Cellular Automata Pseudorandom Sequence Generation

by

Smarak Acharya
BE, Visvesvaraya Technological University, 2011

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Cellular Automata Pseudorandom Sequence Generation

by

Smarak Acharya
BE, Visvesvaraya Technological University, 2011

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Daler Rakhmatov, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Daler Rakhmatov, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Pseudorandom sequences have many applications in fields such as wireless communication, cryptography and built-in self test of integrated circuits. Maximal length sequences (m-sequences) are commonly employed pseudorandom sequences because they have ideal randomness properties like balance, run and autocorrelation. However, the linear complexity of m-sequences is poor. This thesis considers the use of one-dimensional Cellular Automata (CA) to generate pseudorandom sequences that have high linear complexity and good randomness. The properties of these sequences are compared with those of the corresponding m-sequences to determine their suitability.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

My sincere gratitude to my supervisor Dr. T. Aaron Gulliver for providing his valuable guidance and encouragement during my studies as an MASC student at the University of Victoria. His expertise and knowledge were essential for sucessfully completing my thesis. His patience and flexibility enabled me to work on my studies and thesis in a productive and independent manner. I would also like to thank all my friends and fellow students for their unfailing support professionally and personally throughout my studies.

DEDICATION

I dedicate this thesis to my parents, who have been my pillars of support through thick and thin in my life.

Chapter 1

Introduction

Randomness is characterized by a lack of pattern or predictable outcome associated with an event [1]. It has applications in art, politics, sports, science and statistics. The selection of winning numbers in the BC/49 lottery and creation of match fixtures in the English Premier League football use randomness. Random numbers have a prevalent use in the field of cryptography [3], and are typically selected as the key in encryption algorithms to ensure secure message transfers between a transmitter and a receiver. The more random the key is, the less susceptible an encrypted message is to attacks. Random numbers are used in the generation of noise for testing wireless communication systems [7] and also for Built-In Self Test (BIST) in integrated circuits [18]. Such applications employ random numbers in the form of sequences called *random sequences*. The success of these applications depends on the degree of randomness of the generated sequences. In this thesis, only binary sequences are considered.

The generation of true random sequences, is cumbersome and inefficient as it relies on physical phenomena like radioactive decay or noise in electric circuits. Instead, digital circuits are employed to produce bit sequences that appear random [2]. These sequences are called *pseudorandom sequences*. Unlike random sequences (infinite period), they have a pattern that repeats (finite period), so the same sequences are generated in a cycle after every period. Within the period, pseudorandom sequences are similar to random sequences, but to be considered useful these sequences should exhibit properties of statistical randomness such as balance, run and autocorrelation [4]. These properties are described below.

Balance: The balance property for sequences is a direct implication of the *frequency property* [2] for statistical randomness of integers. According to the frequency property, a random sequence has an equal distribution of numbers, so a random integer sequence contains the same number of 0s, 1s, 2s, ... For example, a random binary sequence contains an equal number of 0s and 1s.

Run: A run is a sequence of identical numbers. With an ideal binary sequence, 1/2 of the runs have length 1, 1/4 of the runs have length 2, 1/8 of the runs have length 3, and so on.

Autocorrelation: The periodic autocorrelation is a measure of how similar a sequence is to a delayed copy of itself [4]. It is mathematically given by

$$r(k) = \sum_{m=0}^{N-1} 1 - 2\big(s[m] \oplus s[m-k]\big) \tag{1.1}$$

where $s[k]$ is the kth bit of the sequence, N is the length of the sequence and k is an integer, $0 \leq k \leq N-1$. The autocorrelation of an ideal sequence has low values of $r(k)$ for $k \neq 0$.

Pseudorandom sequences are often generated using a Linear Feedback Shift Register (LFSR). LFSRs are comprised of flip-flops connected in series with linear combinational logic (XOR gates/mod-2 adders) in the feedback path. Figure 1.1 shows an example of an LFSR. Combinations of the mod-2 adder outputs based on primitive polynomials generate sequences with maximum period, called maximum length sequences (m-sequences) [5]. These sequences have ideal balance, run and autocorrelation properties. Another important property of pseudorandom sequences is the linear complexity which is defined as follows.

Linear Complexity: The linear complexity of a sequence is the smallest LFSR that produces the sequence [3]. An ideal sequence has a large linear complexity.

Unfortunately, m-sequences have poor linear complexity. Section 1.1 describes LFSR pseudorandom sequence generation and the randomness properties of the associated sequences.

