# Do There Exist Non-linear Maximal Length Cellular Automata? A Study

Sumit Adak[1(✉)], Sukanya Mukherjee[2], and Sukanta Das[1]

[1] Department of Information Technology,
Indian Institute of Engineering Science and Technology, Shibpur 711103, India
{sumitadak,sukanta}@it.iiests.ac.in
[2] Department of Computer Science and Engineering,
Institute of Engineering and Management, Kolkata 700091, India
sukanya.mukherjee@iemcal.com

**Abstract.** An $n$-cell maximal length cellular automaton (CA) is a binary CA which is having a cycle of length $2^n - 1$. These CAs are linear and have been used in different applications, such as pseudo random number generation, VLSI design & test, cryptosystem etc. For some applications, however, it could be good if we can use non-linear maximal length CAs. In this paper, we arrange an experiment for the search of non-linear maximal length CAs. By experimentation, we have seen that there exists non-linear maximal length CAs.

**Keywords:** Non-linear cellular automata · Reversible
Maximal length · Configuration · Rule

## 1 Introduction

The cellular automata (CAs) that generate large cycles are highly useful in computational processes like pseudo random number generator (PRPG) [5,7], cryptosystem [3,6] etc. Prior works have considered the use of linear *maximal length* cellular automata [1,2] for such applications, where the cycle length is as large as $2^n - 1$ for an $n$-cell binary cellular automaton (CA). Linear maximal length CAs, however, suffer from some drawbacks. Firstly, the availability of $n$-degree primitive polynomial is limited. Besides, linear maximal length sequences are not secure. So, there is a necessity of a construction that can provide both non-linearity and maximal length sequence for optimized crypto-system (see [3,6] for details).

There have been some researches to introduce non-linearity in maximal length CAs [3,6]. The technique referred in [3] manipulates the number of clock cycles, based on inputs, in a maximum length additive CA. This method becomes unsynchronized for different inputs. An efficient technique [6] is devised for generating non-linear maximal length CA from linear maximal length CA by injecting non-linearity in different cell positions. The effect of the non-linearity can be propagated among multiple cells by shifting the non-linear function. However, it


G. Mauri et al. (Eds.): ACRI 2018, LNCS 11115, pp. 289–297, 2018.
https://doi.org/10.1007/978-3-319-99813-8_26

incurs increasing neighborhood dependency. For optimal design, the construction of non-linear maximal length CA limits upto 5 neighborhood. This motivates us to figure out if there exists a non-linear maximal length CA without exceeding the neighborhood dependency. In this paper we answer this question in an affirmative way.

## 2 Basics

### 2.1 Definitions

A 1-d finite CA of size $n$ consists of an array of $n$ cells. Each cell can be in either of two *states*, 0 or 1 as we use binary CA. Let $x_i$ denote the state of cell $i$. Then, a configuration of the CA is $x = (x_0 x_1 \cdots x_{n-1})$ where $x_i \in \{0, 1\}$. In this work, we consider null boundary CA, which means $x_{-1} = x_n = 0$. Cell $i$ of the CA changes its state at every time step following a next state function $f_i : \{0, 1\}^3 \mapsto \{0, 1\}$, which is defined over the present states of cell $i$ and its left and right neighbors. Let us denote $\mathcal{C}$ as the set of all possible configurations of the CA. The CA thus can be interpreted as a function $F : \mathcal{C} \to \mathcal{C}$, which satisfies the following conditions: $y = F(x)$, $x, y \in \mathcal{C}$, where $y = (y_i)_{0 \le i \le n-1}$ and $y_i = f_i(x_{i-1}, x_i, x_{i+1})$.

There can be eight possible combinations depending on the present states of a cell and its two neighbours. The next state for the cell for each of these combinations depends on the next state function. Thus there can be $2^8$ distinct next state functions, and each next state function can be associated to a value between 0 and 255, which we call *rule*. The rule corresponding to a particular next state function is obtained as the decimal equivalent of next state generated for the eight combinations of the present states $x_{i-1}$, $x_i$ and $x_{i+1}$ (as shown in Table 1). For a particular rule $\mathcal{R}$, let $\mathcal{R}[x_{i-1}x_i x_{i+1}]$ denote the next state of cell $i$ for the present states combination $x_{i-1}x_i x_{i+1}$ of cell $i$ and its neighbours. For example, $30[011] = 1$. Thus a CA can alternatively be interpreted as a *rule vector* $\mathcal{R} = (\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1})$, where each $\mathcal{R}_i$ is the rule to which $f_i$ is associated. The uniform CA is a special case where $\mathcal{R}_0 = \mathcal{R}_1 = \cdots = \mathcal{R}_i = \cdots = \mathcal{R}_{n-1}$. If an $\mathcal{R}_i$ of an $n$-cell CA is said to be *linear* if its corresponding $f_i$ follows XOR logic.

