Study of Five-Neighborhood Linear Hybrid Cellular Automata and Their Synthesis

Swapan Maiti(✉) and Dipanwita Roy Chowdhury(✉)

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, Kharagpur, India
swapankumar_maiti@yahoo.co.in, drc@cse.iitkgp.ernet.in

Abstract. Cellular automata (CA) is universally known as very good pseudorandom sequence generator. It has wide applications in several fields like VLSI design, error-correcting codes, test pattern generation, cryptography etc. Most of these applications use 3-neighborhood one dimensional CA. Cellular automata have been chosen as a better crypto-primitives for providing very good pseudorandom sequences and their high diffusion property. The randomness and diffusion properties can be increased with the increase of the size of neighborhood radius of the CA cell. In this work, we study a class of 5-neighborhood null boundary linear CA. We present an algorithm for synthesizing 5-neighborhood linear CA from its characteristic polynomial by assuming that some of the CA sub-polynomials are available.

Keywords: Cellular automata · 5-neighborhood linear rules · CA synthesis algorithm

1 Introduction

Cellular Automata (CA) have long been of interest to researchers for their theoretical properties and practical applications. It was initiated in the early 1950's by John von Neumann [12] and Stan Ulam as a general framework for modeling complex structures capable of self-reproduction and self-repair. In 1986, Wolfram first applied CA in pseudorandom number generation [15]. CA has made understanding of many occurrences in nature easier. The simple and regular structure of CA has attracted researchers and practitioners of different fields. In the last two decades, one-dimensional (1-D) CA based Pseudorandom Number Generators (PRNGs) have been extensively studied [2,5,10,11]. Though the recent interest is more focused on two-dimensional (2-D) CA PRNGs [9,13] since it seems that their randomness is much better than that of 1- D CA PRNGs, but considering the design complexity and computation efficiency, it is quite difficult to conclude which one is better. Compared to 2-D CA PRNGs, 1-D CA PRNGs are easier to be implemented in a large scale [3,8,14]. Random bit generators play an important role in different computer simulation methods such as Monte Carlo techniques, Browmian dynamics, stochastic optimization, computer-based

D. Giri et al. (Eds.): ICMC 2017, CCIS 655, pp. 68–83, 2017.
DOI: 10.1007/978-981-10-4642-1_7

gaming, test pattern generation for VSLI circuit test, error-correcting codes, image processing, neural networks and cryptography etc. Most of these works are devoted to the study of cellular automata as pseudorandom bit generators. A central problem in any stream cipher scheme is to generate long, unpredictable random key sequences and Cellular Automata resolves this problem.

In most of all these applications, 1-D elementary cellular automata (i.e. three-neighborhood CA) are used. There are also some applications [6,9,13] of five or more neighborhood 2-D CA but that need more hardware complexity. In [7], it has been shown a 4-neighborhood nonlinear 1-D CA as a better cryptographic primitive. The randomness and diffusion properties of the CA can be developed with the increase of the size of neighborhood radius of the CA cell. More diffusion property of CA can make fast initialization of a stream cipher. In this paper, we study 5-neighborhood linear 1-D CA for providing very good pseudorandom sequences and high diffusion. We present an algorithm for synthesizing the CA.

This paper is organized as follows. Following the introduction, the basics of CA are presented in Sect. 2. In Sect. 3, we present 5-neighborhood Linear Hybrid Cellular Automata with the CA transition matrix and the characteristic polynomial. A recurrence relation is introduced for determining the characteristic polynomial and a CA synthesis algorithm is presented. We also present the randomness and diffusion properties of 5-neighborhood CA rule vectors and the comparison of their properties with 3/4 neighborhood CA. Finally, the paper is concluded in Sect. 4.

2 Basics of Linear Cellular Automata

Cellular Automata are studied as mathematical model for self organizing statistical systems [12]. CA can be one-dimensional or multi-dimensional. One-dimensional CA random number generators have been extensively studied in the past [4,11,15]. In one-dimensional CA, they can be considered as an array of cells where each cell is a one bit memory element. The neighbor set N(i) is defined as the set of cells on which the state transition function of the i-th cell is dependent on each iteration. In three-neighborhood CA, each cell evolves in every time step based on some combinatorial logic on the cell itself and its two nearest neighbors. More formally, for a three-neighborhood CA, the neighbor set of i-th cell is defined as $N(i) = \{s_{i-1}, s_i, s_{i+1}\}$. The state transition function of is i-th cell of 3-neighborhood CA is as follows:

$$s_i^{t+1} = f_i(s_{i-1}^t, s_i^t, s_{i+1}^t)$$

where, s_i^t denotes the current state of the i-th cell at time step t and s_i^{t+1} denotes the next state of the i-th cell at time step t+1 and f_i denotes some combinatorial logic for i-th cell. The set of all feedback functions is considered as ruleset for the CA. Since, a three-neighborhood CA having two states (0 or 1) in each cell, can have $2^3 = 8$ possible binary states, there are total $2^{2^3} = 256$ possible boolean functions, called rules. Each rule can be represented as an decimal integer from 0 to 255. If the combinatorial logic for the rules have only Boolean XOR operation, then it is called linear or additive rule. Some of the three-neighborhood additive CA rules are 0, 60, 90, 102, 150 etc. Moreover, if the combinatorial logic contains AND/OR operations, then it is called non-linear rule. An n cell CA with cells $\{s_1, s_2, \cdots, s_n\}$ is called null boundary CA if $s_{n+1} = 0$ and $s_0 = 0$. Similarly for a periodic boundary CA $s_{n+1} = s_1$. A CA is called uniform, if all its cells follow the same rule. Otherwise, it is called non-uniform or hybrid CA. If all the ruleset of a hybrid CA are linear, then we call the CA a linear one. However, out of all possible Boolean functions, called rules, only two are of prime interest i.e. Rule 90 and 150 (ascertained from the decimal value of their position in the truth table). The state of the i-th cell at time instant t can be expressed as:

$$s_i^{t+1} = s_{i-1}^t \oplus d_i.s_i^t \oplus s_{i+1}^t, d_i = \begin{cases} 0, \text{ if } d_i \rightarrow \text{Rule 90} \\ 1, \text{ if } d_i \rightarrow \text{Rule 150} \end{cases}$$

