Avalanche in States; Combination of Sandpile and Cellular Automata for Generate Random Numbers

Seyed Morteza Hosseini[1], Hossein Karimi[2], Majid Vafaei Jahan[3]
[1,2,3] Department of Software Engineering, Mashhad Branch - Islamic Azad University, Mashhad, Iran

Abstract- *Cellular Automata (CA) is a self organizing structure with complex behavior which can be used in pseudo-random numbers generation(PRNG). Pure CA has a simple structure but has no ability to produce long sequences of random numbers. In order to rectify this problem, programmable CA (PCA), using stimulating factor or combination of different self organizing Criticality phenomenon can be used. In this paper, a PCA by using Sandpile model is proposed. The Sandpile is a complex system operating at a critical state between chaos and order. This state is known as Self-Organized Criticality (SOC) and is characterized by displaying scale invariant behavior. In the precise case of the Sandpile Model, by randomly and continuously dropping "sand grains" on top of a two dimensional grid lattice, a power-law relationship between the frequency and size of sand "avalanches" is observed. The avalanche behavior and the pure CA behavior are combined in a novel method which can be used as the pseudo-random number generator.*

Keywords: Random Number Generator, Self-Organizing Criticality, Sandpile Model, Cellular Automata.

1 Introduction

Random number generators (RNGs) play an important role in several computational fields, including Monte Carlo techniques [1], Brownian dynamics [2], stochastic optimization methods [2, 3] and key-based cryptography [4]. It is usual to use mathematical or even evolutionary methods to construct RNGs that yields high quality generators. The quality of generators that determined by statistical tests have a great important role; for example, in cryptography, low quality of RNGs causes easily breaking the encrypted context [4]. In solving optimization problems, as shown in [5], performance and speed of algorithms directly depend on quality of used RNGs. Because of simple structure of CA and its complex behavior and high ability to be used parallel, CA has a well ability in generating random numbers. But one of the major problems of CA is its bounded generated random number sequence because of the self-organizing ability and generate frequent numbers with specific rules. Hence, a strategy for increasing the complexity of behavior and the implementation of CA to present a better random number sequence is required. In this paper, a new method for stimulating CA and increasing the complexity of CA`s behavior based on Sandpile model has been presented. Sandpile model, because of existence of avalanche phenomenon has a non-equilibrium behavior. Hybridizing this model with cellular automata causes a random behavior that it leads to generate a qualified sequence of random numbers.

Herein a two dimensional $n \times m$ CA and combination of 8 rules has been used. The obtained results show that the sequence of generated numbers by CA passed all parts of diehard test suite, entropy and chi-square and other static tests. Some of the other of advantages of this method are uniform distribution of generated numbers by CA, high ability of parallel processing and also the sensitivity to bit changes in particular applications such as cryptography. This paper is organized as following: in following section related works discussed. Section 3, contains the basic concepts of CA and Sandpile model. In section 4 the proposed RNG algorithm and its behavior are discussed. In section 5, the experimental results are illustrated and finally, in section 6, , the conclusion and future works are discussed.

2 Related Works

The first work to apply CA as RNG was done by Wolfram in 1986. His work shows the ability of CA to generate random bits [6, 7]. Basic researches on CA are on producing RNG by one dimensional CA with 3 neighbors [7]. Other researches are focused on increasing CA's complexity with combinations of controllable cells [4, 8] or increasing CA`s complexity with increasing dimensionality. RNG are produced by using one dimensional CA studied in [9, 10, 11, 12, 13] and two dimensional CA in [14, 15, 16] and three dimensional CA in [17]. Hortensius proposed the first non-uniform CA or programmable CA (PCA) by using of the combination of two rules, 90 and 150 in 1989[9]. PCA is a non-uniform CA that allows different rules to be used at different time steps on the same cell. He also represented another generator using the combination of rules 30 and 45 in [10] that its output bits have more dependencies to each other rather than rules 90 and 150.

Recently, extensive studies have been done on PCA for generating random numbers [11, 15, 16, 18, 19]. First works on two dimensional CA represented by Chaudhuri et al. in 1994 [14]. Their results show that produced generator using this CA works better rather than one dimensional CA with the same size. In [20, 21] all 256 (simple) elementary cellular automata were investigated (including those with rules given 90 and 150). It was found that CA with nonlinear rules 45 (or its equivalent rules 75, 89 or 101) exhibit chaotic (or pseudo-random) behaviors similar to those obtained in LFSRs.

