Proceedings of the 3rd International Congress APMAS2013, April 24–28, 2013, Antalya, Turkey

On Pseudo Random Bit Generators via Two-Dimensional Hybrid Cellular Automata

F. Temiz[a], I. Siap[a] and H. Akin[b]

[a]Yildiz Technical University, Department of Mathematics, Esenler, Istanbul, Turkey

[b]Zirve University, Department of Mathematics, Gaziantep, Turkey

In this work, we study the structure of two-dimensional linear hybrid cellular automata with respect to adiabatic boundary condition. Further, we check the performance of hybrid cellular automata constructed through the members of this family in generating pseudo random bits.

DOI: 10.12693/APhysPolA.125.534

PACS: 02.10.Yn, 07.05.Kf, 02.10.Ox

1. Introduction

All cryptography schemes are based on producing keys such that they are not predictable. Hence, the key that is random is used in a cryptosystem and producing such keys is one of the main problems in cryptography [1]. Thus, several methods have been proposed in order to accomplish this problem that is to generate pseudo random numbers (PRN) or bits. Generating PRN by using cellular automata (CAs for brevity) is one of these methods. In spite of very good randomness properties of nonlinear CAs [2] ([3]), linear CAs provide a huge convenience due to its algebraic structure. However, it is a well-known fact that linear CA is not good at generating PRNG. Indeed, evolving a nonlinear cellular automaton (singular, CA) requires finding the value of a function with multiple variables for each cell, however with a linear CA since it is a linear transformation, all cells can be evolved at a reasonable time owing to the transition matrix. Without giving up of linearity structure together with its weakness, it is reasonable to use linear hybrid CAs and while applying it instead of choosing blocks we choose a particular cell for this purpose. By this means, we avoid expensiveness of time consuming comparing to nonlinear CA. The key that is generated through this method or any other in general to be in random it has to pass some basic statistical tests: the frequency test, the serial test, the poker test, the run test, and the autocorrelation test [4].

In [5], by using two-dimensional linear hybrid cellular automata (2D) LHCA, generating pseudo random bit sequences has been attempted and the generated bit sequences have been tested whether they are pseudo random or not. Although some very specific examples were given, any classification of the rules was not proposed. Here in this work, we study with respect to another boundary condition which is so-called adiabatic. Moreover, we intend to classify the rules in order to predict which ones or how many of them are suitable to generate pseudo random bit sequences.

2. Cellular automata

CAs are discrete dynamical systems consisting of cells which can exist in k different states. The current state of a cell is determined through its neighboring cell. The evolution of cells or in other words the states of cell are discrete in time. The topological structure of cells can be considered as a lattice. In one dimension this consists of states lined up in a one-dimensional line.

In [4], by using two-dimensional case, we consider a lattice in a plane where we have squares as cells covering the plane. One-dimensional CA can be considered of a lattice network of cells of identical shapes. Each row of this lattice determines the states of cells for a certain time step. The evolution of cells is accomplished with respect to a local transition function that controls how each cell alters. In this paper we study 2D CA. Thus, we define the local transition function and neighborhood for 2D CAs.

2D CAs consist of $m \times n$ cells arranged in m rows and n columns. In this work, we study on finite binary field Z_2 that is each cell can have one of the two states 0 or 1. The local transition function is $(k+1)$-dimensional whose variables are the previous steps of the main cell (will be evolved) and its k neighboring cells [6]: $f : \boldsymbol{Z}_2^{k+1} \to \boldsymbol{Z}_2$. There are two fundamental types of neighborhood: the von Neumann neighborhood and the Moore neighborhood. In this work, we only consider the Moore neighborhood consisting of nine cells array such that a cell is determined dealing with these states. In Fig. 1, one can see the Moore neighborhood of the central cell.

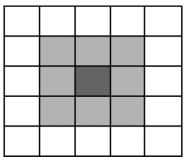

Fig. 1. Moore neighborhood of a cell.

