World Applied Sciences Journal 29 (Computer Sciences, Engineering and Its Applications): 106-110, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.29.csea.2234

Random Number Generation and Cryptography with Four Neighborhood Cellular Automata (4NCA)

R. Udayakumar, K.P. Kaliyamurthie, Khanaa and A.V.Allin Geo

School of Computing Science, Bharath University, Chennai-73, India

Abstract: Design of high-quality pseudo-random number generator is of great importance for many reasons like designing/testing of VLSI circuits, implementing of cryptographic protocols, etc. Although several techniques based on different techniques have been proposed, this paper addresses a design based on the four-neighborhood cellular automaton (4NCA). The 4NCA is an extension of three-neighborhood cellular automata (3NCA) and it provides more advantages over 3NCA like increasing of design space (2^{24} instead of 2^8), reduction of the design cost, etc. The manual selection of 4NCA from large design space for good random number generator is very difficult; we used Genetic Algorithm for optimization, where cell-entropy is considered as the fitness function. The outputs of the selected 4NCA are tested for randomness using Diehard Battery and it is seen that they pass all the tests like birthday spacing, parking lot, 3D sphere, DNA, etc. and better than shift -register, congruential and lagged Fibonacci generators. As an application, the generated pseudo-random numbers have been used as the key in the stream cipher cryptography, where images are encrypted/decrypted and satisfactory results have been found.

Key words: One- dimensional 3NCA and 4NCA · VLSI design and testing · Genetic Algorithm · Diehard battery test · Shift Register/Congruential Generator · Cryptography

INTRODUCTION

The linear feed-back shift register (LFSR) and 3NCA have many interesting uses in areas of complex digital designs and test applications. Both are cellular structure having a number of identical memory elements cascaded in one-dimensional fashion and use a transition function that determines the state at discrete interval of time. The LFSR is easy to design and use global information and XOR gates to perform modulo-2 operation for the next state evaluation. The primitive LFSR, which generate all non-zero states in one cycle, have important application in VLSI designs. On the other hand, the 3NCA, which have been considered as alternatives to LFSR, use local information for next state evaluation. Similar to primitive LFSR, there are primitive 3NCA that can be designed by the combination of the two linear rules 90 and 150 with null boundary conditions. Some notable applications are the generation of pseudorandom, pseudo exhaustive, deterministic test patterns [1-3], Signature analysis [4-6], Error correcting codes [7, 8], designing of Cryptographic systems [9-12].

Many generators have been used for years to produce high quality pseudo-random numbers like shift register generators, Congruential generators, Lagged Fibonacci generator, etc, but the generator based on 3NCA is a new prospective model. An extensive study has been done in this area for its simple, regular, cascadable structure and easy implementation in hardware. In this paper, a new structure called 4NCA [13, 14] has been used as the pseudo-random number generator.

The public communication network like Internet transports most of the private/public data and hence more attention is being paid to the security/privacy of the data and cryptography is one of the effective means to protect the data. It is known that stream cipher is an important class of symmetrical cryptography and its security depends on the high quality random numbers used as the key. It is seen that the 4NCA selected using GA can produce high-quality pseudo-random numbers and hence, they can be used in the stream cipher securely. For verification, several images have been encrypted with the pseudo-random bits

Corresponding Author: R. Udayakumar, School of Computing Science, Bharath University, Chennai-73, India.

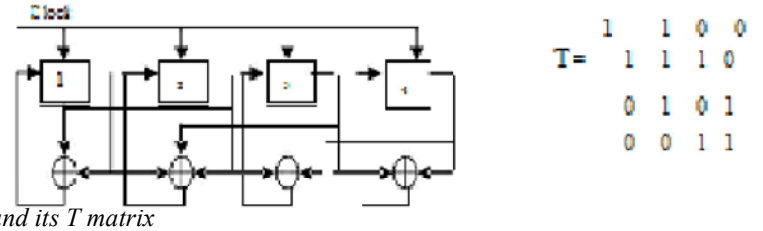

Fig. 1: *A 4-cell 3NCA and its T matrix*

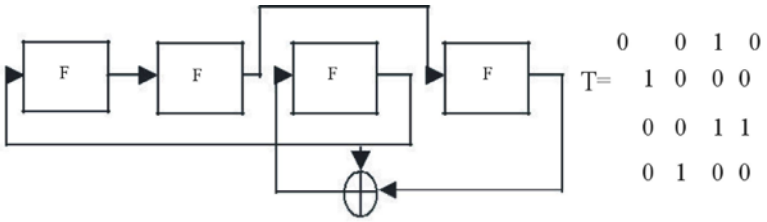

Fig. 2: *A 4-cell 4NCA and its T matrix*

generated by the 4NCA and it is seen that encrypted images show the same randomness as the key being used [15].