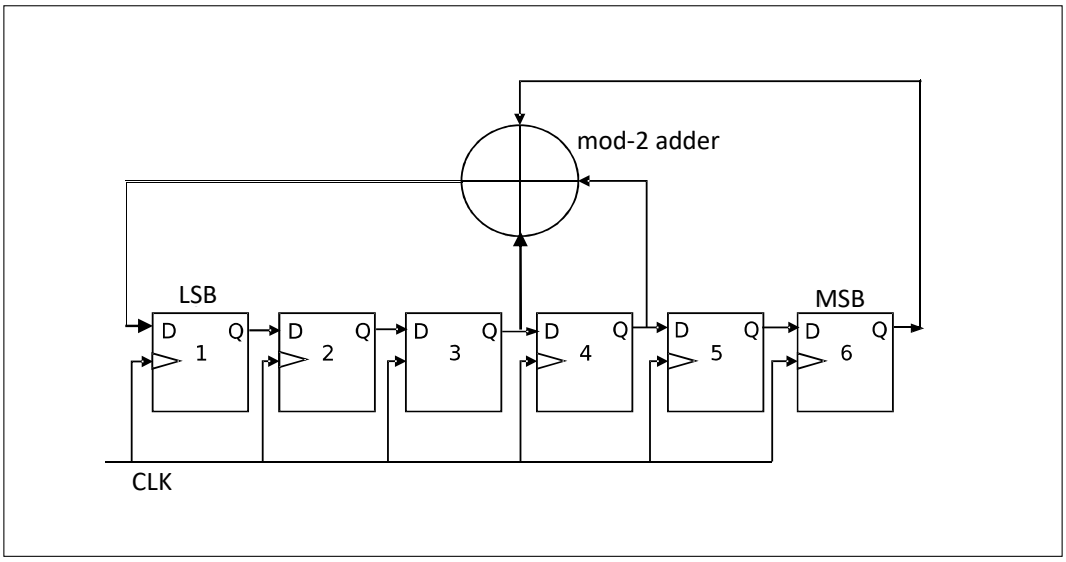

Figure 1.1: An example of a binary linear feedback shift register (LFSR).

1.1 LFSRs and m-Sequences

An LFSR is a shift register with a feedback circuit composed of linear logic. It is synchronized with an external clock. One implementation is called a *Fibonacci LFSR* [5], where the input to the leftmost flip-flop is a linear function of the outputs of one or more of flip-flops. Another implementation, called a *Galois LFSR*, has the inputs of each flip-flop being a linear function of the output of the rightmost flip-flop [5]. Figure 1.1 is an example of a Fibonacci LFSR and Figure 1.2 shows a Galois LFSR. For convenience only Fibonacci LFSRs will be examined in this thesis.

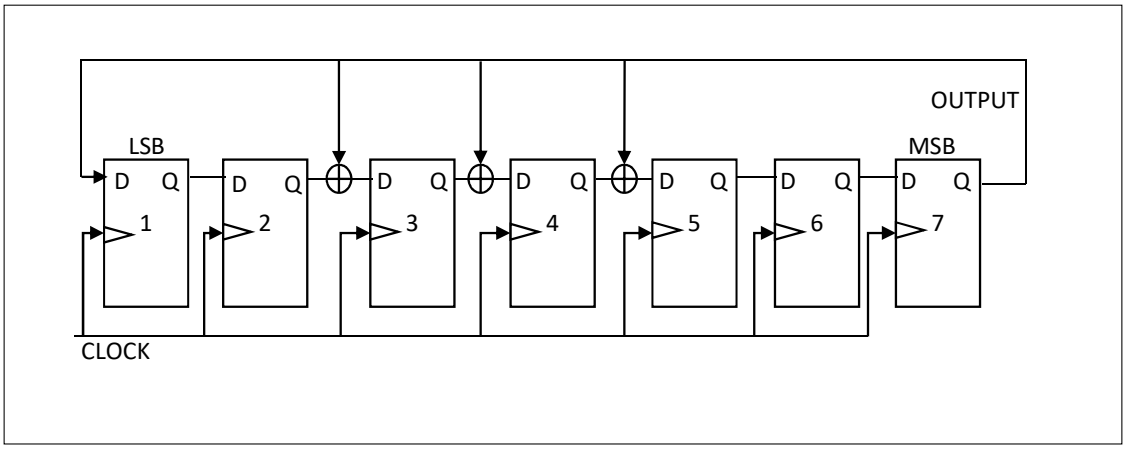

Figure 1.2: An example of a Galois LFSR.

The generic structure of an n-bit LFSR is shown in Figure 1.3. It consists of n

D flip-flops ($D1$ to Dn), n taps and a feedback circuit comprised of mod-2 adders. The taps (g_1 to g_n) can take one of two values, 0 or 1 (0 = open, 1 = closed). The output of a flip-flop is connected to the logic circuit if the corresponding tap is 1 (g_i = 1). The sequence of bits given by ($D1 \ldots Dn$) is called the *state* of the LFSR. The feedback circuit determines the next state (at the next clock pulse), based on the logic and taps. An n-bit LFSR can have 2^n possible states and the circuit will cycle through some of these states based on the feedback logic.