Thus, an LHCA can be completely specified by a combination of Rule 90 and 150, denoted as an n-tuple $[d_1, d_2, \cdots, d_n]$. An example of a 5-cell CA $\mathcal{L}$ can be found in Fig. 1, specified by the rule vector $[1, 1, 1, 1, 0]$. Further details of CA can be found in [4].

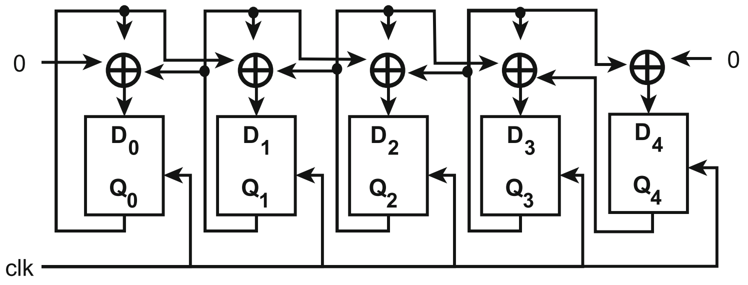

Fig. 1. 3-neighborhood null boundary LHCA $\mathcal{L}$ with rule vector [1, 1, 1, 1, 0]

3 5-Neighborhood Linear Cellular Automata

In the previous section, we have studied 1D elementary CA (i.e. 3-neighborhood CA) [4,11]. In this section, we consider a 5-neighborhood null boundary n-cell Linear Hybrid CA (LHCA) denoted by $\{s_1, s_2, \cdots, s_n\}$, where the state of a cell at a given instant is updated based upon its five neighboring cells including itself and because of null boundary $s_{-1} = s_0 = 0$, $s_{n+1} = s_{n+2} = 0$. More formally, for a five-neighborhood CA, the neighbor set of i-th cell is defined as $N(i) = \{s_{i-2}, s_{i-1}, s_i, s_{i+1}, s_{i+2}\}$. The state transition function of is i-th cell of 5-neighborhood CA is as follows:

$$s_i^{t+1} = f_i(s_{i-2}^t, s_{i-1}^t, s_i^t, s_{i+1}^t, s_{i+2}^t)$$

Table 1. Linear rules of 5-neighborhood LHCA

Rules	State transition function of i^{th} cell
$Rule_0$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i+2}^t$
$Rule_1$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i+1}^t \oplus s_{i+2}^t$
$Rule_2$	$s_i^{t+1} = s_{i-2}^t \oplus s_i^t \oplus s_{i+2}^t$
$Rule_3$	$s_i^{t+1} = s_{i-2}^t \oplus s_i^t \oplus s_{i+1}^t \oplus s_{i+2}^t$
$Rule_4$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_{i+2}^t$
$Rule_5$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_{i+1}^t \oplus s_{i+2}^t$
$Rule_6$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_i^t \oplus s_{i+2}^t$
$Rule_7$	$s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t \oplus s_{i+2}^t$

where, s_i^t denotes the current state of the i-th cell at time step t and s_i^{t+1} denotes the next state of the i-th cell at time step t+1 and f_i denotes some combinatorial logic for i-th cell. Since, a 5-neighborhood CA having two states (0 or 1) in each cell, can have $2^5 = 32$ possible binary states, there are total $2^{2^5} = 2^{32}$ possible boolean functions. Out of all possible Boolean functions, called rules, there are total $2^5 = 32$ possible linear rules. Based on neighborhood radius exactly 5, there are only $2^3 = 8$ liner rules shown in Table 1.

Table 2. Counting rule vectors of max. period 5-bit 5-neighborhood CA

$Rule_1$	2						
$Rule_2$	0	2					
$Rule_3$	2	6	2				
$Rule_4$	2	2	2	4			
$Rule_5$	6	2	4	5	2		
$Rule_6$	2	4	2	2	6	5	
$Rule_7$	4	5	6	2	5	8	2
	$Rule_0$	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$

For all possible pair of these 8 linear rules, maximum period 5-neighborhood CA rule vectors can be obtained. Table 2 shows the number of rule vectors obtained for maximum period 5-bit 5-neighborhood CA against each pair of the linear rules shown in Table 1. From Table 2, we see that only the pair of rule combinations, ($Rule_5$, $Rule_7$), provides largest number of rule vectors (i.e. 8). Therefore, we consider these two linear rules (i.e. $Rule_5$, $Rule_7$), denoted as R_0 and R_1, respectively, to design 5-neighborhood LHCA. These two linear rules can again be specified as follows:

$$R_0 : s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_{i+1}^t \oplus s_{i+2}^t$$

$$R_1 : s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t \oplus s_{i+2}^t$$

where, s_i^t is the current state and s_i^{t+1} is the next state of the i-th cell of the CA. Thus, the state transition function of i-th cell of the CA can be expressed as:

$$s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus d_i.s_i^t \oplus s_{i+1}^t \oplus s_{i+2}^t,\ d_i = \begin{cases} 0, & \text{if } i^{th} \text{ cell follows rule } R_0 \\ 1, & \text{if } i^{th} \text{ cell follows rule } R_1 \end{cases}$$

Thus, a five-neighborhood n-cell LHCA $\mathcal{L}$ denoted by $\{s_1, s_2, \cdots, s_n\}$, can be completely specified by a combination of these two rules R_0 and R_1, denoted as an n-tuple $[d_1, d_2, \cdots, d_n]$, called the rule vector of the CA. An example of a 5-cell null boundary 5-neighborhood CA can be found in Fig. 2, specified by the rule vector $[1, 1, 1, 0, 0]$.