In this respect, we define the $(t+1)$-th time of the (i,j)-th cell as follows [6]:

$$c_{(i,j)}^{(t+1)} = f\Big(c_{(i,j)}^{(t)}, c_{(i+1,j)}^{(t)}, c_{(i+1,j-1)}^{(t)}, c_{(i,j-1)}^{(t)}, c_{(i-1,j-1)}^{(t)}, c_{(i-1,j)}^{(t)}, c_{(i-1,j+1)}^{(t)}, c_{(i,j+1)}^{(t)}, c_{(i+1,j+1)}^{(t)}\Big).$$

However, we only consider the linear rules because they are determined by a matrix. A linear function is of the form

$$c_{(i,j)}^{(t+1)} = ac_{(i,j)}^{(t)} + bc_{(i+1,j)}^{(t)} + cc_{(i+1,j-1)}^{(t)} + dc_{(i,j-1)}^{(t)} + ec_{(i-1,j-1)}^{(t)} + fc_{(i-1,j)}^{(t)} + gc_{(i+1,j-1)}^{(t)} + hc_{(i+1,j)}^{(t)} + kc_{(i+1,j+1)}^{(t)} (\mathrm{mod} 2),$$

where $a, b, c, d, e, f, g, h, k \in \mathbb{Z}_2$. Thus we have $2^9 = 512$ linear rules where we will consider half of them due to computational limits.

64	128	256
32	1	2
16	8	4

shows the fundamental rules of 2D CA. The number within each cell presents the rule for that cell. The number of cells which affect the evolution of the main cell will determine the transition rule number. For instance, the transition rule that determines the next state of the main cell, working on modulo 2 addition of the current states of the cells that are below the main cell, will be $4 + 8 + 16 = 28$.

Since the cellular lattice is finite, we must take into account boundary conditions for the sake of well definiteness of evolution of the CA. Otherwise, extreme cells will be left out which are necessary in order for evolution. The most common boundary conditions are null boundary and periodic boundary. A CA is called null boundary if extreme cells are connected to logic -0 states. A CA is called periodic boundary if the extreme cells are connected to each other. A CA is called adiabatic boundary if extreme cells are duplicated and considered as neighbors.

If every cell of a CA evolves with respect to the same rule, then this type of CA is called uniform. If different cells of the CA evolve with respect to different rules, then the CA is called hybrid CA (HCA).

Now we introduce the transition matrix for 2D CA with adiabatic boundary condition and the block matrices included in it [7]:

$$\begin{pmatrix} 0 & K(d,c,e) & 0_{n\times n} & \dots & \dots & 0_{n\times n} & 0_{n\times n} \\ K(h,k,g) & K(a,b,f) & K(d,c,e) & 0_{n\times n} & \dots & \dots & 0_{n\times n} \\ 0_{n\times n} & K(h,k,g) & \dots & \dots & \dots & \dots & \dots \\ \dots & 0_{n\times n} & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & 0_{n\times n} \\ 0_{n\times n} & \dots & \dots & \dots & \dots & \dots & K(d,c,e) \\ 0_{n\times n} & 0_{n\times n} & \dots & \dots & 0_{n\times n} & K(h,k,g) & 0 \end{pmatrix}, \quad K(x,y,z) = \begin{pmatrix} 0 & y & 0 & . & . & . & 0 & 0 \\ z & x & y & 0 & . & . & . & 0 \\ 0 & z & x & y & 0 & . & . & . \\ . & 0 & z & . & . & . & . & . \\ . & . & 0 & . & . & . & . & . \\ . & . & . & . & . & . & . & 0 \\ 0 & . & . & . & . & . & . & y \\ 0 & 0 & . & . & . & 0 & z & 0 \end{pmatrix}.$$

In this work, we study LHCA which is the combination of two linear 2D CA rules. The cells will be evolved according to two linear rules a and b, respectively, from left to right for each row. The first cell of the next row will use different rule from the last cell of the previous row as expected. Let A be the transition matrix for the rule a, B be the transition matrix for the rule b and H be the transition matrix for the hybrid CA. Therefore, the first row of C will be the first row of A, the second row of C will be the second row of B, and the third row of C will be the third row of A and so on.

3. 2D-LHCA as pseudo random number generators

In this section, we explain how a CA is used to generate pseudo random numbers. Subsequently, we give the statistical test results and argue about their performance.