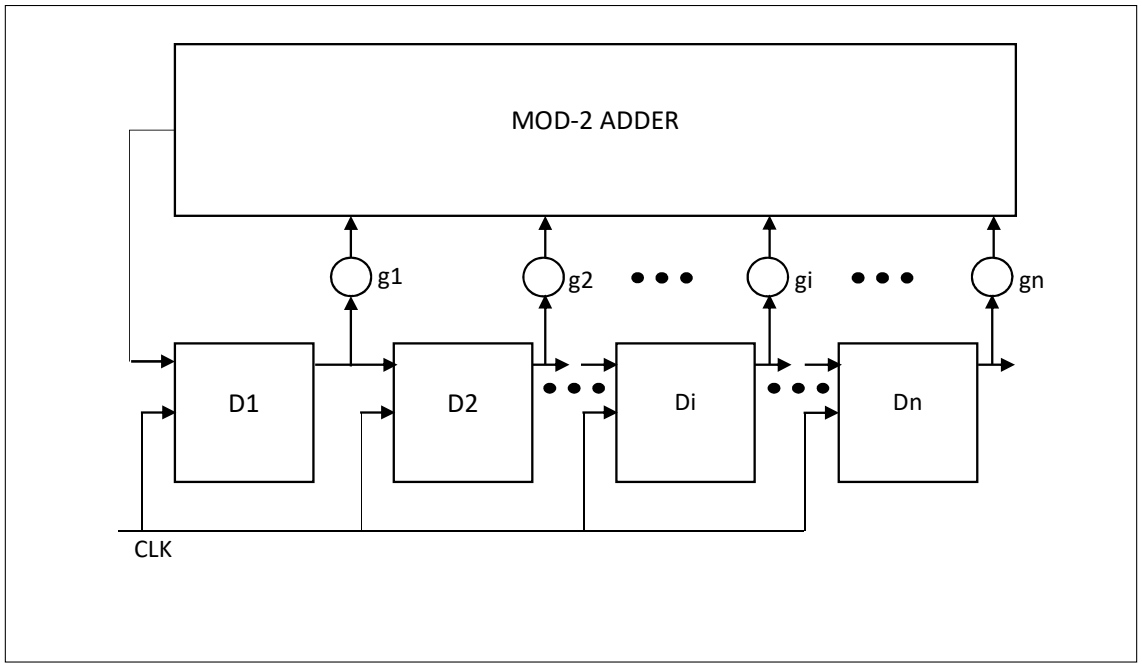

Figure 1.3: Generic n-bit LFSR.

LFSRs can be described by a state transition matrix given by [7]

$$A_{LFSR} = \begin{bmatrix} g_1 & g_2 & \ldots & g_{n-3} & g_{n-2} & g_{n-1} & 1 \\ 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.2}$$

The next state vector s_{k+1} is obtained from the present state vector s_k according to

$$s_{k+1} = A_{LFSR} s_k \tag{1.3}$$

where

$$s_{k+1} = [D1_{k+1}\ D2_{k+1}\ \ldots\ Di_{k+1}\ \ldots\ Dn-1_{k+1}\ Dn_{k+1}]^T$$

and

$$s_k = [D1_k\ D2_k\ \ldots\ D3_k\ \ldots\ Dn-1_k\ Dn_k]^T$$

A binary sequence can be obtained from the output of any flip-flop. The period of this sequence depends on the fraction of the 2^n states the circuit cycles through, and is determined by the feedback logic. Circuits that produce the maximum period are called maximum length LFSRs and the generated sequences are called m-sequences. The period of an m-sequence is $N = 2^n - 1$. This is because the all zero state is not part of the sequence of states [7], as the next state for an all zero initial state will always be all zero irrespective of the feedback logic. Hence, a maximum length LFSR with a non zero initial state will cycle through all $2^n - 1$ non-zero states before returning to the initial state.

The feedback circuit of an LFSR is associated with a *characteristic polynomial* of degree n [6]. Its generic form is $x^n + g_1x^{n-1} + g_2x^{n-2} + \ldots + g_ix^{n-i} + \ldots + g_{n-1}x + 1$. The characteristic polynomial of a maximum length LFSR is a primitive polynomial. A primitive polynomial is a irreducible polynomial that generates all the non-zero field elements of the finite field $GF(p^l)$, where p is prime and l is an integer ≥ 2. Figure 1.4 shows an example of a 4-bit maximum length LFSR [6] corresponding to the characteristic polynomial $x^4 + x^3 + 1$. The corresponding state transition matrix is

$$A_{LFSR} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.4}$$

Assuming an initial state $(D1, D2, D3, D4) = (0, 0, 1, 1)$, this LFSR generates the $2^4 - 1 = 15$ states shown in Table 1.1. The contents of $D1$ is $010001111010110\ldots$ This is an m-sequence of period 15.