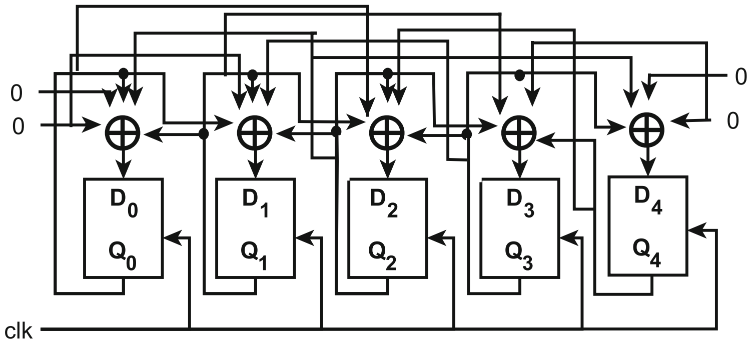

Fig. 2. 5-neighborhood null boundary LHCA $\mathcal{L}$ with rule vector [1, 1, 1, 0, 0]

A five-neighborhood n-cell LHCA $\mathcal{L}$ can be characterised by an $n \times n$ matrix, called characteristic matrix. The characteristic matrix A for the n-cell CA rule vector $[d_1, d_2, \cdots, d_n]$ is as follows:

$$A = \begin{bmatrix} d_1 & 1 & 1 & 0 & 0 & \cdots\cdots & \cdots & \cdots & 0 & 0 \\ 1 & d_2 & 1 & 1 & 0 & \cdots\cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 1 & d_3 & 1 & 1 & \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 1 & 1 & d_4 & 1 & \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\ \ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots\cdots\cdots & & & & 1 & d_{n-3} & 1 & 1 & 0 \\ \vdots & \cdots\cdots\cdots & & & & 1 & 1 & d_{n-2} & 1 & 1 \\ 0 & \cdots\cdots\cdots & & & & 0 & 1 & 1 & d_{n-1} & 1 \\ 0 & 0 & \cdots\cdots\cdots & & & 0 & 0 & 1 & 1 & d_n \end{bmatrix}$$

The state of a CA at time step t is an n-tuple formed from the states of the individual cells. The CA state is expressed in matrix form as follows

$$S^t = [s_1^t, \cdots, s_n^t]$$

The next state of the CA is denoted as

$$S^{t+1} = [s_1^{t+1}, \cdots, s_n^{t+1}]$$

The next-state of the CA, S^{t+1}, is computed as

$$\begin{aligned} (S^{t+1})^T &= A \cdot (S^t)^T \\ \text{or,} \qquad S^{t+1} &= ((S^{t+1})^T)^T \end{aligned}$$

where, A is the CA transition matrix and $(S^t)^T = [s_1^t, \cdots, s_n^t]^T$ (the superscript T represents the transpose of the vector) and the product is a matrix-vector multiplication over GF(2). It has been shown that $A \cdot (S^t)^T$ is indeed the next state of the CA. Therefore, the next state of the i^{th} cell is computed as the product of the i^{th} row of A and $(S^t)^T$ as follows:

$$\begin{aligned} s_i^{t+1} &= [0, \cdots, 0, 1, 1, d_i, 1, 1, 0, \cdots, 0] \\ &\quad \cdot [s_1^t, \cdots, s_{i-2}^t, s_{i-1}^t, s_i^t, s_{i+1}^t, s_{i+2}^t \cdots, s_n^t]^T \\ &= s_{i-2}^t + s_{i-1}^t + d_i \cdot s_i^t + s_{i+1}^t + s_{i+2}^t \end{aligned}$$

The characteristic polynomial Δ_n of the n-cell CA is defined by

$$\Delta_n = |x\mathcal{I} - A|$$

where, x is an indeterminate, $\mathcal{I}$ is the identity matrix of order n, and A is the CA transition matrix. The matrix $x\mathcal{I} - A$ is called the characteristic matrix of the CA. The characteristic polynomial is a degree n polynomial in x.

The following example clearly illustrates how the characteristic polynomial of a 5-neighborhood linear CA can be computed using the characteristic matrix of the CA.

Example 1: Let us consider a 5-cell null boundary 5-neighborhood linear CA with the rule vector [1, 1, 1, 0, 0]. We have $[d_1, d_2, d_3, d_4, d_5] = [1, 1, 1, 0, 0]$. The transition matrix A is as follows:

$$A = \begin{bmatrix} d_1 & 1 & 1 & 0 & 0 \\ 1 & d_2 & 1 & 1 & 0 \\ 1 & 1 & d_3 & 1 & 1 \\ 0 & 1 & 1 & d_4 & 1 \\ 0 & 0 & 1 & 1 & d_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The corresponding characteristic matrix is as follows:

$$x\mathcal{I} - A = \begin{bmatrix} x+d_1 & 1 & 1 & 0 & 0 \\ 1 & x+d_2 & 1 & 1 & 0 \\ 1 & 1 & x+d_3 & 1 & 1 \\ 0 & 1 & 1 & x+d_4 & 1 \\ 0 & 0 & 1 & 1 & x+d_5 \end{bmatrix} = \begin{bmatrix} x+1 & 1 & 1 & 0 & 0 \\ 1 & x+1 & 1 & 1 & 0 \\ 1 & 1 & x+1 & 1 & 1 \\ 0 & 1 & 1 & x & 1 \\ 0 & 0 & 1 & 1 & x \end{bmatrix}$$

where, x is an indeterminate, $\mathcal{I}$ is the identity matrix with dimension 5, and A is the CA transition matrix shown above. The characteristic polynomial Δ_5 of the 5-cell CA is defined as follows:

$$\Delta_5 = |x\mathcal{I} - A|$$

$$\Delta_5 = \begin{vmatrix} x+1 & 1 & 1 & 0 & 0 \\ 1 & x+1 & 1 & 1 & 0 \\ 1 & 1 & x+1 & 1 & 1 \\ 0 & 1 & 1 & x & 1 \\ 0 & 0 & 1 & 1 & x \end{vmatrix} = x^5 + x^4 + x^2 + x + 1$$

Theorem 1. Let Δ_n be the characteristic polynomial of a n-cell null boundary 5-neighborhood Linear CA with rule vector $[d_1, d_2, \cdots, d_n]$. Δ_n satisfies the following recurrence relation:

$$\begin{aligned} &\Delta_{-3} = 0, \quad \Delta_{-2} = 0, \quad \Delta_{-1} = 0, \quad \Delta_0 = 1 \\ &\Delta_n = (x + d_n)\Delta_{n-1} + \Delta_{n-2} + (x + d_{n-1})\Delta_{n-3} + \Delta_{n-4},\ n > 0 \end{aligned} \tag{1}$$

Proof: Consider the transition matrix A for the n-cell null boundary 5-neighborhood Linear CA with rule vector $[d_1, d_2, \cdots, d_n]$

$$A = \begin{bmatrix} d_1 & 1 & 1 & 0 & 0 & \cdots\cdots & \cdots & \cdots & 0 & 0 \\ 1 & d_2 & 1 & 1 & 0 & \cdots\cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 1 & d_3 & 1 & 1 & \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 1 & 1 & d_4 & 1 & \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & 1 & d_{n-3} & 1 & 1 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & 1 & 1 & d_{n-2} & 1 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 1 & d_{n-1} & 1 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 & 1 & d_n \end{bmatrix}$$

The characteristic polynomial Δ_n of the CA is defined by

$$\Delta_n = |x\mathcal{I} - A|$$

$$\Delta_n = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 \cdots\cdots & \cdots & \cdots & 0 & 0 \\ 1 & x+d_2 & 1 & 1 & 0 \cdots\cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 1 & x+d_3 & 1 & 1 \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 1 & 1 & x+d_4 & 1 \cdots\cdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots\ \vdots\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots\ 1 & x+d_{n-3} & 1 & 1 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots\ 1 & 1 & x+d_{n-2} & 1 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots\ 0 & 1 & 1 & x+d_{n-1} & 1 \\ 0 & 0 & \cdots & \cdots & \cdots\cdots\ 0 & 0 & 1 & 1 & x+d_n \end{vmatrix}$$

By expanding the determinant shown above with respect to the last row, we can compute Δ_n as follows: $\Delta_n = (x + d_n) * \Delta_{n-1} + 1 * B + 1 * C$, where B and C with dimension $(n-1) \times (n-1)$ are as follows:

$$B = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 & \cdots\cdots & & \cdots & \cdots & 0 \\ 1 & x+d_2 & 1 & 1 & 0 & \cdots\cdots & & \cdots & \cdots & 0 \\ 1 & 1 & x+d_3 & 1 & 1 & \cdots\cdots & & \cdots & \cdots & \vdots \\ 0 & 1 & 1 & x+d_4 & 1 & \cdots\cdots & & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots & & 1 & x+d_{n-3} & 1 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots & & 1 & 1 & x+d_{n-2} & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots & & 0 & 1 & 1 & 1 \end{vmatrix}$$

and

$$C = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 & \cdots\cdots & & \cdots & 0 & 0 \\ 1 & x+d_2 & 1 & 1 & 0 & \cdots\cdots & & \cdots & \cdots & 0 \\ 1 & 1 & x+d_3 & 1 & 1 & \cdots\cdots & & \cdots & \cdots & \vdots \\ 0 & 1 & 1 & x+d_4 & 1 & \cdots\cdots & & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots & & 1 & x+d_{n-3} & 1 & 0 \\ \vdots & \cdots & & \cdots & \cdots\cdots & & 1 & 1 & 1 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots & & 0 & 1 & x+d_{n-1} & 1 \end{vmatrix}$$

By expanding the determinant B with respect to the last column, we can compute B as follows: $B = \Delta_{n-2} + D$, where D with dimension $(n-2) \times (n-2)$ is as follows:

$$D = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 & \cdots\cdots & & \cdots & 0 \\ 1 & x+d_2 & 1 & 1 & 0 & \cdots\cdots & & \cdots & \cdots \\ 1 & 1 & x+d_3 & 1 & 1 & \cdots\cdots & & \cdots & \cdots \\ 0 & 1 & 1 & x+d_4 & 1 & \cdots\cdots & & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots & & 1 & x+d_{n-3} & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots & & 0 & 1 & 1 \end{vmatrix}$$