**Definition 1** *If all the rules of a rule vector $\mathcal{R}$ are linear/additive, then the CA is linear/additive.*

Here, we consider only seven rules as linear – 60, 90, 102, 150, 170, 204 and 240. Another seven rules (15, 51, 85, 105, 153, 165 and 195) are complemented additive rules.

**Definition 2.** *If any $\mathcal{R}_i$ of $\mathcal{R}$ is not linear/ additive, then CA is non-linear.*

**Definition 3.** *A configuration $x \in \mathcal{C}$ is said to be cyclic if $x = F^t(x)$ for some finite $t \in \mathbb{N}$.*

**Definition 4.** *A CA is reversible if all the configurations are cyclic.*

**Table 1.** Rules 90, 150, 54 and 30

| Present state | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 | Rule |
|---|---|---|---|---|---|---|---|---|---|
| (i) Next state | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 90 |
| (ii) Next state | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 150 |
| (iii) Next state | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 54 |
| (iv) Next state | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 30 |

### 2.2 Synthesis of Reversible CAs

Synthesis of a reversible CA given in [4]. Here we briefly present the methodologies for sake of completeness.

Only 62 out of the 256 possible rules are used to form non-uniform reversible CAs. These rules can be classified into different classes as shown in Tables 2, 3 and 4. We now state how a reversible CA can be generated from these tables. The rule at the cell zero i.e., $\mathcal{R}_0$ is selected out of the rules given in first column of Table 2. Note that the first and last rules of a null boundary CA are to be chosen differently (see [4] for details). However, the selected rule at the cell zero defines the class (second column of Table 2) from which $\mathcal{R}_1$ has to be selected. For every $i$ between 1 and $n-2$, the first column of Table 4 shows the probable classes of rule $\mathcal{R}_i$; the second column shows the possible rules for rule $\mathcal{R}_i$ from each class, while corresponding to a particular rule $\mathcal{R}_i$, the third column defines the class from which $\mathcal{R}_{i+1}$ has to be selected. Finally, depending on the class of the rule at the last cell, the rule $\mathcal{R}_{n-1}$ is selected from Table 3.

*Example 1.* Let us consider a 4-cell CA $(10, 150, 90, 20)$ which is reversible. It is obtained as follows: here, $\mathcal{R}_0$ is 10. The class of $\mathcal{R}_1$ is II (see Table 2). Thus the rule at cell 1 must be selected from the row corresponding to class II of Table 4. In particular, let $\mathcal{R}_1$ is selected to be 150. $\mathcal{R}_2$ should be selected from class I, as the class for selecting the next rule corresponding to rule 150 is class I (see last column of Table 4). Let $\mathcal{R}_2$ is selected as 90. Applying the same methodology, the class of rule $\mathcal{R}_3$ comes out to be class II. From Table 3, the rule $\mathcal{R}_3$ is selected from the row corresponding to class II, in particular $\mathcal{R}_3$ is selected to be 20.

**Definition 5.** *An $n$-cell CA is maximal length if, for a configuration $x \in \mathcal{C}$, $x = F^{2^n-1}(x)$, but $x \neq F^t(x)$ where $1 \leq t < 2^n - 1$.*

The CA $(10, 150, 90, 20)$ is equivalent to the CA $(90, 150, 90, 150)$ when the boundary condition is null. That is, this CA is linear. Further, it is a maximal length CA (see Fig. 1).

