gives a pseudorandom sequence of numbers. However, the permanency of their functioning and even with small number of cells leads to the formation of the repetitions of numerical values. At the same time, if it is initially known that the main element is CA, then the rule for each cell can be calculated. Known works are focused on the analysis of the stability of the known generators based on the CA and there are no proposed approaches to improving the quality of the generator.

There were proposed some developed PRNGs, which are based on CA, that are implemented with usage of two CAs and additional generator. Additional generator is built on the linear feedback shift register (LFSR) [23-25]. Using additional LFSR practically is introduced an additional outside source of pseudorandom bits, which deprives the CA of the status of the main defining element.

All PRNGs based on the CA depend on the behavior of the CA itself and also they depend of the particular qualities of their construction. Very little attention is paid in literature sources to the usage of existing methods and means for analysis of the quality and behavior of the PRNG based on CA.

Different approaches are used to analyze the properties of the PRNG. They are implemented as software products that are available in the Internet in appropriate sites [26-29]. They include such software as: ENT, DIEHARD, NIST and etc. DIEHARD and NIST gained the highest popularity. Their use does not let us claim that the bit sequence is random. Instead they allow to determine that the bit pattern is not random. The test picture is composed thanks to these tests, and it allows to judge whether the generator satisfies your requirements or not.

The paper analyzes three PRNGs based on the CA and there are the results of using NIST tests for analysis presented here.

2 PRNG Based on the CA That the Additional Bits Using

The first PRNG is implemented on CA in which only one cell (the active cell) changes its own state at each time step. In such a CA the neighborhood of cells are constitute in such a way so that the one active cell could be selected out of the whole neighborhood of active cell. This cell will be set in active state. Each active cell has one informational output b (data-output) and N active outputs (Figure 1).

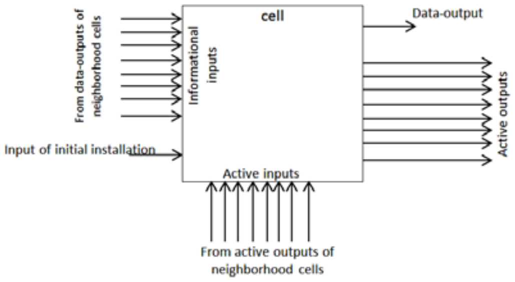

cell
From data-outputs of neighborhood cells
Informational inputs
Input of initial installation
Data-output
Active outputs
Active inputs
From active outputs of neighborhood cells

Figure 1. Scheme of the CA cell for the neighborhood by Moore.

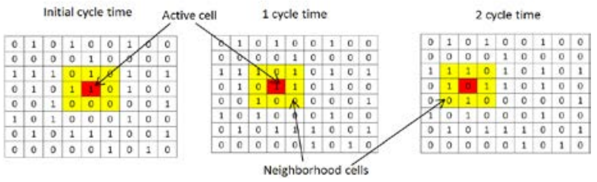
Behavior of CA can be described by the following model.

$$b_i(t+1)=\begin{cases} f\left[b_{N_j}^{i}(t)\right], & if \quad \exists b_{N_j}^{i,act}(t)=1 \\ b_i(t), & in \quad other \quad case \end{cases},$$

where $b_{N_j}^{i}(t)$ - signals on the information outputs of the cells that constitute the cell neighborhood of i-th cell at time t; $b_{N_j}^{i,act}(t)$ - signal at the j-th activation input of i-th cell and this signal comes from activation output of the cell that belongs to the cell neighborhood of i-th cell at time t $\left(j=\overline{1,N}\right)$; N – the amount of neighbor cells, that makes the neighborhood of the i-th cell.

The model shows the process of selecting of the active cell in the next time step. The activating signal can be transmitted only to one cell of this neighborhood. At the same time the state of the cell changes according to the function f[], only on condition that it is activated.

Thus, only one cell changes its state, which is in the state of being activated. Each cell can hold the main (informational) and additional (active) states. In fact, such a CA functions as asynchronous one. The pictorial representation of the active signal transmission and the information state changes are represented on the Figure 2.