3 Cellular Automata and Sandpile Model

3.1 Cellular Automata

A cellular automaton (CA), introduced by Von Neumann in 1940s, is a dynamic system in which its time, space and states are all discrete. The CA evolves deterministically in discrete time steps and each cell takes its value from a finite set S, called the State Set. A CA is named Boolean if $s = \{0,1\}$. The $i-th$ cell is denoted by ‹i› and the state of cell ‹i› at time t is denoted by a_i^t. For each cell ‹i›, called central cell, a symmetric neighborhood of radius r is defined by (1):

$$v_i = \{\langle i-r\rangle, \dots, \langle i\rangle, \dots, \langle i+r\rangle\} \qquad (1)$$

the value of each cell ‹i› is updated by a local transition function f_i-called rule- which for a symmetric neighborhood with radius r is defined as follows (2):

$$a_i^{t+1} = f(a_{i-r}^t, \dots, a_i^t, \dots, a_{i+r}^t) \qquad (2)$$

or equivalently by (3):

$$a_i^{t+1} = f(v_i^t) \qquad (3)$$

Such that v_i^t is as follows (4):

$$v_i^t = f(a_{i-r}^t, \dots, a_i^t, \dots, a_{i+r}^t) \qquad (4)$$

To represent a symmetric rule of radius r for a Boolean CA, a binary string of length L is used, where $L = 2^{2r+1}$, Table 1 Shows the rule 90 of radius one (r=1).

Table 1. The Rule Representation Of Boolean Symmetric Rule 90 Of Radius One

Neighborhood Number	7	6	5	4	3	2	1	0
v_i^t	111	110	101	100	011	010	001	000
$f(v_i^t)$	0	1	0	1	1	0	1	0

If all CA cells obey the same rule, then the CA is said to be a uniform CA; otherwise, it is a non-uniform CA[22]; in addition, a CA is said to be a CA with periodic boundary condition if the extreme cells are adjacent to each other else it called null-boundary CA. If a CA rule involves only XOR logic, it is called a linear rule; rules involving XNOR logic are referred to complemented rules. A CA with all cells having linear rules is called linear CA, whereas a CA having a combination of linear and complemented rules is called an additive CA [23]. Nandi et al. presented a programmable CA (PCA) in 1994 [23]. A CA is said to be a PCA if it uses a control CA to determine the rules of each cell. A control CA is essentially just another basic CA which is usually of uniform nature. The rule function used by each cell changes with time and is decided by the control CA. PCA is, in fact, a non-uniform CA because all its cells collectively use different rule functions. A PCA may use m-bit control CA, where $m \geq 1$. For each cell, there are 2^m rules to choose from, thereby, allowing less probability of correlations among the cells. Compared to uniform CA, PCA allows several control lines per cell. Through these control lines, different rules can be applied to the same cell at different time steps according to the rule control signals. Fig. 1, shows a PCA cell structure.

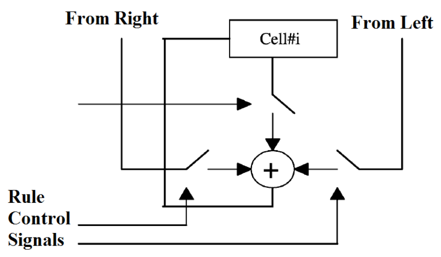


Fig. 1. A PCA cell structure

As Illustrated in Fig. 1, control signals select cell's rule. In this paper a two dimensional Sandpile model and two dimensional PCA with non-periodic boundary condition is considered. Each PCA cell's state can be a number as 0, 1, 2, 3. Herein, applied rules are the same rules that were used in elementary CA.

3.2 Sandpile Model

In 1987, Bak at al. [24] identified the SOC phenomenon associated with dynamical systems. The first system were SOC was observed was named after its inspiration as the Sandpile model, and consists of a cellular automata where at each cell of the lattice, there is a value which corresponds to the slope of the pile. Grains of sand are randomly “thrown” into the lattice where they pile up and increment the values of the cells. When a value exceeds a specific threshold, an avalanche takes place and four grains belonging to that cell are distributed by the neighboring sites (*von Neumann* neighborhood). If one of those sites also exceeds the threshold value(z_c), the avalanche continues, and the grains are also sent to the adjacent cells. The procedure of the Sandpile model is shown in fig. 2.