## 3 Cellular Automata with Large Cycles

Maximal length CAs are having the largest possible cycle length for given CA size $n$. In this section we develop a process to design CAs which are expected

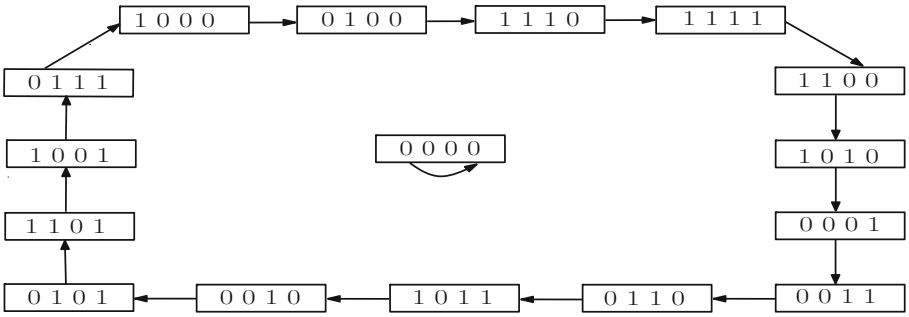


**Fig. 1.** Configuration transition diagram of the CA $(10, 150, 90, 20)$

**Table 2.** First rule table

| Rules for $\mathcal{R}_0$ | Class of $\mathcal{R}_1$ |
|---|---|
| 3, 12 | I |
| 5, 10 | II |
| 6, 9 | III |

**Table 3.** Last rule table

| Rule class for $\mathcal{R}_{n-1}$ | Rule set for $\mathcal{R}_{n-1}$ |
|---|---|
| I | 17, 20, 65, 68 |
| II | 5, 20, 65, 80 |
| III | 5,17, 68, 80 |
| IV | 20, 65 |
| V | 17, 68 |
| VI | 5, 80 |

to have large cycles. If non-linear maximal length CAs really exist, we can get such CAs by repeatedly applying this process.

We first intuitively present the idea behind our approach. Clearly, a cell $i$ changes its state in next time step depending on the present states of itself and its neighbours. If the cells of a CA does not depend on their neighbors, the CA cannot produce large cycles. For example, in the extreme case, if $f_i(x_{i-1}, x_i, x_{i+1}) = x_i$ for all $i$, then every cycle is of length one. Similarly, if $f_i(x_{i-1}, x_i, x_{i+1}) = 1 - x_i$ for all $i$, every cycle is of length two. Thus lower the dependency of the next state function on the present state of the neighbours of a cell, smaller will be the length of the cycles generated by the corresponding CA. Conversely, if the next state of a cell is more *influenced* by the state of its neighbours, greater is the chance of obtaining a large length cycle.

Let $(x_0 x_1 \cdots x_i \cdots x_{n-1})$ denote a configuration of the CA. Suppose $f_i(x_{i-1}, x_i, x_{i+1}) = f_i(x_{i-1}, x_i, 1 - x_{i+1})$, for all values of $x_i$ and $x_{i-1}$. This implies that the next state of cell $i$ is not influenced by the present state of cell $i + 1$; we say that cell $i$ is *independent* of its right neighbor. In an analogous manner, if $f_i(x_{i-1}, x_i, x_{i+1}) = f_i(1 - x_{i-1}, x_i, x_{i+1})$ for all values of $x_i$ and $x_{i+1}$, cell $i$ is independent of its left neighbor.

We can define the degree of dependence on the neighbor of a cell as follows. Let $\alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y})$ denote the dependence of cell $i$ on its right neighbor when the present states of $x_{i-1}$ and $x_i$ are respectively `x` and `y`. Note that each of `x` and `y` can be either 0 or 1.

**Table 4.** Class relationship of $R_i$ and $R_{i+1}$

| Class of $R_i$ | $R_i$ | Class of $R_{i+1}$ |
|---|---|---|
| I | 51, 204, 60, 195 | I |
| | 85, 90, 165, 170 | II |
| | 102, 105, 150, 153 | III |
| | 53, 58, 83, 92, 163, 172, 197, 202 | IV |
| | 54, 57, 99, 108, 147, 156, 198, 201 | V |
| | 86, 89, 101, 106, 149, 154, 166, 169 | VI |
| II | 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240 | I |
| III | 51, 204, 15, 240 | I |
| | 85, 105, 150, 170 | II |
| | 90, 102, 153, 165 | III |
| | 23, 43, 77, 113, 142, 178, 212, 232 | IV |
| | 27, 39, 78, 114, 141, 177, 216, 228 | V |
| | 86, 89, 101, 106, 149, 154, 166, 169 | VI |
| IV | 60, 195 | I |
| | 90, 165 | IV |
| | 105, 150 | V |
| V | 51, 204 | I |
| | 85, 170 | II |
| | 102, 153 | III |
| | 86, 89, 90, 101, 105, 106, 149, 150, 154, 165, 166, 169 | VI |
| VI | 15, 240 | I |
| | 105, 150 | IV |
| | 90, 165 | V |

$$\alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y}) = \begin{cases} 1 & \text{if } f_i(\mathtt{x}, \mathtt{y}, x_{i+1}) \neq f_i(\mathtt{x}, \mathtt{y}, 1 - x_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