By expanding the determinant C with respect to the last column, we can compute C as follows: $C = E + F$, where E and F with dimension $(n-2) \times (n-2)$ are as follows:

$$E = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 \cdots\cdots & \cdots & 0 \\ 1 & x+d_2 & 1 & 1 & 0 \cdots\cdots & \cdots & \cdots \\ 1 & 1 & x+d_3 & 1 & 1 \cdots\cdots & \cdots & \cdots \\ 0 & 1 & 1 & x+d_4 & 1 \cdots\cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \; \vdots \; \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots \; 1 & x+d_{n-3} & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots \; 1 & 1 & 1 \end{vmatrix}$$

and

$$F = \begin{vmatrix} x+d_1 & 1 & 1 & 0 & 0 \cdots\cdots & \cdots & 0 \\ 1 & x+d_2 & 1 & 1 & 0 \cdots\cdots & \cdots & \cdots \\ 1 & 1 & x+d_3 & 1 & 1 \cdots\cdots & \cdots & \cdots \\ 0 & 1 & 1 & x+d_4 & 1 \cdots\cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \; \vdots \; \vdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots\cdots \; 1 & x+d_{n-3} & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots\cdots \; 0 & 1 & x+d_{n-1} \end{vmatrix}$$

By expanding the determinant F with respect to the last column, we can compute F as follows:

$$F = (x + d_{n-1}) * \Delta_{n-3} + \Delta_{n-4}$$

Note that the determinant E can be easily found by changing rows into columns and columns into rows of the determinant D, therefore, D and E determines the same polynomial and so, D+E determines zero in GF(2). Finally, we have

$$\begin{aligned} \Delta_n &= (x + d_n) * \Delta_{n-1} + 1 * B + 1 * C \\ &= (x + d_n) * \Delta_{n-1} + (\Delta_{n-2} + D) + (E + F) \\ &= (x + d_n) * \Delta_{n-1} + \Delta_{n-2} + F \\ &= (x + d_n) * \Delta_{n-1} + \Delta_{n-2} + (x + d_{n-1}) * \Delta_{n-3} + \Delta_{n-4} \end{aligned}$$

Theorem 1 provides an efficient algorithm to compute the Characteristic polynomial of a CA. Initially, Δ_{-3}, Δ_{-2}, Δ_{-1} are all set to zero and Δ_0 is set to one. Equation (1) is applied to obtain Δ_1. It is then reapplied to calculate Δ_2 from Δ_{-2} to Δ_1, Continuing, the polynomials $\Delta_3, \Delta_4, \cdots, \Delta_n$ are computed.

The following example clearly illustrates how the characteristic polynomial of a 5-neighborhood linear CA can be computed using the recurrence relation shown above. Table 3 shows characteristic polynomials of a 5-cell null boundary 5-neighborhood linear CA.

Example 2: Let us consider a 5-cell null boundary 5-neighborhood linear CA with the rule vector [1, 1, 1, 0, 0]. We have, $[d_1, d_2, d_3, d_4, d_5] = [1, 1, 1, 0, 0]$

$$\begin{aligned}
\Delta_{-3} &= 0, \quad \Delta_{-2} = 0, \quad \Delta_{-1} = 0, \quad \Delta_0 = 1 \\
\Delta_1 &= (x+d_1)\Delta_0 + \Delta_{-1} + (x+d_0)\Delta_{-2} + \Delta_{-3} \\
&= (x+1).1 + 0 + 0 + 0 = x+1 \\
\Delta_2 &= (x+d_2)\Delta_1 + \Delta_0 + (x+d_1)\Delta_{-1} + \Delta_{-2} \\
&= (x+1)(x+1) + 1 + 0 + 0 = x^2 \\
\Delta_3 &= (x+d_3)\Delta_2 + \Delta_1 + (x+d_2)\Delta_0 + \Delta_{-1} \\
&= (x+1)x^2 + (x+1) + (x+1) + 0 \\
&= x^3 + x^2 \\
\Delta_4 &= (x+d_4)\Delta_3 + \Delta_2 + (x+d_3)\Delta_1 + \Delta_0 \\
&= (x+0)(x^3+x^2) + x^2 + (x+1)(x+1) + 1 \\
&= x^4 + x^3 + x^2 + x^2 + 1 + 1 \\
&= x^4 + x^3 \\
\Delta_5 &= (x+d_5)\Delta_4 + \Delta_3 + (x+d_4)\Delta_2 + \Delta_1 \\
&= (x+0)(x^4+x^3) + (x^3+x^2) + (x+0)(x^2) + (x+1) \\
&= x^5 + x^4 + x^3 + x^2 + x^3 + x + 1 \\
&= x^5 + x^4 + x^2 + x + 1
\end{aligned}$$

3.1 Synthesis of 5-Neighborhood Linear CA

In this section, we present an algorithm Algorithm 1 for synthesizing 5-neighborhood CA from its characteristic polynomial.

Algorithm 1. Synthesis Algotithm

Input: The characteristic polynomial of an n-cell CA, Δ_n
Output: 5-neighborhood rule vector $[d_1, d_2, \cdots, d_n]$
Suppose, Δ_{n-1}, Δ_{n-2} and Δ_{n-3} are known and $\Delta_{-3} = \Delta_{-2} = \Delta_{-1} = 0, \Delta_0 = 1$.
Here, all operations are done in $GF(2)$.

```
1. Consider  $\Delta_n = (x+d_n)\Delta_{n-1} + \Delta_{n-2} + (x+d_{n-1})\Delta_{n-3} + \Delta_{n-4}$
2. Compute  $x+d_n$  using Division Algorithm
3. For k=n downto 3
4.       Consider  $\Delta_k = (x+d_k)\Delta_{k-1} + \Delta_{k-2} + (x+d_{k-1})\Delta_{k-3} + \Delta_{k-4}$
5.       Compute  $x+d_{k-1}$ *and* $\Delta_{k-4}$  using Division Algorithm
   End  for
6. Consider  $\Delta_1 = (x+d_1)\Delta_0$
7. Compute  $x+d_1$
8. **Return** $[d_1, d_2, \cdots, d_n]$
```