The properties of m-sequences are often used as a benchmark to determine the quality of pseudorandom sequences. These properties will be used in this thesis to

Clock	State				Clock	State			
	D1	D2	D3	D4		D1	D2	D3	D4
0	0	0	1	1	8	1	1	1	1
1	1	0	0	1	9	0	1	1	1
2	0	1	0	0	10	1	0	1	1
3	0	0	1	0	11	0	1	0	1
4	0	0	0	1	12	1	0	1	0
5	1	0	0	0	13	1	1	0	1
6	1	1	0	0	14	0	1	1	0
7	1	1	1	0	15	0	0	1	1

Table 1.1: LFSR State Transitions

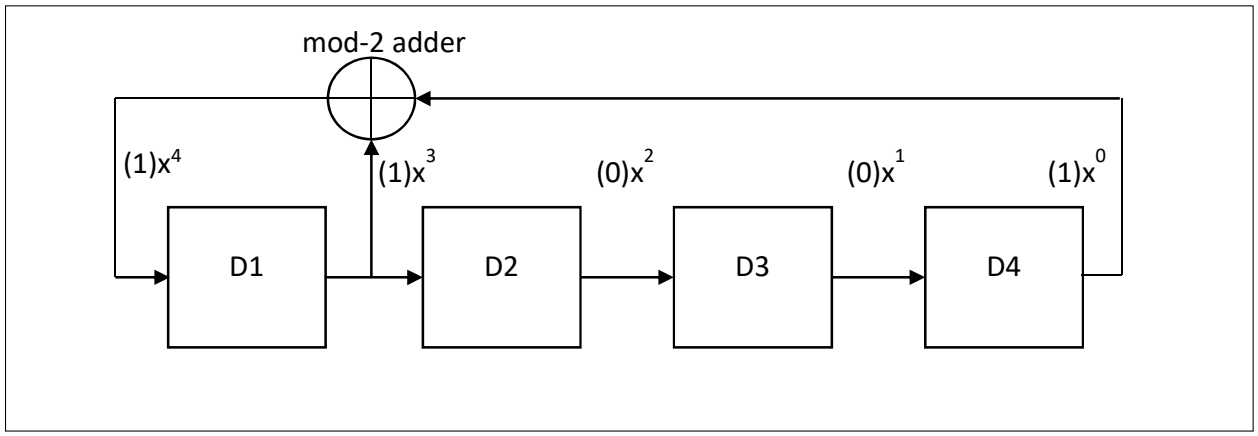

Figure 1.4: An example of a 4-bit maximum length LFSR.

evaluate sequences generated by cellular automata, and are as follows [7].

1. **Balance:** The number of ones contained in a m-sequence is one greater than the number of zeros. It has $2^{n-1}(0.5(N+1))$ ones and $2^{n-1} - 1(0.5(N-1))$ zeros within a period (N). The balance property of an m-sequence is optimal.

2. **Run:** For an m-sequence

   - there is 1 run of ones of length n,
   - there is 1 run of zeros of length $n-1$,
   - there are 1 run of ones and 1 run of zeros of length $n-2$,
   - there are 2 runs of ones and 2 runs of zeros of length $n-3$,
   - there are 4 runs of ones and 4 runs of zeros of length $n-4$,

...

- there are 2^{n-3} runs of ones and 2^{n-3} runs of zeros of length 1.

3. **Autocorrelation:** The autocorrelation of an m-sequence is given by

$$r(k) = \begin{cases} N, & k = aN \\ -1, & k \neq aN \end{cases} \tag{1.5}$$

4. **Linear Complexity:** The linear complexity of an m-sequence is

$$LC = \lceil \log_2 N \rceil = n \tag{1.6}$$

Thus, m-sequences have low linear complexity.

1.2 Cellular Automata

Cellular Automata (CA) are simple circuits which produce outputs based on predefined rules [8] [9]. These circuits were first conceptualized by Ulam and von Neumann in 1940 and their scope and applications in computer systems were investigated by Wolfram in 2001 [9]. CAs are structured as an array of binary cells in multiple dimensions. The next state (0 or 1) of a cell is determined by a rule that takes the current states of cells in its neighborhood as inputs. All cells are synchronized with an external clock. These circuits have the ability to produce highly complex sequences even though the individual rules are simple [7]. Thus, cellular automata are utilized to produce pseudorandom sequences in this thesis.