The degree of dependence of cell $i$ on its right neighbor is the ratio of the number of combinations of values of $x_i$ and $x_{i-1}$ for which the next state function on $x_i$ depends on $x_{i-1}$. This is called the *degree of right dependence* for rule $\mathcal{R}_i$ and denoted by $\mathtt{P_r}(\mathcal{R}_i)$. Clearly, $\mathtt{P_r}(\mathcal{R}_i)$ can take values 0, 0.5 or 1. Formally,

$$\mathtt{P_r}(\mathcal{R}_i) = \frac{\sum_{\mathtt{x}\in\{0,1\}} \sum_{\mathtt{y}\in\{0,1\}} \alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y})}{4}$$

Similarly, let $\alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y})$ denote the dependence of cell $i$ on its left neighbor when the present states of $x_i$ and $x_{i+1}$ are respectively $\mathtt{x}$ and $\mathtt{y}$.

$$\alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y}) = \begin{cases} 1 & \text{if } f_i(x_{i-1}, \mathtt{x}, \mathtt{y}) \neq f_i(1 - x_{i-1}, \mathtt{x}, \mathtt{y}) \\ 0 & \text{otherwise} \end{cases}$$

In an analogous way, we define the parameter $\mathtt{P_l}$ which determines how much a cell $i$ depends on its left neighbor. It is the ratio of the number of combinations of values of $x_i$ and $x_{i+1}$ for which the next state function on $x_i$ depends on $x_{i+1}$. This is called the *degree of left dependence* for rule $\mathcal{R}_i$, and denoted by $\mathtt{P_l}(\mathcal{R}_i)$.

$$\mathtt{P_l}(\mathcal{R}_i) = \frac{\sum_{\mathtt{x}\in\{0,1\}} \sum_{\mathtt{y}\in\{0,1\}} \alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y})}{4}$$

*Example 2.* Let us consider rule 54. We observe that for the next state function corresponding to this rule, $\alpha_{rd}(x_{i-1} = \mathtt{0}, x_i = \mathtt{0}) = 1, \alpha_{rd}(x_{i-1} = \mathtt{0}, x_i = \mathtt{1}) = 1$, while $\alpha_{rd}(x_{i-1} = \mathtt{1}, x_i = \mathtt{0}) = 0, \alpha_{rd}(x_{i-1} = \mathtt{1}, x_i = \mathtt{1}) = 0$. Therefore, $\mathtt{P_r}(\mathtt{54})$ is 0.5. On the other hand, $\alpha_{ld}(x_i = \mathtt{0}, x_{i+1} = \mathtt{0}) = 1, \alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{0}) = 1$, while $\alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{1}) = 0, \alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{0}) = 0$. Therefore, $\mathtt{P_l}(54)$ is 0.5. Similarly, for rules 90 and 60, $\mathtt{P_r}(90) = 1$ and $\mathtt{P_r}(60) = 0$. For rules 150 and 170, we get $\mathtt{P_l}(150) = 1$ and $\mathtt{P_l}(170) = 0$.

The rules of reversible CAs can be classified into three categories depending on $\mathtt{P_r}$ and $\mathtt{P_l}$ parameter. The three categories are named as *completely right dependent*, *partially right dependent* and *right independent*, and they correspond respectively to right dependence degree values of 0, 0.5 and 1. In null boundary condition, all possible inputs to first and last rules are not valid. So, we need to classify first and last rules separately using the same process stated above.