With these settings, and depending on the state of the lattice and the position of the new grain, a grain may cause rather different responses. It may not cause any change in the system if it falls in a cell with its value bellow threshold (other than increasing the sand on the cell, of course) and it may generate large avalanches of sand that will strongly redefine the shape of the pile.

```
Procedure Sandpile
Considerer a lattice (x,y) and a function z(x,y) which represents the number o grains in the cells.
Starting with a flat surface $z(x,y) = 0$ for all $x$ and $y$:
Add a grain of sand: $z(x,y) = z(x,y) + 1$
 if $z(x,y) > z_c$ then an avalanche occurs
          $z(x,y) = z(x,y) - z_c$
          $z(x \pm 1, y) = z(x \pm 1, y) + 1$
          $z(x, y \pm 1) = z(x, y \pm 1) + 1$
if $z(x, y \pm 1) = 4$ or $z(x \pm 1, y) = 4$
Update z recursively
```

Fig. 2. 2D Bak-Tang-Wisenfeld Sandpile Model

4 Proposed RNG Based On Combination of Sandpile and PCA

4.1 Proposed RNG

In this scheme a two dimensional $n \times m$ PCA with Null boundary condition is used to generate random numbers by using 8 rules: 153, 30, 90, 165, 86, 105, 110, 150. According to [25], generated numbers by these rules have the best results in different tests such as entropy, chi-square and diehard. The Boolean expression of each CA rule is shown in Table 2.

Table 2: The detail and Boolean expression of each CA Rule

Rule Name	**Possible Input Configuration**								**Boolean Representation**
	111	**110**	**101**	**100**	**011**	**010**	**001**	**000**	
101	0	1	1	0	0	1	0	1	[x_{i-1} *nor* x_{i+1}] *or* [(x_i *xor* x_{i+1}) *and* x_{i-1}]
105	0	1	1	0	1	0	0	1	*Not[*x_{i-1} *xor* x_i *xor* x_{i+1}]
86	0	1	0	1	0	1	1	0	[x_{i-1} *nor* x_i] *xor* [*not*(x_{i+1})]
165	1	0	1	0	0	1	0	1	[x_{i-1}] *xnor* [x_{i+1}]
90	0	1	0	1	1	0	1	0	[x_{i-1}] *xor* [x_{i+1}]
30	0	0	0	1	1	1	1	0	[x_{i-1}] *xor* [x_i *or* x_{i+1}]
153	1	0	0	1	1	0	0	1	[x_i] *xnor* [x_{i+1}]
150	1	0	0	1	0	1	1	0	[x_{i-1}] *xor* [x_i] *xor* [x_{i+1}]

In this paper, for Sandpile implementation, the threshold value is considered equal to 4 and each cell has four nearest neighbors: up, down, left and right (Von-Neuman neighborhood). The value of each CA`s cell is an integer between 0 and 3. For converting these numbers to 0 and 1(binary state), their residual over 2 is used. Hence, numbers 0 and 2 (even numbers) are delegated to 0 and numbers 1 and 3 (odd numbers) are delegated to 1. In order to generate random numbers, the CA was initialized by random numbers between 0 and 3. At each time step, there are two steps that are discussed in the following:

Firstly, a $n \times m$ CA is set to the binary state and each row divides into 4-cell's parts (each part has four cells). In each word the first two cells show the number of time that Sandpile run on each cell (φ) and the second two cells show the cell that action should be run on it (α). Each word (φ, α), is extracted from the word in the previous row. Because of increasing φ has no tangible effect on the quality of the generated random numbers and only increases processing time, φ is restricted to the maximum equal to 2. If the first two cell of each word is equal to 0, 1 or 2 the Sandpile action is run once; else if it is equal to 3 the Sandpile action is run twice. Fig. 3, show the hardware schema process of way of selection for performing sandpile action and the number of sandpile actions for a 4 cell section in row i+1.

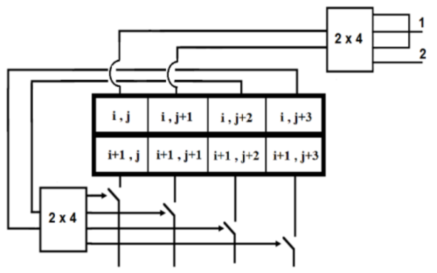


Fig. 3. Proposed CCA with selection of a cell for running Sandpile action.

For example, Fig. 4(a), shows the three rows of CA; and Fig. 4(b), shows its binary state. These figures show the number of Sandpile runs and the cells that Sandpile applied on them.