The paper is organized as follows. In Section 2, the preliminaries of both 3NCA and 4NCA are given and the Section 3 describes the design of the proposed 4NCA based pseudo-random number generator. In Section 4, the tests for randomness are given. Section 5 addresses the stream cipher technique and the paper is concluded in Section 6.

Preliminaries of the 3NCA and 4NCA: For clarity, the basic design principles of the 3NCA and 4NCA are given in this section. Both are almost similar in the design and appearance. The design of 3NCA is now addressed.

Three-Neighborhood Cellular Automata (3NCA): J. Von Neuman [16] in early 50's, first proposed a complex cellular structure and called Cellular Automata (CA). Later, S. Wolfram [17] investigated it and proposed a very simple regular 1-D structure consisting of cells that follow a linear function for its evolution. The next state of each cell depends on itself, left and right neighbors and evolves in discrete time steps using a transition function. The state transition function for cell i, where $1 \le i \le n$, can be written as $s_i(t+1) = f_i(s_i(t), s_{i+1}(t), s_{i-1}(t))$, where $s_i(t)$ is the state of the cell i at time t and (i-1) & (i+1) are the nearest left and right neighbors, respectively. Since there are eight combinations for 3-neighborhood dependency and each combination can produce either 0 or 1, so there exist $2^8 = 256$ transition rules for 3NCA and only 7 are linear rules. Generally, a matrix representation is used to represent 3NCA, called characteristic matrix, T and it is used in next state computation and for overall characterization of the CA. The T matrix is formed as $T(i, j) = 1$ if the cell-i is connected with cell-j and $T(i, j) = 0$ otherwise. The design of a 4-cell 3NCA is illustrated below.

Example 1: A 4-cell 3NCA is shown in Fig. 1, where the cells 1, 2, 3 and 4 are connected using rules 150, 150, 90 and 150, respectively. The T matrix representation is also shown.

Four-Neighborhood Cellular Automata (4NCA): The 4NCA, which is an extension of the 3NCA, is first proposed in [13]. The extension is done by adding one more nearest neighbor to each cell, where the additional cell is taken either from left or right side. Therefore, each cell i may have either of following neighborhood dependency

- $q[i](t+1)=f_i(q[i-2](t),q[i-1](t),q[i](t),q[i+1](t))$ (two left neighbors, self and one right neighbor dependency)
- $q[i](t+1)=f_i(q[i-1](t), q[i](t),q[i+1](t),q[i+2](t))$ (two right neighbors, self and one left neighbor dependency).

The same T matrix representation, similar to the 3NCA can be used for 4NCA representation. An example is shown below for further clarification [9].

Example 2: The design of the 3NCA as shown in Example 1 is shown in Fig. 2. The matrix representation is also shown that needs only five 1's in the T matrix, whereas the same implementation requires nine 1's as in case of 3NCA (also 4NCA and 3NCA require one 2-input XOR gate and five 2-input XOR gates, respectively).

The 4NCA use 4-neighborhood dependency and this dependency is selected from *5* neighboring cells. Since both the additional left and right cells will never be considered at the same time, therefore *24* combinations (out of 32) exist per cell and it results $2^{24} = 16M$ transition rules. It's very large rule space and the rule specification as used in 3NCA may not be feasible. Since we need only linear rules in the design, a method similar to [12] as given below is followed

$$q[i](t+1) = P.q[i-2](t) \oplus Q.q[i-1](t) \oplus R.q[i](t) \oplus S.q[i+1](t) \oplus T.q[i+2](t)$$

where, *P, Q, R, S* and *T* are the impact coefficients of the cells *i-2, i+1, i, i+1* and *i+2*, respectively such that $P.T \neq 1$. Only five bits are used to represent the linear 4NCA rules.

Design of Pseudo-random Number Generator: Basically, the design of 4NCA-based pseudo-random number generator (4NCA-RNG) means to select those 4NCA that give high quality random bits on their evolution. Since its design rule space is very large, we used GA for optimal searching. Some GA basics are now addressed.