The neighborhood of a cell depends on the number of dimensions of the cellular automaton. For example, in a 1D cellular automaton the cells to the left and right of a cell and the cell itself comprise the neighborhood (3 cells). However, the cells at the ends only have two cells in their neighborhood (itself and the cell adjacent to it). Higher dimensional structures have larger neighborhoods. This thesis considers only 1D cellular automata. 2D CAs are considered as part of future work.

The maximum size of a neighborhood in a 1D CA is 3, so, the next state of a cell

can be considered a boolean function of 3 inputs

$$s_i(k+1) = F(s_{i-1}(k), s_i(k), s_{i+1}(k)) \tag{1.7}$$

where
$s_i(k+1)$ is the next state of a cell,
$s_i(k)$ is the current state of a cell,
$s_{i-1}(k)$ is the current state of the left cell in the neighborhood,
$s_{i+1}(k)$ is the current state of the right cell in the neighborhood, and
$F()$ is the logical function or rule.

Cells at the ends of a 1D CA have 2 cells in their neighborhood, but can be considered to have 3 inputs with one input always zero. Figure 1.5 shows an example of a 1D CA of size $n = 8$. There are $2^3 = 8$ possible states in the state table, and $2^8 = 256$ possible state tables based on different possible next states. Each of these is called a *Wolfram rule* [9]. These rules are numbered 0 to 255, based on the next state generated by the respective state tables. In a cellular automaton, every cell can be configured with a separate rule. State tables for rule 30 (00011110) and rule 90 (01011010) are shown in Tables 1.2 and 1.3, respectively. In these tables it can be seen that the 3-bit current states produce 8 1-bit next states, which together make up the Wolfram rule. The MSB of the rule corresponds to current state 111 while the LSB corresponds to current state 000.

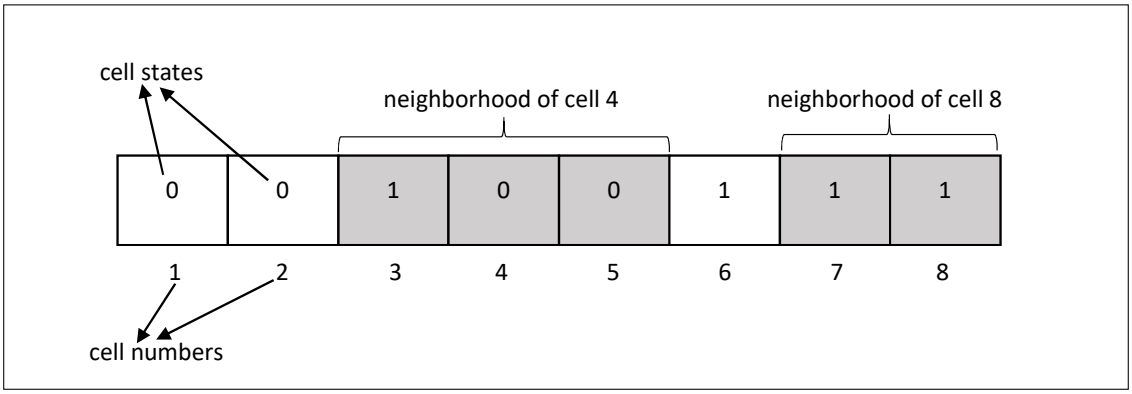

Figure 1.5: An example of a 1D cellular automaton.

Each of the 256 Wolfram rules has an associated *complementary rule*. A rule produces the same next state as its complement with the neighborhood cell positions

Current State			Next State	
$s_{i-1}(k)$	$s_i(k)$	$s_{i+1}(k)$	$s_i(k+1)$	
1	1	1	0	30
1	1	0	0	
1	0	1	0	
1	0	0	1	
0	1	1	1	
0	1	0	1	
0	0	1	1	
0	0	0	0	

Table 1.2: Rule 30 State Table

Current State			Next State	
$s_{i-1}(k)$	$s_i(k)$	$s_{i+1}(k)$	$s_i(k+1)$	
1	1	1	0	90
1	1	0	1	
1	0	1	0	
1	0	0	1	
0	1	1	1	
0	1	0	0	
0	0	1	1	
0	0	0	0	

Table 1.3: Rule 90 State Table

reversed. For example, let

$$s = s_i(k+1),\ a_0 = s_i(k),\ a_{-1} = s_{i-1}(k),\ a_1 = s_{i+1}(k)$$

Then the logical equation of rule 143 is given by

$$s = F(a_{-1}, a_0, a_1) = (a_0 a_1) + \overline{a}_{-1}$$

where $\overline{a}$ is the logical inverse operation and $F()$ is the logical function or rule. The complement of rule 143 is rule 213, which is given by

$$s = F(a_{-1}, a_0, a_1) = (a_0 a_{-1}) + \overline{a}_1$$

It can be seen from these equations that the positions of the current states (a_{-1} and a_1) of rules 143 and 213 are reversed.