Explanation: Suppose, Δ_{n-1}, Δ_{n-2} and Δ_{n-3} are known. Here, all operations are done in $GF(2)$. We consider the recurrence relation:

$$\Delta_n = (x+d_n)\Delta_{n-1} + \Delta_{n-2} + (x+d_{n-1})\Delta_{n-3} + \Delta_{n-4}$$

Table 3. Characteristic polynomials of null boundary 5-neighborhood LHCA

Sl No.	Rule vector	Characteristic polynomial	Primitive polynomial
1	00000	$x^5 + x^3 + x$	NO
2	00001	$x^5 + x^4 + x^3 + x^2 + x$	NO
3	00010	$x^5 + x^4 + x^3 + x + 1$	YES
4	00011	$x^5 + x^2 + 1$	YES
5	00100	$x^5 + x^4 + x^3 + x^2 + x + 1$	NO
6	00101	$x^5 + x + 1$	NO
7	00110	$x^5 + x^2$	NO
8	00111	$x^5 + x^4 + x^2 + x + 1$	YES
9	01000	$x^5 + x^4 + x^3 + x + 1$	YES
10	01001	$x^5 + x^2 + x + 1$	NO
11	01010	$x^5 + x$	NO
12	01011	$x^5 + x^4 + 1$	NO
13	01100	$x^5 + x^2$	NO
14	01101	$x^5 + x^4 + x^2$	NO
15	01110	$x^5 + x^4 + x + 1$	NO
16	01111	$x^5 + x^3 + x + 1$	NO
17	10000	$x^5 + x^4 + x^3 + x^2 + x$	NO
18	10001	x^5	NO
19	10010	$x^5 + x^2 + x + 1$	NO
20	10011	$x^5 + x^4 + x^2 + x$	NO
21	10100	$x^5 + x + 1$	NO
22	10101	$x^5 + x^4$	NO
23	10110	$x^5 + x^4 + x^2$	NO
24	10111	$x^5 + x^3 + x^2 + x + 1$	YES
25	11000	$x^5 + x^2 + 1$	YES
26	11001	$x^5 + x^4 + x^2 + x$	NO
27	11010	$x^5 + x^4 + 1$	NO
28	11011	$x^5 + x^3 + x$	NO
29	11100	$x^5 + x^4 + x^2 + x + 1$	YES
30	11101	$x^5 + x^3 + x^2 + x + 1$	YES
31	11110	$x^5 + x^3 + x + 1$	NO
32	11111	$x^5 + x^4 + x^3 + x^2 + x + 1$	NO

0-Rule R_0; 1-Rule R_1

Now, we follow the Table 4. In the step 1, Δ_n and Δ_{n-1} are known. By the polynomial division algorithm, considering Δ_n as dividend and Δ_{n-1} as divisor, the degree 1 quotient polynomial $(x + d_n)$ is uniquely determined and easily calculated; since, the remainder polynomial in the relation (i.e. $\Delta_{n-2} + (x + d_{n-1})\Delta_{n-3} + \Delta_{n-4}$) is of degree less than $n-1$. In the step 2, Δ_n, Δ_{n-1}, Δ_{n-2} and Δ_{n-3} are known. In the above relation, the polynomial $\Delta_n + (x+d_n)\Delta_{n-1} + \Delta_{n-2}$ is of degree $n-2$. Now, if the polynomial division algorithm is again applied considering $\Delta_n + (x + d_n)\Delta_{n-1} + \Delta_{n-2}$ as dividend and Δ_{n-3} as divisor then, it will calculate $(x + d_{n-1})$ as quotient and Δ_{n-4} as remainder from the above relation. In the step 3, we consider the relation:

$$\Delta_{n-1} = (x + d_{n-1})\Delta_{n-2} + \Delta_{n-3} + (x + d_{n-2})\Delta_{n-4} + \Delta_{n-5}$$

Now, Δ_{n-1}, Δ_{n-2}, Δ_{n-3} and Δ_{n-4} are known and $(x+d_{n-1})$ is also known as it is computed in the previous step. If we apply the division algorithm considering $\Delta_{n-1} + (x+d_{n-1})\Delta_{n-2} + \Delta_{n-3}$ as dividend and Δ_{n-4} as divisor, it can calculate $(x + d_{n-2})$ as quotient and Δ_{n-5} as remainder from the above relation. In this way, if we proceed for n steps, then we get the sequence of degree 1 quotient polynomials as follows:

$$[(x + d_n), (x + d_{n-1}), (x + d_{n-2}), \cdots, (x + d_2), (x + d_1)]$$

where $d_k (1 \leq k \leq n)$ is either 0 or 1. By taking the constant terms of these quotient polynomials and reversing, we get the rule vector $[d_1, d_2, \cdots, d_n]$ for a 5-neighborhood LHCA with the characteristic polynomial Δ_n. The total number of polynomial divisions performed is O(n), where, n is degree of the characteristic polynomial Δ_n of n-bit CA. Each polynomial division needs O(n^2) time. Therefore, the required time complexity for this algorithm is O(n^3).