Some Basics of Genetic Algorithm: Genetic Algorithm is a search technique designed to mimic the process of natural selection and evolution. A *population* in GA consists of a fixed number of *chromosomes*, each of which is a fixed-length string of *genes*. The *fitness* of a chromosome is some measure of how desirable it is to have that chromosome in the population. A *generation* is a time step in which several events occur. Some of the most "unfit" chromosomes die and are removed from the population. To replace these, some number of *crossover* operations is applied to the population. A crossover is an operation analogous to mating or gene splicing. It combines part of one chromosome with part of another chromosome to create a new chromosome. Finally, some amount of *mutation* occurs in the population, in which genes are changed randomly with some low probability. The process is repeated until a specified termination criterion is met. Each 4NCA linear rule is treated as chromosome and it is represented by a 5-bit string. The fitness function is the cell *entropy*, the calculation of which is given below.

Entropy Estimation: The quality of random numbers can be measured in a variety of ways and one common method is to compute the *information density,* or *entropy,* in a series of numbers. In order to calculate the entropy of a bit sequence, the bit sequence is divided into fixed-length sub-sequences and the all possible yields per sub-sequences are recorded. If k is the number of state of each cell, then for the sub-sequences of length h, total possible sub-sequence is k^h. Thus the entropy, E can be calculated as $E = -\sum_{j=1}^{k^h} P_{h_j} \; P_{h\,j} \log_2 P_{h\,j}$, where hj is the j^{th} permutation of total k^h permutations and $P_{h\,j}$ is the probability of subsequences hj. It is maximized when all $P_{h\,j}$ are same and equal to $1/k$.

Proposed Algorithm for 4NCA-RNG: Initially, an n-cell 4NCA with an arbitrary 5-bit cell rule is considered. The cells are then initialized with a random non-zero value and evolved for certain time steps. The calculation of the fitness and the final selection of the 4NCA are made. The proposed algorithm is given below.

Algorithm: Randomizer-4NCA():

- Consider a 4NCA with finite number of cells and assign an arbitrary 5-bit rule to each cell, set the fitness value of each cell to 0.
- Initialize with a non-zero random value and evolve the 4NCA for *M* time steps.
- Calculate the fitness value of each cell's bit sequence.
- Repeat Step-2 and Step-3 for *C* times with different random seeds.
- Determine the accumulated fitness values of each cell and incorporate Genetic operators to obtain new rule for cell *i* as.

  Selection: (a) If the neighbor's fitness are both smaller than cell *i*, then the rule of cell *i* keeps unchanged, (b) If only one neighbor's fitness value is larger than the fitness of cell *i*, then the new rule of cell *i* is substituted by the rule of larger one.

  C**rossover:** If the two neighbors' fitness values are larger than fitness of cell *i*, then the neighbors' rules are mated and select an offspring at random as the new rule of cell *i*. **Mutation** – Each gene of every chromosome performs the mutation (just flip the value of gene) operation at probability P_m, which is very less.
- Stop if terminating conditions are met *else* goto Step-2.

Table 1: *Test results for Diehard Battery for randomness*

Test Name	Shift Register	Extended Congruential	Lagged Fibonacci	4NCA1	4NCA2
Birthday spacing	Y	Y	N	Y	Y
OPERMS Permutation 1	Y	Y	Y	Y	Y
Binary Rank for 32× 32	N	Y	Y	Y	Y
Count 1's	Y	Y	N	Y	Y
Parking lot	Y	Y	Y	Y	Y
Minimum distance	Y	Y	N	Y	Y
3D SPHERE	Y	Y	Y	Y	Y
SQEEZE	Y	Y	Y	Y	Y
Overlapping Sums	Y	Y	Y	Y	Y
Runs up 2	N	N	Y	Y	Y
Runs down 2	Y	Y	Y	Y	Y
CRAPS	Y	Y	N	Y	Y
OQSO	NA	NA	NA	Y	Y
DNA	NA	NA	NA	Y	Y

Caption: Y - Passed the Test, N - Failed the Test, NA – Results not available

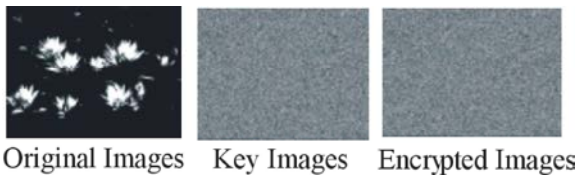

Fig. 3: *Encryption of Images with Key Images*

As an estimation of the complexity, it is seen that out of *24* combinations, *22* combinations are valid and therefore, there exists $(22)^n$ possible 4NCA, which are to be searched for finding best 4NCA rules to generate good-quality random bits.