In order to have a CA generates a cycle of $2^n - 1$ length, it is desirable to have the rules of the CA dependent on both the left and the right neighbours. The degree of dependence of a rule $\mathcal{R}_i$ on both of its neighbours can be determined by the product of $\mathtt{P_r}(\mathcal{R}_i)$ and $\mathtt{P_l}(\mathcal{R}_i)$, and we denote this by $\mathtt{P}(\mathcal{R}_i)$.

$$\mathtt{P}(\mathcal{R}_i) = \mathtt{P_r}(\mathcal{R}_i) * \mathtt{P_l}(\mathcal{R}_i).$$

Clearly, $\mathtt{P}(\mathcal{R}_i)$ can take values 0, 0.25, 0.5 or 1. We can thus classify the rules here into four categories based on the $\mathtt{P}$ parameter. As shown in Table 5, any rule can correspond to either of the four categories *completely dependent*, *partially dependent*, *weakly dependent* and *independent* depending on the $\mathtt{P}$ values of 1, 0.5, 0.25 and 0 respectively. However, to obtain a large cycle, we generate the corresponding CA by selecting rules from Table 5 as follows.

The first and the last rules of every CA are selected uniformly at random from the class of *completely dependent*. For the remaining, we pick $n - 2$ rules randomly following Gaussian distribution in such a way that the maximum rules are selected from the category of *completely dependent*, some selected from the category of *partially dependent*, and a very few from the category of *weakly dependent*. Since most of the rules, selected in this manner, have high degree of dependence on both of their neighbours, it is highly likely that the corresponding CA will have a large cycle.

**Table 5.** Four categories of reversible CA rules on the parameter P

| Category | $\mathcal{R}_i$ | $\mathcal{R}_0$ | $\mathcal{R}_{n-1}$ |
|---|---|---|---|
| Completely dependent | 90, 165, 150, 105 | 5, 6, 9, 10 | 5, 20, 65, 80 |
| Partially dependent | 30, 45, 75, 120, 135, 180, 210, 225, 86, 89, 101, 106, 149, 154, 166, 169 | | |
| Weakly dependent | 53, 58, 83, 92, 163, 172, 197, 202, 54, 57, 99, 108, 147, 156, 198, 201, 23, 43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228, | | |
| Independent | 51, 204, 85, 170, 102, 153, 60, 195, 15, 240 | 3, 12 | 17, 68 |

**Table 6.** Cycles are close to $2^k - 1$ for $k$-cell CA (Here $k = 10$)

| Cycle length | 10-cell CA |
|---|---|
| 1015 | (9, 90, 43, 150, 166, 90, 165, 150, 90, 65) |
| 923 | (9, 166, 105, 105, 101, 150, 150, 105, 150, 20) |
| 801 | (9, 86, 90, 149, 105, 90, 165, 165, 90, 65) |
| 1008 | (10, 165, 86, 150, 165, 90, 105, 90, 150, 65) |
| **1023** | **(10, 90, 150, 169, 165, 101, 150, 90, 165, 20)** |
| 1001 | (10, 90, 53, 90, 90, 150, 89, 90, 105, 80) |
| 1003 | (10, 105, 90, 150, 57, 150, 90, 105, 150, 65) |
| 761 | (5, 90, 165, 180, 154, 165, 106, 165, 90, 80) |
| 1000 | (10, 165, 86, 90, 101, 165, 105, 105, 165, 65) |
| 920 | (10, 165, 90, 150, 86, 105, 105, 90, 150, 20) |
| 1022 | (9, 106, 150, 105, 90, 105, 150, 90, 150, 65) |
| 827 | (9, 43, 105, 154, 105, 165, 150, 90, 150, 20) |
| 1017 | (9, 90, 89, 150, 165, 150, 106, 90, 89, 80) |
| 728 | (10, 90, 57, 105, 165, 101, 150, 165, 90, 65) |
| **1023** | **(6, 150, 210, 53, 150, 150, 165, 105, 150, 20)** |

## 4 Experimental Results

Using the above mentioned approach, we generate a number of CAs of different sizes. We observe that the cycles of the synthesized CAs are large, and most of the time, the largest cycle of such CAs are close to $2^n - 1$. Table 6 shows a sample result of our experiment. Bold faced rows are the non-linear maximal length CAs. This result proves that there exist non-linear maximal length CAs.

Let us now understand the percentage frequency distribution of different categories of rules which can generate non-linear maximal length by using the

above process. For CA size 10, we generate 100 non-linear maximal length CAs and using these data, we observe that 83% of the rules belong to *completely dependent* category, 13.8% belong to the *partially dependent* class, while 3.02% belong to the *weakly dependent* class.