3.2 Randomness of 5-Neighborhood Linear CA Rule Vectors

A statistical test suite is developed by National Institute of Standards and Technology (NIST) that is known as NIST-statistical test suite [1]. The NIST Test Suite is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. To test the randomness of 5-neighborhood linear CA rule vectors, we consider a 24-bit 5-neighborhood maximum period LHCA with rule vector

$$[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1]$$

where $d_i = 0$ in the rulevector $[d_0, \cdots, d_{23}]$ represents that i^{th} cell of the CA follows rule R_0 and $d_i = 1$ in the rulevector $[d_0, \cdots, d_{23}]$ represents that i^{th} cell of the CA follows rule R_1. 100 bit-streams with each stream of 1,00,000 bits are generated from the middle cell ($12^{th} cell$) of this 24-bit LHCA and stored in a data file, and then the data file is fed to NIST test suite. The generated bit-streams show high randomness property as depicted in Table 5.

3.3 Diffusion Property of 5-Neighborhood Linear CA Rule Vectors

To test the diffusion property of 5-neighborhood linear CA rule vectors, we consider a 24-bit 5-neighborhood maximum period LHCA $[s_0, \cdots, s_{23}]$ with the same rule vector

Table 4. Synthesis of 5-neighborhood linear CA

Step	Known quotient	Known poly, subpoly	Relation used	Evaluated quotient	Evaluated sub-poly
1	—	Δ_n, Δ_{n-1}	$\Delta_n = (x+d_n)\Delta_{n-1} + \Delta_{n-2} + (x+d_{n-1})\Delta_{n-3} + \Delta_{n-4}$	$x+d_n$	—
2	$x+d_n$	$\Delta_n, \Delta_{n-1}, \Delta_{n-2}, \Delta_{n-3}$	$\Delta_n = (x+d_n)\Delta_{n-1} + \Delta_{n-2} + (x+d_{n-1})\Delta_{n-3} + \Delta_{n-4}$	$x+d_{n-1}$	Δ_{n-4}
3	$x+d_{n-1}$	$\Delta_{n-1}, \Delta_{n-2}, \Delta_{n-3}, \Delta_{n-4}$	$\Delta_{n-1} = (x+d_{n-1})\Delta_{n-2} + \Delta_{n-3} + (x+d_{n-2})\Delta_{n-4} + \Delta_{n-5}$	$x+d_{n-2}$	Δ_{n-5}
4	$x+d_{n-2}$	$\Delta_{n-2}, \Delta_{n-3}, \Delta_{n-4}, \Delta_{n-5}$	$\Delta_{n-2} = (x+d_{n-2})\Delta_{n-3} + \Delta_{n-4} + (x+d_{n-3})\Delta_{n-5} + \Delta_{n-6}$	$x+d_{n-3}$	Δ_{n-6}
⋮	⋮	⋮	⋮	⋮	⋮
n-3	$x+d_5$	$\Delta_5, \Delta_4, \Delta_3, \Delta_2$	$\Delta_5 = (x+d_5)\Delta_4 + \Delta_3 + (x+d_4)\Delta_2 + \Delta_1$	$x+d_4$	Δ_1
n-2	$x+d_4$	$\Delta_4, \Delta_3, \Delta_2, \Delta_1, \Delta_0$	$\Delta_4 = (x+d_4)\Delta_3 + \Delta_2 + (x+d_3)\Delta_1 + \Delta_0$	$x+d_3$	—
n-1	$x+d_3$	$\Delta_3, \Delta_2, \Delta_1, \Delta_0, \Delta_{-1}$	$\Delta_3 = (x+d_3)\Delta_2 + \Delta_1 + (x+d_2)\Delta_0 + \Delta_{-1}$	$x+d_2$	—
n	—	Δ_1, Δ_0	$\Delta_1 = (x+d_1)\Delta_0$	$x+d_1$	—

Table 5. Results of NIST-statistical test suite

Sl. No	Test name	P-value	Status
1	Frequency test	0.883171	Pass
2	BlockFrequency (block len.=128)	0.851383	Pass
3	Cumulative sums	0.574903	Pass
4	Runs	0.383827	Pass
5	Longest run	0.867692	Pass
6	FFT	0.401199	Pass
7	Non-OverlappingTemplate (block len.=9)	0.474986	Pass
8	OverlappingTemplate (block len.=9)	0.066882	Pass
9	ApproximateEntropy (block len.=10)	0.798139	Pass
10	Random excursions test	0.350485	Pass
11	Random excursions variant Test	0.534146	Pass

$$[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,1]$$

as considered in the previous section, and some CA initial values and we notice the status of the CA cells in some clock cycles. The result of the CA states for some clock cycles is depicted in Table 6. The result shows that the diffusion rate of CA cell contents is 2 times faster than 3-neighborhood CA. For the sake of simplicity, the rule value of the CA is given in hexadecimal notation i.e. a CA rule value $0xA5$ denotes the rule vector $[1,0,1,0,0,1,0,1]$ and a CA initial value $0xA5$ denotes the CA value [10100101].