Experimentation Setup and Statistic Test for Randomness: We consider a 50-cell 4NCA, i.e., $n = 50$, in our experimentation. The 4NCA is evolved for $M = 4096$ time steps with a random seed value. In total, $C = 300$ different random seed values are considered for determining the cell entropy. It gives 50 random sequences of 4096 bits each and if these fifty 4096-bit sequences are connected to form one long sequence, it becomes a sequence of length 204800 bits. Since this process is repeated for 300 times, thus one can obtain a sequence of a little more than 61 million binary bits. This bit sequence is actually put into the Diehard battery test [18] for randomness. The results of test for two 4NCA are summarized in the Table 1. It is seen that they outperform other generators and pass all the tests.

Stream Cipher with 4NCA: The pseudo-random numbers as generated by 4NCA are used as the secret keys in the symmetric stream encryption technique. Let P be the plain text containing n bits and the key K is the n bits, then the cipher text C contains n bits as obtained using $c_i = p_i \oplus k_i$, where each message bit p_i *is XORed with the key bit* k_i *to get each cipher text bit* c_i. *Note that the original plain text can be recovered by applying the same operation on the cipher text as* $p_i = ci \oplus k_i$. This algorithm is *theoretically is unbreakable if the key is truly random bits and uses only once. The encryption of a black and white image with the pseudo-random key generated using 4NCA. key image and encrypted image is shown in Figure 3.*

CONCLUSIONS

The design of high-quality pseudo-random number generator using 4NCA is addressed. As it has huge design space, GA has been used as an optimizer to select best possible 4NCA. The Dirhard battery test has been used to test the randomness of bits generated and it is seen that it passes all tests and even better than the common generators such as Shift Register Generator, Congruential Generator and Lagged Fibonacci Generator. The generator can produce high quality random numbers quickly and can be implemented conveniently by hardware. As an application, the pseudo-random numbers have been used as the secret keys in the stream cipher of images and it has been found that the encrypted images are well secured.

REFERENCES

1. Hortensius P.D., *et al*., 1989. Cellular Automata-based pseudorandom number generators for Built-in Self-test, IEEE Trans. On Computer-Aided Design, 8: 842-859.

2. Bardell, P.H., 1990. Analysis of Cellular Automata used as Pseudorandom Pattern Generators, Proceedings IEEE Intl. Test Conf., pp: 762-767.
3. Serra M., *et al*., 1990. Analysis of One Dimensional Cellular Automata and their Aliasing Properties, IEEE Trans. Computer-Aided Design, 9: 767-778.
4. Das A.K., *et al*, 1990. Signature Analyzer Based on additive Cellular Automata, Proc. 20th Fault Tolerant Computing Systems, pp: 265-272.
5. Hortensius P.D., *et al*., 1990. Cellular Automata Based Signature Analysis for Built-In Self-Test, IEEE Trans. Computers, 39: 1273-1283.
6. Tsalides P., 1990. Cellular Automata Based Built-In Self-Test Structures for VLSI Systems, Elect. Lett., 26: 1350-1352.
7. Chowdhury D.R., *et al*., 1994. Design of CAECC-Cellular Automata Based Error Correcting Codes, IEEE Trans. Computers, 43: 759-764.
8. Chattopadhyay S., *et al*., 1995. Board level fault diagnosis using cellular automata array, 8th Int'l Conf. on VLSI Design, pp: 343.
9. Nandi S., *et al*., 1994. Theory and Application of Cellular Automata in Cryptography, IEEE Trans. on Computers, 43: 1346-1352.
10. Tomassini, M., *et at*., 2001. Cryptography with cellular automata, Applied Soft Computing Journal, 1: 151-160.
11. Seredynski F., *et al*., 2004. Cellular automata computations and secret key cryptography, Parallel Computing, 30: 753-766.
12. Xuelong, Z., *et al*., 2005. A Symmetric cryptography based on extended cellular automata, Intl. Conf on Systems, Man and Cybernetics, 1: 499-503.
13. Biswas, G.P., 2003. Modification of 3-Neighborhood Cellular Automata to 4-Neighborhood Cellular Automata to Increase their Capabilities, Jour of CSI, 33: 27-34.
14. Udayakumar, R., V. Khanaa and K.P. Kaliyamurthie, 2013. Optical Ring Architecture Performance Evaluation using ordinary receiver, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(6): 4742-4747.
15. Thooyamani, K.P., V. Khanaa and R. Udayakumar, 2013. Weed control system of tea garden using GIS based database Management system, Middle-East Journal of Scientific Research, ISSN:1990-9233 15(12): 1702-1706
16. Von Neumann, J., 1966. The Theory of Self-Reproducing Automata, Univ. of Illions Press.
17. Wolfram, S., 1986. Random sequence generation by Cellular Automata, Advances in applied mathematics, 7: 123-169.
18. Marsaglia, G., 1998. Diehard, http:// stat.fsu.edu/ ~geo/ diehard.html.