0	1	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	0
1	1	1	0	1	0	1	0	1
0	1	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	0	1
0	0	0	0	0	1	0	1	0

0	1	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	0
1	1	1	0	1	0	1	0	1
0	1	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	0	1
0	0	0	0	0	1	0	1	0

0	1	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	0
1	1	1	0	1	0	1	0	1
0	1	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	0	1
0	0	0	0	0	1	0	1	0

Figure 2. The example of active signal transmission and the information state changes.

Bits of the pseudorandom binary sequences are composed from the bits that come out from activated cells outputs in each time step. For elimination of the cycles an additional bit is used which is also taken out of one of the CA cells. The information state of a cell is formed around the Moore's neighborhood by the following model.

$$b_j(t+1)=b_{j,1}(t)\oplus b_{j,2}(t)\oplus\ldots\oplus b_{j,8}(t)\oplus b_{add}(t)\oplus b_j(t),$$

where $b_{j,1}(t)\ldots b_{j,8}(t)$ - the main informational state of cells, that make the Moore neighborhood for the cells with active state at time t; $b_{add}(t)$- informational state of the cell, the value of which is added according to XOR function to the state of the active cell at time t (additional bit value).

Each cell is connected with neighborhood cells by circuits of activation. The activating signal is transmitted by these circuits. The active cell analyzes the information state of neighborhood cells, forms the "1" signal on one of the active outputs, which is defined by a function of being activated and by the signals of neighborhood cells.

The cell of neighborhood is selected on the odd time steps of the generator that is at the state of logic "1" and has bigger code of numbering among the neighborhood cells, which have the state of the log. "1" (Fig. 3). In the even-numbered time step the neighborhood cell of the active cell is selected, it has a state of logic "0" and has bigger value of numbering among the rest neighborhood cells in the zero state. The numbering of the neighborhood cells is shown on Figure 3 for the neighborhood by Moore.

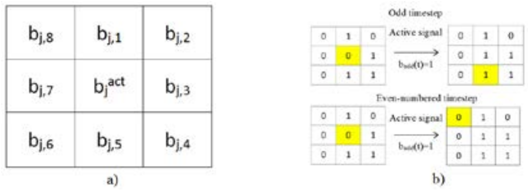


Figure 3. Example of neighborhood cells numbering a) and transmission of the active signal b).

The structure of such PRNG is shown on the Figure 4.

PRNG consists of two CA: the basic and additional (KA$_{add}$) one. There is the initial state of CA stored in CA$_{add}$, which is written by the memory buffer (MB). This state is stored during N×M time steps in the CA$_{add}$ (where N×M - the size of two-dimensional CA). Additional bits in each time step are selected from CA$_{add}$ cells. A signal of "1" on the active output of cell allows to connect its data-output to the output of the generator with the help of switching system (SC). The example of generator operation is represented on Figure 5.

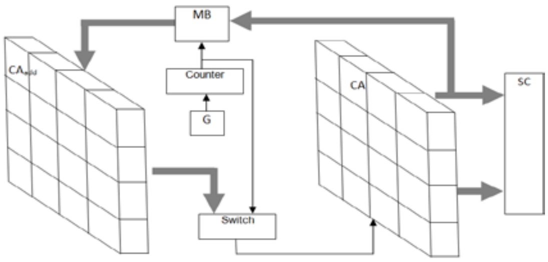


Figure 4. The structure of the PRNG with intermediate storage of arrays.

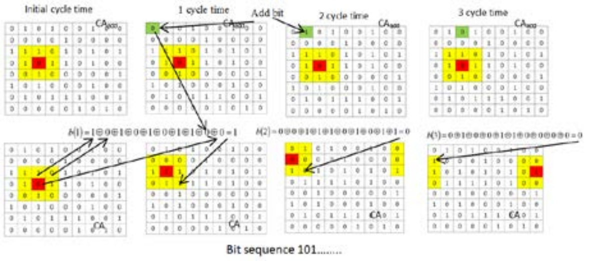


Figure 5. An example of the first generator functioning.

The second PRNG that participated in testing does not memorize the state of CA cells after certain number of time steps. There is no CA$_{add}$ in its structure. It is constructed on the basis of a single CA. The scanning process of the current state of the cells CA is implemented in it. It allows reducing the time wasted for generating pseudorandom sequence. In such a PRNG the cells functioning in the same way. An example of the second PRNG functioning is represented on Figure 6.