	0	1	2	3	4	5	6	7
0	3	1	1	2	3	0	2	0
1	1	2	3	0	1	0	2	3
2	2	2	1	0	0	3	1	3

(a) three rows of CA

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

(b) binary state of three rows

Fig. 4. An example of CA and its binary state

For determination of (φ, α), in the first word of the second row, the corresponding data in the previous row i.e. first row was used. The value of the first two bits of the first word of first row is $(11)_2$ that imply number 3. So as mentioned, the number of Sandpile run in first word of the second row would be equal to 3. The value of the second two bits of the first word of the first row is $(10)_2$ that shows cell 2 is selected. So Sandpile is run twice on cell 2 of the first word of second row i.e. the cell [1, 2]. Because the considered CA is periodic, previous row of the first row is the last row (seventh row). It is repeated for second word of second row similarly. The second word of the first row, which determine the number of Sandpile run is $(10)_2 = 2$ and the next word which determine the cell that the Sandpile applied it was $(00)_2 = 0$, Thus on the zero cell of the second word of the second row i.e. cell [1, 4], the Sandpile was performed once. For all rows, these data inferred synchronically.

In the second step, the CA has been updated by the eight mentioned rules. This step comprised three parts. In the first part, a rule for each cell, according to the Table 3 synchronically determined. The number of rules which used in this paper is specified. To determine a rule for cell [i, j], the procedure is shown as following: for cells [i-1, j-1], [i-1, j], [i-1, j+1] and [i+1, j-1], [i+1, j], [i+1, j+1], the XOR operator was used on them correspondingly (i.e. the XOR operator applied on cells [i-1,j-1] and [i+1,j-1] and so forth) and generate an integer between 0 and 7. Fig. 5, shows the PCA Structure and hardware presentation of determined rules 90/ 150/ 165/ 105/ 101/ 86/ 30/153 for cell [i,j].

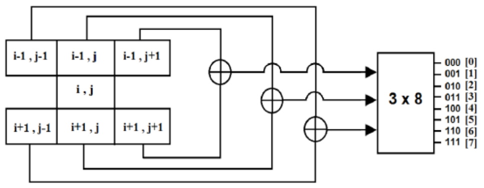


Fig. 5. Proposed PCA with Determined rules for cell [i,j]

The obtained number is the rule number that must be applied on the cell [i, j]. For instance, if after performing Sandpile action, the values of CA will be Fig. 4(b), the number of determined rule for cell [1, 1] is equal to $111 \oplus 001 = 110$, i.e. which is the 6th rule. In other words, according to Table 3 in performing rule section on CA, the rule which should be applied on the cell [1, 1] is 153.

Table 3. CA rules lookup Table

0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
101	105	86	165	90	30	153	150

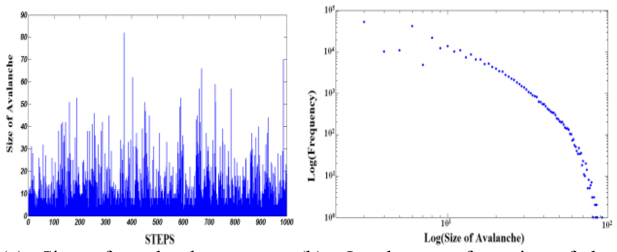
In the second part, determined rules were applied. Using rules also is synchronously and the rules would be applied with respect to the rows. For example, in Fig. 4(b), the result after performing rule 153 on cell [1, 1] is equal to $Rule_{153}(101) = 0$. In the third part, for updating the CA, the value of each CA cell should be added to the value that obtained from the used rule on that cell, which will be 0 or 1, and since all values must be an integer between 0 and 3, their integer residual of them by 4 were calculated.

4.2 How Use of Sandpile Model Results in Random State in Cellular Automata?

Role of sandpile model in this model is to actuate and produce the maximum disturbance in cellular automata for preventing from cycle formation and reaching the maximum entropy in cellular automata. As it was stated, The Sandpile is a complex system operating at a critical state between chaos and order and a power-law relationship between the frequency and size of sand “avalanches” is observed .In a system exhibiting critical behavior, A small perturbation in one given location of the system may generate a small effect on its neighbourhoods or a chain reaction that affects all the constituents of the system. The statistical distributions describing the response of the system exhibiting SOC are given by power laws in the form

$$P(s) \sim s^{-\tau} \qquad (5)$$

where s is the number of constituents of the system affected by the perturbation, d is the duration of the chain reaction(lifetime), and τ are constants. Large avalanches are very rare while small ones appear very often. Without any fine-tuning of parameters, the system evolves to a non-equilibrium critical state. Fig. 6 shows a distribution of avalanches created by our sandpile model with a dimension of 12 $\times$ 12, which has been running for 100000 steps.