To understand the efficacy of the above mentioned approach, we conduct experiments. We generate random non-linear maximal length reversible CAs of size $n$ extensively. Obviously, each CA follows a distribution for the rules being selected from the different categories based on `P` which mentioned already. By maintaining this distribution, we get CAs of length $2^n - 1$ of a fixed CA size $n$. We perform experiment for different values of $n$ ranges from 4 to 20. Here, by experiments, we observe that there exists a non-linear maximal length CA for any $n$. In Table 7, we shows the CAs which contributes maximal length CA for sizes 4 to 20.

From the experimental results, however, we observe that *sixteen* rules from category *weakly dependent* have not participated in the maximal length CA generation. These rules are 92, 172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228.

**Table 7.** $n$-cell non-linear maximal length CAs

| $n$ (CA size) | $\mathcal{R}$ (CA) |
|---|---|
| 4 | (6, 178, 90, 20) |
| 5 | (5, 150, 99, 165, 5) |
| 6 | (5, 90, 106, 90, 166, 5) |
| 7 | (6, 101, 90, 154, 105, 165, 65) |
| 8 | (9, 90, 105, 30, 54, 150, 105, 65) |
| 9 | (5, 180, 150, 105, 165, 149, 150, 90, 65) |
| 10 | (10, 105, 54, 154, 90, 166, 90, 86, 105, 65) |
| 11 | (5, 150, 165, 30, 58, 90, 150, 86, 105, 90, 65) |
| 12 | (6, 105, 165, 90, 180, 147, 165, 165, 105, 165, 150, 8) |
| 13 | (6, 86, 90, 169, 105, 150, 89, 90, 165, 150, 90, 90, 65) |
| 14 | (9, 177, 89, 90, 89, 90, 101, 105, 165, 90, 150, 90, 150, 65) |
| 15 | (10, 75, 90, 90, 166, 90, 86, 105, 90, 150, 166, 105, 90, 90, 20) |
| 16 | (6, 90, 178, 150, 154, 150, 105, 105, 90, 150, 150, 90, 165, 105, 90, 80) |
| 17 | (6, 165, 150, 165, 150, 150, 90, 101, 150, 165, 150, 105, 165, 169, 150, 165, 20) |
| 18 | (9, 165, 86, 150, 90, 90, 165, 150, 105, 150, 165, 150, 105, 105, 150, 149, 150, 20) |
| 19 | (10, 45, 58, 90, 165, 105, 165, 150, 165, 150, 149, 165, 90, 165, 90, 105, 105, 105, 5) |
| 20 | (10, 150, 165, 105, 149, 165, 165, 150, 150, 89, 90, 105, 105, 165, 105, 150, 150, 165, 165, 20) |

## References

1. Cattel, K., Muzio, J.C.: Synthesis of one dimensional linear hybrid cellular automata. IEEE Trans. CAD **15**, 325–335 (1996)
2. Pal Chaudhuri, P., Roy Chowdhury, D., Nandi, S., Chatterjee, S.: Additive Cellular Automata - Theory and Applications, vol. 1. IEEE Computer Society Press, Los Alamitos (1997). ISBN 0-8186-7717-1

3. Das, S., Roy Chowdhury, D.: Generating cryptographically suitable non-linear maximum length cellular automata. In: Bandini, S., Manzoni, S., Umeo, H., Vizzari, G. (eds.) ACRI 2010. LNCS, vol. 6350, pp. 241–250. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15979-4_26
4. Das, S.: Theory and applications of nonlinear cellular automata in VLSI design. Ph.D. thesis, Bengal Engineering and Science University, Shibpur, India (2007)
5. Das, S., Kundu, A., Sikdar, B.K.: Nonlinear CA based design of test set generator targeting pseudo-random pattern resistant faults. In: Proceedings of Asian Test Symposium, pp. 196–201 (2004)
6. Ghosh, S., Sengupta, A., Saha, D., Chowdhury, D.R.: A scalable method for constructing non-linear cellular automata with period $2^n - 1$. In: Wąs, J., Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 65–74. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11520-7_8
7. Wolfram, S.: Random sequence generation by cellular automata. Adv. Appl. Math **7**, 123 (1984)