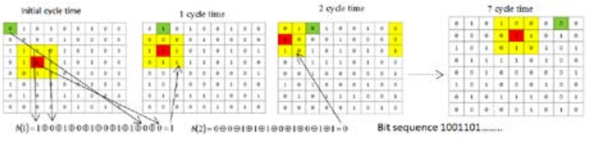


Figure 6. Example of the PRNG with the inner CA scanning.

3 PRNG Based on the CA with Heterogeneous Cells

The generators that are suggested above have a set of structural disadvantages. These generators require a constant additional scanning of CA area for the formation of additional bits. Moreover, the generator uses a complicated switching circuitry for the constant switching the generator output to the data-output of active cells CA. Both generators have a large amount of connections, which deteriorate the reliability of functioning.

PRNG that contains the CA with homogeneous and heterogeneous cells is proposed in this work. This PRNG removes set of disadvantages and also it has the longer repeating period of pseudorandom sequence.

Homogeneous cells are called all CA cells, which perform the same functions, and heterogeneous cells - are the cells that perform functions different from homogeneous cell function. The location and amount of homogeneous cells set is assigned by the PRNG user. The structure of such a PRNG is shown on Figure 7.

The generator works in such a way.

At the initial time all the CA cells are set in the state of the logical "1" and "0", and all of them perform the same function. They are homogeneous cells. There is a chosen cell, which date-output is connected to the output of the PRNG. Then, the few cells are selected among all the cells of CA. These cells will perform another function. These cells are heterogeneous ones.

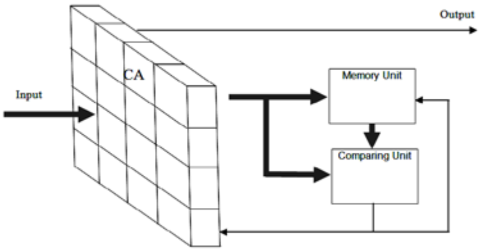


Figure 7. Block diagram of the PRNG based on the CA.

From the very beginning of PRNG work all cells perform the local function. The XOR function is chosen for homogeneous cells, which perform this function above the signals of the neighborhood cells and the signal of its own state. Heterogeneous cells perform the majoritarian function (more than half, of the total quantity of logic "1" states or zero states) of the signals from neighborhood cells and its own state.

At each time step the CA state is recorded to the memory unit (MU), and it is also being compared with all the previous states, which are recorded in the MU.

If the MU meaning does not equal the current state of CA, the comparison unit (CU) produces a signal to the control input MU and allows recording the current state of the CA inside of it.

If the state of CA equals one of the MU code, CU resets the MU down to zero and CA changes the coordinates of the heterogeneous cells. In the next time steps PRNG works in a similar way. The function of calculating new coordinates of heterogeneous cells is selected in advance. Figure 8 depicts the example of PRNG work.

The figure shows the initial state of the CA and also CA states, which are stored in the MU at each time step operation. Formed bits are read from the data-output of the main cell at each time step. This PRNG allows to increase significantly the repeating period, however, it has a low speed.

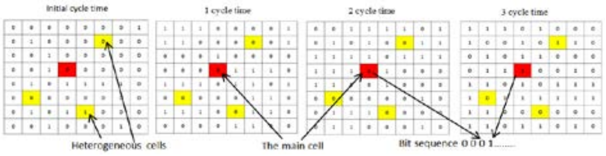


Figure 8. An example of PRNG operation is based on the CA with heterogeneous cells.

4 Analysis of the Quality of the PRNG Work Based on CA

Currently the quality of PRNG is evaluated by means of specially designed tests. These tests make it possible to determine the generator behavior. Due to these tests the set of formed bit consequences of generator is being analyzed. The obtained results help to determine the optimum range of the initial settings that satisfy bigger amount of tests.

Now, almost all the tests are implemented in software. Therefore, for the analyses of hardware generators it is necessary to model them in software way.