(a) Size of avalanches over time (steps); right: Log-log

(b) Log-log transformation of the size of avalanches in relation to their frequency of occurence

Fig. 6. Power law number output of the sandpile model.

As it is shown in figure 6, behavior of applied sandpile model in the proposed generator follows power law and the number of avalanche occurrences is inversely proportional with the size of avalanche. The average occurred state change in cellular automata was measured equal to %47.83 after performing sandpile on rows.

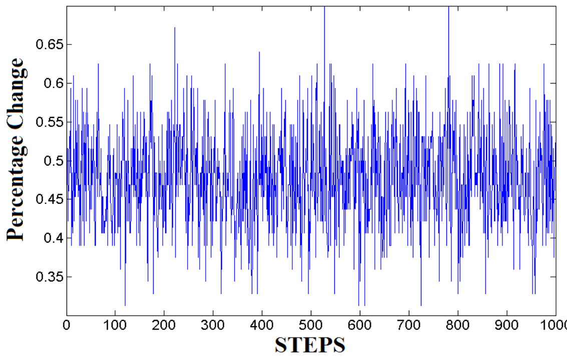


Fig. 7. Percentage Changes in Cells After Applying Sandpile

Fig. 7 shows percentage changes in cells after 1000 times sandpile performance. Combining this model with cellular automata, in addition to disturbing cells state, causes a severe mutation in values of cellular automata because of creating huge avalanches and prevents from short period length sequences and leads to the maximum entropy in cells values. Average of Percentage changes after performing rules of cellular automata and performing sandpile action on cells is %50.1.

5 Experimental Results

For analyze the proposed generator, the generated bits sequence divided into 4-bit parts and so, different tests such as entropy, chi-square and the changes sensitivity and other tests on the obtained numbers were assessed which are between 0 and 15. To perform all tests which will be presented in following sections, an 8×8 cellular automata is applied with random initial values.

5.1 Several Basic Statistical Tests For PRNG

Let s $= s_0, s_1, s_2, \dots,\ s_{n-1}$ be a binary sequence of length n. This subsection presents several basic statistical tests that are commonly used for determining whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to exhibit. It is emphasized again that the outcome of each test is not definite, but rather probabilistic.

5.1.1 Frequency Test (Monobit Test)

The purpose of this test is to determine whether the number of 0's and 1's in s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of 0's and 1's in s, respectively. The statistic used is:

$$x1 = \frac{(n_0 - n_1)^2}{n} \tag{6}$$

which approximately follows $a\, x^2$distribution with 1 degree of freedom if $n \geq 10$. For a significance level of α = 0.05, the threshold values for this test is 3.8415 [26].

5.1.2 Serial Test (Two-Bit Test)

The purpose of this test is to determine whether the number of occurrences of $00, 01, 10,$ and 11 as subsequences of s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of 0's and 1's in s, respectively, and let $n00, n01, n10, n11$ denote the number of occurrences of $00, 01, 10, 11$ in s, respectively. Note that $n00 + n01 + n10 + n11 = (n - 1)$ since the subsequences are allowed to overlap. The statistic used is:

$$\frac{4}{n-1}(n00^2 + n01^2 + n10^2 + n11^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1 \tag{7}$$

which approximately follows $a\, x^2$distribution with 2 degrees of freedom if $n \geq 21$. For a significance level of α = 0.05, the threshold values for this test is 5.9915 [26].

5.1.3 Poker Test

Let m be a positive integer such that $\left\lfloor \frac{n}{m} \right\rfloor \geq 5.(2^m)$ and let $k = \left\lfloor \frac{n}{m} \right\rfloor$. Divide the sequence s into k non-overlapping parts each of length m, and let n_i be the number of occurrences of the i^{th} type of sequence of length $m, 1 \leq i \leq 2m$. The poker test determines whether the sequences of length m each appear approximately the same number of times in s, as would be expected for a random sequence. The statistic used is:

$$x3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k \tag{8}$$

Which approximately follows $a\, x^2$ distribution with $2^m - 1$degrees of freedom. Note that the poker test is a generalization of the frequency test: setting m = 1in the poker test yields the frequency test. . For a significance level of α = 0.05, the threshold values for this test is 14.0671 [26].