There are eight rules (0, 60, 90, 102, 150, 170, 204 and 240) that are linear. The

next state for these rules is a linear function of the current state of the neighborhood. The logical equations of these rules are given by

$$\begin{aligned}
\text{rule } 0 &: s_i(k+1) = 0, \\
\text{rule } 60 &: s_i(k+1) = s_{i-1}(k) \oplus s_i(k), \\
\text{rule } 90 &: s_i(k+1) = s_{i-1}(k) \oplus s_{i+1}(k), \\
\text{rule } 102 &: s_i(k+1) = s_i(k) \oplus s_{i+1}(k), \\
\text{rule } 150 &: s_i(k+1) = s_{i-1}(k) \oplus s_i(k) \oplus s_{i+1}(k), \\
\text{rule } 170 &: s_i(k+1) = s_{i+1}(k), \\
\text{rule } 204 &: s_i(k+1) = s_i(k), \\
\text{rule } 240 &: s_i(k+1) = s_{i+1}(k).
\end{aligned} \tag{1.8}$$

Rules 0, 60, 102, 170, 204 and 240 produce poor results with respect to pseudorandom sequence generation. Rule 0 makes the state of a cell zero, while 204 retains the current state of the cell. Further, rules 60, 102, 170 and 240 divide the CA into two parts [9]. However, rules 90 and 150 can be used in a CA to generate m-sequences [11] [12]. Rules 90 and 150 can be implemented using XOR gates/mod-2 adders. Rule 90 takes only adjacent cells as inputs (2 inputs), while rule 150 takes adjacent cells and the cell itself as inputs (3 inputs), to generate the next state of the cell. It was shown that every primitive polynomial has at least one CA associated with it which is composed of rules 90 and 150. The transition matrix of such a CA is given by

$$A_{CA} = \begin{bmatrix}
y_1 & z_1 & 0 & \dots & \dots & 0 & 0 \\
x_2 & y_2 & z_2 & \ddots & & & 0 \\
0 & x_3 & y_3 & \ddots & \ddots & & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & \ddots & \ddots & y_{n-2} & z_{n-2} & 0 \\
0 & & & \ddots & x_{n-1} & y_{n-1} & z_{n-1} \\
0 & 0 & \dots & \dots & 0 & x_n & y_n
\end{bmatrix} \tag{1.9}$$

This matrix can be simplified in terms of the linear rules 90 and 150 as

$$A_{CA} = \begin{bmatrix} d_1 & 1 & 0 & \dots & \dots & 0 & 0 \\ 1 & d_2 & 1 & \ddots & & & 0 \\ 0 & 1 & d_3 & \ddots & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & d_{n-2} & 1 & 0 \\ 0 & & & \ddots & 1 & d_{n-1} & 1 \\ 0 & 0 & \dots & \dots & 0 & 1 & d_n \end{bmatrix} \tag{1.10}$$

Each element of the diagonal vector $(d_1, d_2, \dots, d_{n-1}, d_n)$ signifies a linear rule according to

$$d_i = \begin{cases} 0 \text{ rule } 90 \\ 1 \text{ rule } 150 \end{cases}$$

The diagonal vector can be obtained from the characteristic polynomial using the algorithm in [11]. The state transition equation is $s_{k+1} = A_{CA} s_k$, where s_k is the current state of the CA. Figure 1.6 shows a 4-cell CA with a combination of linear rules that produces the m-sequence with characteristic polynomial $x^4 + x + 1$. The states produced by this CA are identical to those in Table 1.1, but they are not in the same order as with the corresponding LFSR.

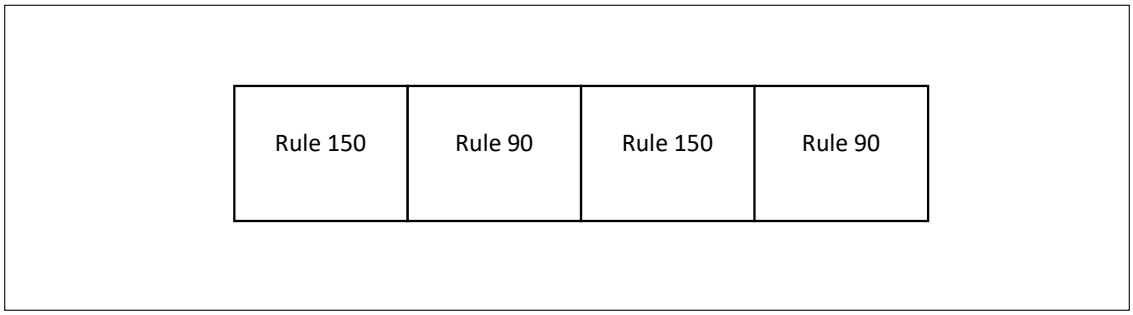

Figure 1.6: An example of a 4-bit 1D CA that produce an m-sequence.