Among all the tests the DIEHARD and NIST tests have the greatest authority. To use them, you have to create the files of appropriate format that store the generated bit sequence. ENT and NIST tests were used in this paper for the analysis of the proposed PRNG. These tests are presented at the appropriate sites on the Internet and are available for free usage. ENT tests do not give a complete analysis of PRNG. Therefore these tests were used for the series of bit sequences from each PRNG. They gave positive results for sequences of different length [14]. Positive results on entropy and chi-square were obtained. That is represented in [14].

Because of the ENT and NIST tests are implemented in software, each generator is represented by the programming model. Programming interfaces that implement the model of each generator are shown on Figure 9.

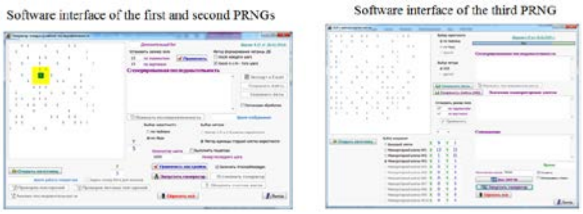

Software interface of the first and second PRNGs
Software interface of the third PRNG

Figure 9. Software interfaces of proposed PRNGs.

There were NIST tests used for the analyses of these generators. And each of the tests was implemented in a separate program. Each program allows selecting the necessary parameters to perform the comprehensive analysis of the PRNG behavior. The results of the analyzing the behavior of all three PRNG are shown in Tables 1, 2.

Table 1. Results of the analysis of the first and second PRNGs (Only the Moore neighborhood).

Name of Test	The Length of the Bit Sequence	Test Result	
		PRNG 1	PRNG 2
The Frequency (Monobit) Test	104, 10^3, 10^4, 10^5, 10^6	+	+
Frequency Test within a Block	104, 10^3, 10^4, 10^5, 10^6	+	+
The Runs Test	104, 10^3, 10^4, 10^5	+	+
	10^6	-	+
Tests for the Longest-Run-of-Ones in a Block	128, 6272,750000	+	+
The Binary Matrix Rank Test	104, 10^3	+	+
	10^4, 10^5, 10^6	-	-
The Discrete Fourier Transform (Spectral) Test	10^3, 10^4, 10^5, 10^6	+	+
The Non-overlapping Template Matching Test	10^3, 10^4, 10^5, 10^6	+	+
The Overlapping Template Matching Test	10^6	-	-
Maurer's "Universal Statistical" Test	387840, 904960	+	+
The Lempel–Ziv Compression Test	10^6	+	+
The Linear Complexity Test	10^6	+	+
The Serial Test	104, 10^3	+	+
	10^4, 10^5, 10^6	-	-
The Approximate Entropy Test	104, 10^3, 104, 10^5	+	+
The Cumulative Sums (Cusums) Test	104, 10^3, 10^4, 10^5, 10^6	+	+
The Random Excursions Test	10^6	+	+
The Random Excursions Variant Test	10^6	+	+

The symbol "+" indicates the positive result, and "-" – the negative one. If «±» is present in the table cell, this means that there were negative results among positive ones but these negative results do not exceed the permissible limits.

The second PRNG has the significantly greater operating speed because it spends the least time on the formation of a single bit of sequence. The third PRNG has poor operating speed performance. After each time step the generator has to carry out a comparison of all the stored arrays of CA states. However, the third PRNG has the greatest length of the repeating period. The experiment showed that if the number of heterogeneous cells is more than 3, then there are not any coincidences of the CA states. Taking into account the given result and removing the feedback, the third PRNG will have the highest speed operation.

Table 2. Results of the analysis of the third PRNG (Only the Neumann neighborhood).