5.1.4 Autocorrelation Test

The purpose of this test is to check for correlations between the sequence s and (non-cyclic) shifted versions of it. Let d be a fixed integer, $1 \leq d \leq \lfloor n/2 \rfloor$. The number of bits in s not equal to their d-shifts is $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$ where ⊕ denotes the XOR operator. The statistic used is:

$$x_5 = 2\left(A(d) - \frac{n-d}{2}\right)/\sqrt{n-d} \tag{9}$$

Which approximately follows an $N(0;\,1)$ distribution if $n - d \geq 10$. Since small values of $A(d)$ are as unexpected as large values of $A(d)$, a two-sided test should be used. . For a significance level of α = 0.05, the threshold values for this test is 1.96 [26]. In Table 4, values of discussed tests are presented for the proposed generator. For this, a sequence of random numbers is generated with 1 million bits and discussed tests are implemented on it. This procedure is repeated 100 times and its average is given, too..

Table 4. Values of 4 basic statistical test

TESTS				Pass
Frequency	**Serial**	**Poker**	**Autocorrelation**	
0.459	2.533	8.851	0.312	4/4

As it is shown in Table 4, generator is able to pass all tests.

5.2 ENT Test

The ENT test is useful for evaluating pseudorandom number generators for encryption and statistical sampling applications, compression algorithms, and other applications where the information density of a file is of interest [27]. The ENT test is a collective term for three tests, known as the Entropy test, Chi-square test, and Serial correlation coefficient (SCC) test. Table 5 shows values of this test for the proposed generator. In entropy test, its maximum value is 4 and Chi- Square test with freedom degree 4 and precision of 0.1 is used. For doing these tests, a sequence of length 2^{17} is used.

Table 5. ENT Test

TESTS		
Entropy	**Chi-Square**	**SCC**
3.9999	3.0185	0.00008

As it is represented in above table, generated sequence is able to pass all tests successfully and with good result.

5.3 PRNG Quality Evaluation

To compare how our PRNG performs against several different PRNGs, we use Diehard test suite [28]. For this reason, the proposed generator based on obtained score from DIEHRD test is compared with other generators. we used Johnson's scoring scheme [29]: we initialized (*a*0, *a*1, *a*2, *a*3, *a*4, *a*5, *a*6, *a*7) with 32 different random values obtained from http://randomnumber.org, got 32 different 10MB files, and then assigned scores based on the results of the Diehard tests. The PRNGs we have compared to ours are of several different kinds: Linear Congruential Generators (rand [30], rand1k [31], pm [32]), Multiply with Carry Generators (mother [33]), Additive and Subtractive Generators (add [30], sub [32]), Compound Generators (shsub [30], shpm [32], shlec [32]), Feedback Shift Register Generators (tgfsr [34], fsr [35]), and Tausworthe Generators (tauss [36]).

Each of the Diehard tests produces one or more p-values. We categorize them as good, suspect or rejected. We classify a p-value as rejected if $p \geq 0.998$, and as suspect if $0.95 \leq p < 0.998$; all other p-values are considered to be good. We assign two points for every rejection, one point for every suspect classification, and no points for the rest. Finally, we add up these points to produce a global Diehard score for each PRNG, and compute the average over the 32 evaluations: low scores indicate good PRNG quality. The information relating to the different PRNGs was taken from [31, 37]. The results are presented in Table 6. We note that our PRNG is outstandingly better than the rest of the analyzed PRNGs: the lowest scores correspond to shsub (17.125) and fsr (17.90625), significantly greater than our PRNG (12.718750). On the other hand, the average scores increase up to 50.59375 (pm), 66.53125 (rand), and even 291.78125 (rand1k).

Table 6. PRNG Diehard Scores

PRNG	Total Score	Mean
Rand	2129	66.531250
rand1k	9337	291.78125
Pm	1619	50.593750
Mother	602	18.812500
Add	577	18.031250
Sub	655	20.468750
Shsub	548	17.125000
shpm	799	24.968750
shlec	751	23.468750
fsr	573	17.906250
tgfsr	584	18.250000
tauss	935	29.218750
Proposed PRNG	407	12.718750

5.4 Avalanche Effect

Bit change sensitivity analysis is used to analyze the RNGs that are used in cryptography. One of the desired properties in each cryptography algorithm is that a small change in plaintext or key yields salient changes in ciphertext. In special case, changing one bit in key or plaintext should change in half of the ciphertext. This property is known as avalanche and represented by Fiestel in 1973 [16].

As mentioned before, the proposed generator could be used to generate key in cryptography. To generate a unique key in both encoding and decoding, the initial values must be available. Thus, for high security in cryptography, generated bits stream must have too much dependency to this parameter. As mentioned before, a 8×8 CA with an integer between 0 and 3 as the value of each cell was used. Then two bits are needed to determine the value of each cell and 128 bits could determine the value of cells. Two analysis of this section, 128 bits randomly generated to determine the initial state and the generated bits sequence generated from one cell. Then one of these 128 bits has been inverted and the sequence of generated bits with the new initial state of the same cell generated. At last, the both sequences have been compared with each other. Fig. 8, shows the changes percent of generated sequences from one specific cell with two initial states that differ only in the ith bit (horizontal axis).