One advantage of using CAs to generate sequences is that there are no long feedback paths, which can cause delays particularly with large LFSRs. This is because all computations happen in local neighborhoods, so the feedback paths are minimized. Further, they allow for easy generation of a variety of sequences. This is the motivation in this thesis to generate pseudorandom sequences from CAs.

1.3 Thesis Outline

The thesis structure is as follows.

Chapter 2 describes the 1D CA evaluation system. It outlines the parameters associated with the system and the functions of its modules. It also provides the filtering criteria based on linear complexity, balance, run and autocorrelation for selection of rule combinations that produce the best pseudorandom sequences.

Chapter 3 discusses the results obtained from the CA evaluation system for CA sizes $n = 3$ to 6 and provides an analysis of the generated sequences. Filtered results are given for each CA size based on the criteria specified in Chapter 2. These results are used to determine those parameters that produce pseudorandom sequences with high linear complexity and good randomness. These parameters are then applied to CA sizes $n > 6$. The properties of the sequences obtained are compared with those of the m-sequences.

Chapter 4 provides the conclusions of this thesis and suggestions for future work associated with pseudorandom sequence generation using CAs.

Chapter 2

Cellular Automata System for Pseudorandom Sequence Generation

In Chapter 1 it was seen that combinations of linear rules in a CA can generate m-sequences. The use of non-linear rules in CAs is considered in this chapter to generate pseudorandom sequences that have high linear complexity and good randomness. This is achieved by analyzing CAs with combinations of linear and non-linear rules. The m-sequences and their properties are used as a reference for this analysis [10] [18].

In [6], an evaluation system was designed for 1D CAs. Each cell was initially assigned a linear rule (90 or 150) with a combination that produces m-sequences. One of these cells was replaced with a random rule (non-linear) and the linear complexity of the generated sequence was calculated. Furthermore, linear rules based on primitive polynomials were combined with non-linear rules and the maximum complexity was obtained for different CA sizes. In this thesis, the system in [6] is expanded. All 2^n combinations of rule 90 and 150 are considered for the cells in the CA with one cell replaced with all possible non-linear rules (0 to 255, excluding 0, 60, 90, 102, 150, 170, 204 and 240). The linear complexity and randomness of the generated sequences are determined based on multiple criteria. The new evaluation system is characterized by the parameters given below.

1. **Size (n):** The size of the CA is the number of cells in the 1D CA. For example, Figure 2.1 shows a 1D CA of size $n = 5$.

2. **Linear Rules (LR):** Linear rules is the set of rules 90 and 150 initially assigned to the 1D CA. For convenience, rule 90 is denoted by bit 0 and rule 150 by bit 1. This forms an n-bit vector that represents the CA. For example the 5-bit 1D CA, shown in Figure 2.1 is represented by $LR = 11001$. For convenience, LR is also expressed in decimal.

3. **Start Value (SV):** Start Value is the initial state of the 1D CA. SV is a vector of n-bits. In Figure 2.1, the initial state of the CA is $SV = 10001$. For convenience, SV is also expressed in decimal.

4. **Random Rule (RR) and Randomized Cell (RC):** Random rule is one of the 248 non-linear rules that replaces a linear rule in the 1D CA. The non-linear rule is associated with a cell position which is the randomized cell. In Figure 2.1, non-linear rule $RR = 23$ replaces rule 90 at cell position $RC = 3$.

5. **Observed Cell (OC):** Observed cell is the cell position that is monitored to obtain a sequence of bits. In Figure 2.1, $OC = 2$ (cell position 2).

6. **Linear Complexity (LC):** The linear complexity of the sequence from OC.

7. **Balance (B):** The balance of the sequence from OC. It is the number of zeros subtracted from the number of ones in S.

8. **Run (R):** The run of the sequence from OC. It is an array of size 12 containing runs of length 1, 2, 3, ..., 12.

9. **Autocorrelation (AC) and Max Sidelobe Ratio (MSR):** Autocorrelation is an array containing values of the autocorrelation of the sequence from OC. Maximum sidelobe ratio is the ratio of the magnitude of the second largest value to the magnitude of the largest value (N) in the autocorrelation. The lower the value of MSR, the better the autocorrelation of the sequence.