Name of Test	The Length of the Bit Sequence	Test Results for Given Number of Heterogeneous Cells			
		1	2	3	4
The Frequency (Monobit) Test	104 - 10^6	±	+	+	+
Frequency Test within a Block	104 - 10^6	±	+	+	+
The Runs Test	104 - 10^6	±	+	+	+
Tests for the Longest-Run-of-Ones in a Block	128, 6272, 750000	±	+	+	+
The Binary Matrix Rank Test	3800	-	-	-	-
The Discrete Fourier Transform (Spectral) Test	10^4, 10^5	±	+	+	+
The Non-overlapping Template Matching Test	10^4	+	+	+	+
	10^5, 10^6	±	±	+	+
The Overlapping Template Matching Test	10^5, 10^6	-	-	-	-
Maurer's "Universal Statistical" Test	387840, 904960	±	±	+	+
The Lempel–Ziv Compression Test	10^6	±	±	+	+
The Linear Complexity Test	10^6	±	+	+	+
The Serial Test	128	±	+	+	+
	10^3	±	±	+	+
The Approximate Entropy Test	128, 10^3	±	±	+	+
The Cumulative Sums (Cusums) Test	128, 10^3	+	+	+	+
The Random Excursions Test	10^6	±	±	+	+
The Random Excursions Variant Test	10^6	-	±	+	+

ENT and NIST tests allow us to estimate the bit sequence based on obtained statistics. They give a quantitative characteristic, which indicates the presence of any sequence defect. However, they can not determine existence of interactions between elements of the sequence.

For more information about the generator the graphic tests are used. One of these tests is to construct of the histogram of the distribution of elements of the bit sequence. It shows the frequency of occurrence of numbers in sequence. If the frequencies are equal within the predetermined quantitative range, it is considered that the test has given a positive result. The bit sequences by eight bits per byte were divided. Each byte encodes a number, which in the test was estimated. All the three generators have successfully passed the test. An example of the resulting histogram is presented on Figure 10.

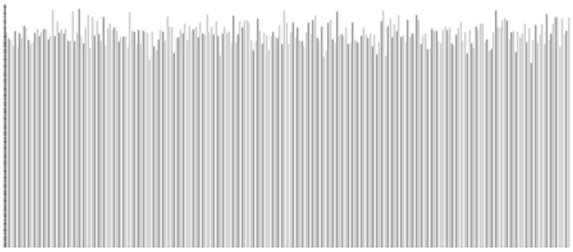


Figure 10. An example of obtained histogram for a bit sequence of the PRNG. The length of the sequence 10^6 bit used.

4 Conclusion

The made PRNG analysis showed that the application of CA allows us to construct the PRNG that can compete with existing generators according to the main characteristics. The analysis of the proposed PRNG based on the CA has given the opportunity to describe their behavior. The test application has shown that the PRNG based on CA has good characteristics, as they have positive results for the majority of tests. Research gave the opportunity to identify that the initial number of single cells and the size of the CA has the main influence on the quality of the PRNG. PRNGs showed the most effective and constant work when the cells were in the logic state “1” and their amount was variable from 35% to 70%. The best CA size is determined from 17 × 17 to 30×30 of cells. By the way, the additional bit is used to avoid of the repeating cycles. This bit was formed by cells of CA itself, which excluded the applying of the external source of random numbers. Experimental research of PRNG based on the CA with heterogeneous cells allowed determining the amount of the initial CA cells with "1" state and size of CA, which give a positive result. Also results allowed to determine the number of heterogeneous cells (more than 3), in which there are almost no repeats. It was found that there is no need in the intermediate comparing. PRNG on the CA has high speed of random sequence generating without intermediate comparison. The proposed generators can be easily implemented on FPGA.

References:

[1] Von Neumann J., Various techniques used in connection with random digits, *Applied Mathematics Series*, Vol. 12, 1951, pp. 36-38.

[2] Bruce Schneier. *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C*, Wiley Computer Publishing, John Wiley & Sons, Inc,. 784, 1996.

[3] L'Ecuyer P.: Uniform random number generation, *Annals of Operations Research*. Vol. 53, 1994, pp. 77-120.

[4] Lehmer D.: Mathematical methods in large-scale computing units, *Large-Scale Digital Calculating Machinery: Symp. proc. Harvard*, 1951, pp. 141-146.

[5] Thomson W., A modified congruence method of generating pseudo-random numbers, *Computer Journal*, Vol. 1, 1958, pp. 83-86.

[6] Hammer P., The mid-square method of generating digits. Monte Carlo Method, *Symp. proc (Los Angeles, 1949). Washington*, Vol. 12, 1951, pp. 33.

[7] Marsaglia G., Random number generators. *Journal of Modern Applied Statistical Methods*, Vol. 2, 2003, pp. 2-13.