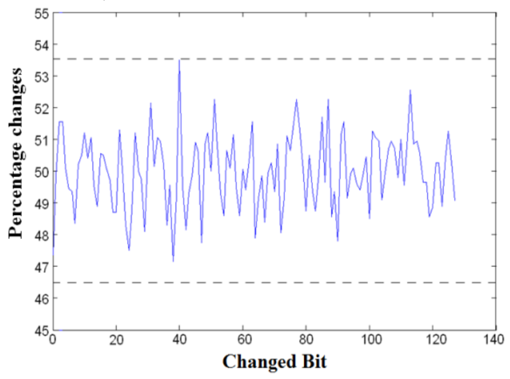


Fig. 8. changes percentage between generated data from two keys that only differ in one bit

6 Conclusion

In this paper, a new PCA for generating random numbers using CA and Sandpile model presented. This method, have the high performance in all tests and could be used in cryptography. Because of avalanche and self organizing properties of Sandpile model, it has a complex behavior and could be used as a convenient factor in stimulating of CA to generate a high quality sequence of random numbers. In each step, first the Sandpile process applied on the four neighbors of each cell of the two dimensional automata and then the CA has been updated by the synthetic of 8 rules 165, 105, 90, 150, 153, 101, 30, 86. The results of applied tests on the generated numbers show that this generator has the maximum entropy and since passing the chi-square and diehard tests. This generator also has a convenient speed and holds the ability of parallelism of CA.

Reference

[1] J. Gentle, "Random number generation and Monte Carlo methods," *Springer New York* 2003, 2th edition, 2004, ISBN-10: 0387001786
[2] A. Reese, "Random number generators in genetic algorithms for unconstrained and constrained optimization," *Nonlinear Analysis: Theory, Methods & Applications*, Vol. 71, pp: 679 - 692, 2009.
[3] P.L.Ecuyer, "Random numbers for simulation, " *Communications of the ACM*, Vol. 33, pp:85-97, 1990.
[4] M. Tomassini, M. Sipper, and M. Perrenoud, "On the generation of high quality random numbers by two-dimensional Cellular Automata," *IEEE Transactions on Computers*, Vol. 49, pp: 1146 –1151, 2000.

[5] J.Carmelo, A. Bastos-Filho, D. Jaulio, D. Andrade, R. Marcelo, S. Pita, D. Ramos, "Impact of the Quality of Random Numbers Generators on the Performance of Particle Swarm Optimization," *IEEE International Conference on Systems, Man and Cybernetics* , pp: 4988–4993, 2009.

[6] S. Wolfram, "Cryptography with cellular automata," in *Proc. CRTPTO 85 Advances in Cryptography*, Vol. 218, pp: 429–432, 1985.

[7] S. Wolfram, "Theory and Applications of Cellular Automata," *River Edge, NJ: World Scientific*, pp:1983–1986, 1986.

[8] S.-U. Guan and S. Zhang, "A family of controllable cellular automata for pseudorandom number generation," *International Journal of Modern Physics C*, Vol. 13, Issue 8, pp:1047-1073 2002.

[9] P. D. Hortensius, R. D. Mcleod, and H. C. Card, "Parallel random number generation for VLSI system using cellular automata," *IEEE Transactions on Computers*, Vol. 38, pp: 1466–1473, 1989.

[10] P. D. Hortensius, R. D. Mcleod,W. Pries, D. M. Miller, and H. C. Card, "Cellular automata-based pseudorandom number generators for built-in self-test," *IEEE Transactions on Computers*, Vol. 8, pp: 842–859, 1989.

[11] P. Anghelescu "Encryption Algorithm using Programmable Cellular Automata", World Congress on Internet Security (WorldCIS), pp: 233 – 239, 2011

[12] S.H. Shin, K.Y. Yoo, "Analysis of 2-State, 3-Neighborhood Cellular Automata Rules for Cryptographic Pseudorandom Number Generation
",International Conference on Computational Science and Engineering, CSE '09, pp: 399 – 404,2009.

[13] X.Xuewen, L.Yuanxiang, X.Zhuliang, W.Rong, "Data Encryption Based on Multi-Granularity Reversible Cellular Automata", International Conference on Computational Intelligence and Security, 2009. CIS '09, pp: 192 – 196, 2009.