Table 6. Diffusion of 5-neighborhood LHCA rule vector

	CA initial (in Hex)	Remarks
Average case	000800	12^{th} cell bit is diffused to MSB/LSB in 6/7 clock cycles, respectively.
	001000	11^{th} cell bit is diffused to MSB/LSB in 11/6 clock cycles, respectively.
Worst case	800000	0^{th} cell bit is diffused to LSB in 16 clock cycles
	000001	23^{rd} cell bit is diffused to MSB in 16 clock cycles

Table 7. Comparison of 5-neighborhood linear CA with 3/4 neighborhood CA

Properties	3-neighborhood LHCA	4-neighborhood LHCA	5-neighborhood LHCA
State transition function of i^{th} cell	[a] $s_i^{t+1} = f_i(s_{i-1}^t, s_i^t, s_{i+1}^t)$	$s_i^{t+1} = f_i(s_{i-1}^t, s_i^t, s_{i+1}^t, s_{i+2}^t)$ or $s_i^{t+1} = f_i(s_{i-2}^t, s_{i-1}^t, s_i^t, s_{i+1}^t)$	$s_i^{t+1} = f_i(s_{i-2}^t, s_{i-1}^t, s_i^t, s_{i+1}^t, s_{i+2}^t)$
# of linear rules (neighborhood radius at most r, r=3,4,5)	$2^3 = 8$	$2^4 = 16, 2^4 = 16$	$2^5 = 32$
# of linear rules (neighborhood radius exactly r, r=3,4,5)	$2^1 = 2$	$2^2 = 4, 2^2 = 4$	$2^3 = 8$
Rules combinations (with largest no. of max period CA rule vectors)	< Rule 90, Rule 150 >	$< s_i^{t+1} = s_{i-1}^t \oplus s_{i+1}^t \oplus s_{i+2}^t$, $s_i^{t+1} = s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t \oplus s_{i+2}^t >$ or $< s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_{i+1}^t$, $s_i^{t+1} = s_{i-2}^t \oplus s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t >$	$< R_0, R_1 >$[b]
Diffusion rate of n-bit CA (Average case)	At least $n/2$ clock cycles	At least $n/4$ clock cycles	At least $n/4$ clock cycles
Diffusion rate of n-bit CA (Worst case)	At most $(n-1)$ clock cycles	At most $(n-1)$ clock cycles	At most $3n/4$ clock cycles

a s_i^{t+1} denotes the state of the i-th cell at time step t+1

b Rules R_0, R_1 are defined in Sect. 3.

3.4 Comparison of Properties of 5-Neighborhood Linear CA with 3/4 Neighborhood Linear CA

In this section, we study the comparison of properties of 5-neighborhood linear CA with 3/4 neighborhood linear CA, shown in Table 7. Delay will obviously increase for 5-neighborhood CA with respect to 3-neighborhood CA. On the other hand, one clock cycle period is at least the time period required for one time CA evolving and the average diffusion rate for 5-neighborhood CA is 2 times faster than 3-neighborhood CA. Therefore, because of high diffusion rate, 5-neighborhood CA is also suitable for high speed application.

4 Conclusion

In this paper, we have studied 5-neighborhood null boundary linear CA with two linear rules. The characteristic polynomial has been realized from 5-neighborhood rule vector of the CA. We have presented an algorithm for synthesizing the 5-neighborhood CA from its characteristic polynomial by assuming some CA sub-polynomials. We have shown the randomness and diffusion properties of the 5-neighborhood CA rule vectors and the comparison of their properties with 3/4 neighborhood CA. At present, we are working on how the CA can be synthesized from its characteristic polynomial without the knowledge of CA sub-polynomials.

References

1. NIST SP 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. U.S. Department of Commerce (2010)
2. Bardell, P.H.: Analysis of cellular automata used as pseudorandom pattern generators. In: Proceedings IEEE International Test Conference 1990, Washington, D.C., USA, 10–14 September 1990, pp. 762–768 (1990)
3. Bouganim, L., Guo, Y.: Database Encryption in Encyclopedia of Cryptography and Security, 2nd edn. Springer, New York (2010)
4. Chaudhuri, P.P., Roy Chowdhury, D., Nandi, S., Chattopadhyay, S.: Additive Cellular Automata: Theory and Applications. IEEE Computer Socity Press, New York (1997)
5. la Guaz-Martinez, D., Fuster-Sabater, A.: Cryptographic design based on cellular automata In: Proceedings of IEEE International Symposium on Information Theory, p. 180 (1997)
6. Ikenaga, T., Ogura, T.: Real-time morphology processing using highly parallel 2-D cellular automata. IEEE Trans. Image Process. **9**(12), 2018–2026 (2000)
7. Jose, J., Roy Chowdhury, D.: Four neighbourhood cellular automata as better cryptographic primitives. IACR Cryptology ePrint Archive 2015, 700 (2015)
8. Kumar, K.J.J., Sudharsan, S., Karthick, V.: FPGA implementation of cellular automata based stream cipher: Yugam-128. IJAREEIE **3** (2014)
9. Tomassini, M., Sipper, M., Perrenoud, M.: On the generation of high-quality random numbers by two-dimensional cellular automata. IEEE Trans. Comput. **49**, 1146–1151 (2000)

10. Matsumoto, M.: Simple cellular automata as pseudorandom m-sequence generators for built-in self-test. ACM Trans. Model. Comput. Simul. **8**(1), 31–42 (1998)
11. Nandi, S., Kar, B.K., Chaudhuri, P.P.: Theory and applications of cellular automata in cryptography. IEEE Trans. Comput. **43**(12), 1346–1357 (1994)
12. Neumann, J.V.: The Theory of Self- Reproducing Automata. University of Illinois Press Urbana (1966). (Edited by Burks, A.W.)
13. Roy Chowdhury, D., Sengupta, I., Chaudhuri, P.P.: A class of two-dimensional cellular automata and their applications in random pattern testing. J. Electron. Test. **5**(1), 67–82 (1994)
14. Sudhakar, P., Chinnarao, B., Latha, D.M.M.: Optimization of 1D and 2D cellular automata for pseudo random number generator. IOSR J. VLSI Sig. Proc. (IOSR-JVSP) **4**, 28–33 (2014)
15. Wolfram, S.: Theory and Applications of Cellular Automata (Including Selected Papers 1983–1986). World Scientific Pub. Co., Inc., River Edge (1986)