[8] Eichenauer J., Lehn J., Topuzoglu A., A nonlinear congruential pseudorandom number generator with power of two modulus, *Mathematics of Computation*, Vol. 51, 1988, pp. 757-759.

[9] Lewis T., Payne W., Generalized feedback shift register pseudorandom number algorithms, *Journal of ACM*, Vol. 21, 1973, pp. 456-468.

[10] Matsumoto M., Nishimura T., Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, 1998, pp. 3-30.

[11] Hell M., Johansson T., Meier W., Grain —A stream cipher for constrained environments, *The eSTREAM Project*, 14, 2005, http://www.

ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.

[12] Ulam S., Random Processes and Transformations, *Procedings Int. Congr. Mathem*, N. 2, 1952, pp. 264-275.

[13] Chugunkov E.V., *Methods and tools to evaluate the quality of pseudo-random sequence generators, focused on solving problems of information security*, Textbook. M.: NEYAU MIFI, 236, 2012.

[14] S. Bilan, M. Bilan, S. Bilan, Novel pseudorandom sequence of numbers generator based cellular automata. *Information Technology and Security*, Vol. 3(1), 2015, pp. 38-50.

[15] Wolfram S., Cellular automata. *Los Alamos Science*, Vol. 9, 1983, pp. 2-21.

[16] Wolfram S., Cryptography with cellular automata. *Lecture Notes in Computer Science*, Vol. 218, 1986, pp. 429-432.

[17] Wolfram S., Random Sequence Generation by Cellular Automata, *Advances in Applied Mathematics*, vol. 7, 1986, pp. 429 – 432.

[18] C. Fraile Ruboi, L., Hernandez Encinas, S. Hoya White, A. Martin del Rey, Rodrigues Sancher., The use of Linear Hybrid Cellular Automata as Pseudorandom bit Generators in Cryptography. *Neural Parallel & Scientific Comp. 12(2)*, 2004, pp. 175-192.

[19] Bruno Martin, Patrick Sole, Pseudo-random Sequences Generated by Cellular Automata, *International Conference on Ralations, Orders and Graphs: Interaction with Computer Scince*, May 2008, Mandia, Tunisia, Nouha editions, 2008, pp. 401-410.

[20] S. Wolfram., Theory and applications of cellular automata, *World Scientific Publishing Co. Ltd*, 1986, pp. 485-557.

[21] K. Cattell, J. C. Muzio, Synthesis of one-dimensional linear hybrid cellular automata, *IEEE Trans. On Computer-aided design of integrated circuits and systems*, 15(3), 1996, pp. 325-335.

[22] S.J. Cho, U. S. Choi, H.D. Kim, Y.H. Hwang, J.G. Kim, S.H. Heo., New syntheesis of one-dimensional 90/150 liner hybrid group CA, *IEEE Transactions on comput-aided design of integrated circuits and systems*, 25(9), 2007, pp. 1720-1724.

[23] Suhinin B.M., High generators of pseudorandom sequences based on cellular automata, *Applied discrete mathematics*, № 2, 2010, pp. 34 – 41.

[24] Suhinin B.M., Development of generators of pseudorandom binary sequences based on cellular automata, *Science and education*, № 9, 2010, pp. 1 – 21.

[25] David H. K. Hoe, Jonathan M. Comer, Juan C. Cerda, Chris D. Martinez, Mukul V. Shirvaikar, Cellular Automata-Based Parallel Random Number Generators Using FPGAs. *International Journal of Reconfigurable Computing*, Vol. 2012, 2012, pp. 1-13, Article ID 219028

[26] By John Walker, *ENT*. A Pseudorandom Number Sequence Test Program.-. January 28th, 2008. http://www.fourmilab.ch/random.

[27] *NIST.- National institute of standarts and technology*, Computer security division. Computer security resource center.- Download Documentation and Software.: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

[28] George Marsaglia, *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness*, Department of statistics and supercomputer computations and research institute.: http://www.stat.fsu.edu/pub/diehard

[29] By Johan Gerard van der Galiën, RABENZIX Randomness Test Suite. 5.4, 2006, Full paper and documentation.: http://members.tele2.nl/galien8/rabenzix/rabenzix.html