[14] D. R. Chowdhury, I. S. Gupta, and P. P. Chaudhuri, "A class of two-dimensional cellular automata and applications in random pattern testing," *Journal of Electronic Testing: Theory and Applications*, Vol. 5, pp: 65–80, 1994.

[15] B.H.Kang, D.H.Lee, C.P.Hong, "High-Performance Pseudorandom Number Generator Using Two-Dimensional Cellular Automata", 4th IEEE International Symposium on Electronic Design, Test and Applications, pp:597 - 602, 2008

[16] B.H.Kang, D.H.Lee, C.P.Hong, "Pseudorandom Number Generation Using Cellular Automata", Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics, pp:401-404, 2008.

[17] S.H.Shin, G.D.Park, K.Y.Yoo, "A Virtual Three-Dimension Cellular Automata Pseudorandom Number Generator Based on the Moore Neighborhood Method", 4th International Conference on Intelligent Computing, ICIC 2008, pp: 174-181, 2008.

[18] A. Ray, D. Das, "Encryption Algorithm for Block Ciphers Based on Programmable Cellular Automata", Information Processing and Management, Vol.70, pp:269-275, 2010.

[19] N.S.Maiti, S.Ghosh, B.K.Shikdar, P.P.Chaudhuri , "Programmable Cellular Automata (PCA) Based Advanced Encryption Standard (AES) Hardware", 9th International Conference on Cellular Automata for Research and Industry, ACRI, pp:271-274, 2010.

[20] R. Dogaru, I. Dogaru, and H. Kim, "Binary chaos synchronization inelementary cellular automata", *Int. J. Bifurcation and Chaos,* 19, 2009.

[21] R. Dogaru, I. Dogaru, H.Kim,"Synchronization in elementary cellular automata", *Proceedings of the 1Oth International Workshop on Multimedia Signal Processing and Transmission (MSPT'08*, July 21-22, pp. 35-40, 2008.

[22] I.Kokolakis, I.Andreadis, and P. Tsalids, "Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators," *Microprocessors and Microsystems,* Vol. 20, pp: 643–658, 1997.

[23] S. Nandi, B. K. Kar, and P. P. Chowdhuri, "Theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, Vol. 43, pp:1346–1357, 1994.

[24] P.Bak, C.Tang, , K.Wiesenfeld, "Self-organized criticality: an explanation of 1/f noise", Physical Review of Letters, pp: 381-384, 1987

[25] F. Seredynski, P. Bouvry, and A.Y. Zomaya, "Cellular automata computations and secret key cryptography," *Parallel Computing*, Vol.30, pp: 753-766, 2004.

[26] Alfred J. Menezes, Paul C. van Oorschot, Scott A.Vanstone "Handbook of Applied Cryptograph", CRC Press; 1 edition, pp:181-183, 1996, ISBN-10: 0849385237.

[27] ENT Test Suite, http:// www.fourmilab.ch/random

[28] G.Marsaglia, Diehard test, http://stat.fsu.edu/ ~geo/diehard.html, 1998.

[29] B.C. Johnson. Radix-b extensions to some common empirical tests for PRNGs. *ACM Trans. on Modeling and Comp. Sim.*, 6(4):261–273, 1996.

[30] D.E. Knuth. *The Art of Computer Programming, Volume 2,* Addison-Wesley, 3rd edition, 1998.

[31] M.M. Meysenburg and J.A. Foster. Randomness and GA performance, revisited.In *Proc. of GECCO'99*, volume 1, pages 425–432. Morgan Kaufmann, 1999.

[32] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 2nd edition, 1992.

[33] G. Marsaglia. Yet another RNG. Posted to sci.stat.math, 1994.

[34] M. Matsumoto and Y. Kurita. Twisted GFSR generators. *ACM Trans. on Modeling and Comp. Sim.*, 2(3):179–194,1992.

[35] B. Schneier. *Applied Cryptography*. John Wiley, 1994.

[36] S. Tezuka and P. L'Ecuyer. Efficient and portable combined Tausworthe Random Number Generators. *ACM Trans. on Modeling and Comp. Sim.*, 1(2):99–112, 1991.

[37] M.M. Meysenburg and J.A. Foster. The quality of PRNGs and simple genetic algorithm performance. In *Proc. of the 7th Int. Conference on Genetic Algorithms*, pp: 276–281, 1997.

[38] W. Stallings, "Cryptography and Network Security," *Prentice Hall, 3th Edition*, 2002, ISBN-10: 0130914290.