Pseudo random number generation: in CAs case a pseudo random number generator is obtained by seeding (a short string) the CA to obtain much longer random sequences [1].

In general, it is not easy to guarantee that a generator is a random generator in fact. The generator which is proposed as pseudo random has to pass some statistical tests. Also, these tests are not sufficient but efficient in order to accept that a generator is indeed a random one. The five fundamental tests are usually used to determine if a binary sequence has some special properties that a truly random sequence would be likely to exhibit. We need some definitions before we state those statistical tests. Let s be a bit sequence. A run of s is a subsequence consisting of consecutive 0's or 1's. A run of zeros is called a gap and a run of ones is called a block [4].

1. *Frequency test:* The aim of this test is to determine whether the number of 0's and 1's are evenly distributed or not, as one would expect for a random sequence to enjoy.

2. *Serial test:* The aim of this test is to determine whether the number of the subsequences 00, 01, 10 and 11 are evenly distributed or not, as one would expect for a random sequence to enjoy.

3. *Poker test:* In this test s is divided into k non--overlapping subsequences of length m, where m is an integer such that $\lfloor n/m \rfloor \geq 5 \cdot 2^m$. Let n_i be the number of occurrences of the i-th type of sequence of length m, $1 \leq i \leq 2^m$. The aim of poker test is to determine whether the sequences of length m each

appear approximately the same number of times in s.

4. *Runs test:* The aim of this test is to determine whether the number of runs of various lengths in s is as expected for a random sequence. The expected value of number of gaps (or blocks) of length i in a random sequence of length n is $e_i = (n-i+3)/2^{i+2}$.

5. *Autocorrelation test:* This test aims to check correlations between s and non-cyclic shifted versions of it.

Generating bits via CAs: In [8] two methods are described for generating bits by means of CAs. Firstly, CA can be considered as pseudo random bit generator, quite simply, by concatenating first k configuration, $C^{(0)}, C^{(1)}, \ldots, C^{(k-1)}$ where each configuration consists of n^2 cells. Here, since we work on 2D CA, we can consider our $n \times n$ lattice as a sequence of length n^2. Obviously, this method is not secure for cryptographic purposes: When a part of such sequence is obtained, generating the rest of sequence is immediate if CA used is known.

The second method, which is more secure, is fixing a cell for each configuration and obtaining a sequence by concatenating states of these fixed cells. Hence we will use the second method for our purpose.

In Ref. [7] some CAs have been considered and classified as having chaotic behavior which could encourage the randomness.

4. Interpretation of test results

In this study, we have not tried all possible hybridizations due to computational limits. For each pair of hybrid rules, we have generated 100 bit sequences of length 512 and we have admitted hybrid rules as pseudo random if at least 90 of 100 generated bit sequences have passed all the statistical tests. Table shows test results according to the cell numbers included in the hybridized rule numbers. The first column consists of number of cells used in the hybridization. Second column consists of number of passed (successful) rules among all hybrid rules. After the tests, we observe that there is no rule when hybridized with all others gives a positive result. So we evaluate the results in pairs. The first row of the Table shows us one of the main observations which is that if the hybridized rules consist of only one cell i.e. if they are just fundamental rules, then this hybrid rule does not generate pseudo random bit sequences. Eighth, ninth and tenth rows of Table shows us the weakness of the fundamental rules in order to generate pseudo random bit sequences. We also observe that the rule 2 is the only fundamental rule which is suitable to hybridize with some rules. However, we observe that the rules composed by much more cells are better to hybridize.

For example, if we hybridize the rules containing three cells in their transition rule function with each other's, we consider $C(C(8,3),2) = 1540$ pairs of rules and 80 of them generated 100 bit sequences of length 512 where

TABLE

Test results.

Number of cells included in rule numbers	Number of passed rules	Number of failed rules	Approximate success ratio
1 & 1	0	28	0
2 & 2	0	378	0
3 & 3	80	1460	0.054
4 & 4	504	1911	0.263
5 & 5	642	898	0.714
6 & 6	227	151	1.503
7 & 7	18	10	1.800
1 & 5	14	434	0.032
1 & 6	7	217	0.032
1 & 7	2	62	0.032
2 & 5	129	1439	0.089
2 & 6	91	693	0.131
2 & 7	39	185	0.210
3 & 5	554	2582	0.214
3 & 6	387	1181	0.327
3 & 7	122	326	0.374
4 & 5	1171	2749	0.425
4 & 6	653	1307	0.499
4 & 7	201	359	0.559
5 & 6	685	883	0.775
5 & 7	216	232	0.931
6 & 7	138	86	1.604

at least 90 of these sequences have passed all five statistical tests. This should not be interpreted as only 80 rules composed by three cells are suitable to hybridize. Because, when we hybridize the rules composed by three cells and five cells, we consider $C(8,3) \times C(8,6) = 1568$ pairs of rules and 387 of them achieved to pass all statistical tests.

As another example, while the rules composed by two cells, are not suitable for hybridization with each other's in order to generate pseudo random bit sequences, they can be suitable to hybridize with some rules composed by 5, 6, or 7 cells.

However, we observe that some special rules are not suitable to hybridize with none of the other rules or most of the others. For example, rule 125 is not suitable to hybridize with any of the other rules and it is the only such rule which is composed by six cells. There is no such a rule composed by seven cells. Nevertheless, we can find much more such rules which are composed by three, four or five cells.

In Fig. 2, the weakest ones composed by six, five, four, and three cells are presented. It is observed that if the cells which are not included in the transition rule are separated from the cells included in the transition rule, then it will not be a good rule in order to hybridize.

On the other hand, there are some special rules suitable to hybridize with most of the rules. In Fig. 3, the best ones are obtained by composing six, five, four, and three cells. By contrast with the figure above, the cells in-

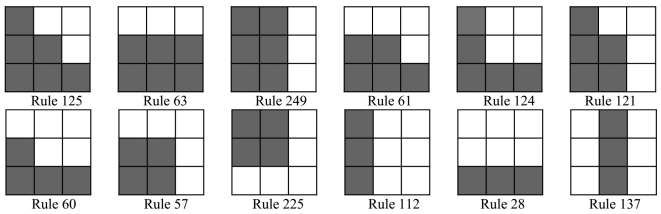

Fig. 2. The weakest CA rules for generating pseudo random bits.

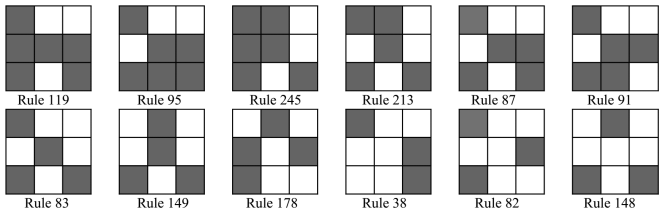

Fig. 3. The best CA rules for generating pseudo random bits.

cluded and non-included in the transition rule are placed randomly.

Acknowledgments

This work is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) (project number: 110T713).

References

[1] D.R. Stinson, *Cryptography Theory and Practice*, CRC Press, Boca Raton 2006, p. 323.

[2] S. Wolfram, in: *Advances in Cryptology: Crypto '85 Proc.*, Ed. H.C. Williams, Springer, New York1986, p. 429.

[3] S. Nandi, B.K. Kar, P.P. Chaudhuri, *IEEE Trans. Comput.* **43**, 1346 (1994).

[4] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton 1997, p. 181.

[5] F. Temiz, I. Siap, H. Akin, M.E. Koroglu, *AWER Procedia* **1**, 1649 (2012).

[6] J.L. Schiff, *A Discrete View of the World*, Wiley, New Jersey 2008, p. 43.

[7] S. Uguz, U. Sahin, H. Akin, I. Siap, *Int. J. Bifurc. Chaos* **24**, 14300 (2014).

[8] C. Fraile Rubio, L. Hernández Encinas, S. Hoya White, A. Martín del Rey, G. Rodríguez Sánchez, *Neural Parallel Sci. Comput.* **12**, 175 (2004).