10. **Sequence (S):** The sequence from OC. The length of the bit stream is $2^n - 1$ which is the period of an m-sequence generated by an LFSR of the same length as the CA (n).

In this system, parameters 1 to 5 are varied and the parameters 6 to 10 are analyzed. Section 2.1 provides a brief description of the software framework of the 1D CA evaluation system.

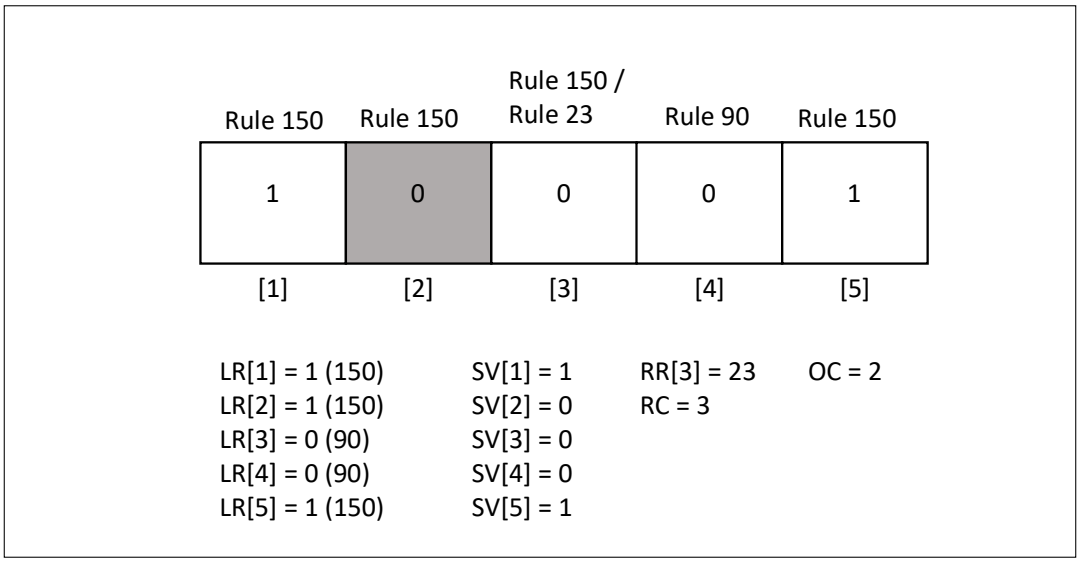

Figure 2.1: An example of the configured rules in the 1D CA evaluation system.

2.1 The 1D CA Evaluation System

The evaluation system was developed using the C programming language on the Linux Ubuntu 14.04 operating system. The objective of this system is to provide a means of analyzing 1D CAs containing a non-linear rule for pseudorandom sequence generation. The program is divided into modules as shown in Figure 2.2 and described below.

Main Control Module: The main control module generates all possible values of parameters 1 to 5 for the CA module. With one set of these parameters, an iteration of the CA module produces a sequence. The number of iterations is determined by the

- 2^n possible values of LR,
- 2^n possible values of SV,
- n possible values of OC,
- 248 possible non-linear rules (RR),
- n possible values of RC. However, in order to reduce the number of calculations the edge cells are not considered [6]. The non-linear rules have less effect on these cells as one of the cells in the neighborhood is a null boundary cell. Null boundaries are explained under CA Module. Hence, the possible values of RC is reduced to $n - 2$.

Thus, for one random cell replacement, the number of iterations is

$$2^n \times 2^n \times n \times 248 \times (n-2) \tag{2.1}$$

so for $n = 4$

$$2^4 \times 2^4 \times 4 \times 248 \times (4-2) = 507904$$

CA Module: The CA module consists of a variable length 1D CA with null boundaries. The parameters generated by the Main Control module are used to configure the CA. The input n is the size of the CA. Null boundary cells are added to the ends of the n-bit CA and are numbered 0 and $n+1$. No CA rules are associated with them and their states are always 0. As mentioned in Section 1.2, the cells at the edge of a CA (1 and n) have one input zero in their respective neighborhoods. The null boundaries provide these zero inputs to the edge cells. The bit vector LR is the linear rules of the n cells. RR and RC replace one linear rule with a non-linear rule in a specific position. The cell corresponding to OC is used to obtain the output sequence S. The SV bit vector is the initial state of the CA. The state of the CA is updated based on the combination of linear rules replaced with one non-linear rule. The number of output bits is twice the period of the corresponding m-sequence ($2 \times (2^n - 1)$). The initial $2^n - 1$ bits are discarded and the last half is used as the output sequence S, as shown in Figure 2.3. This is done to remove any unwanted effects of the initial conditions. S is used by the other modules which are described below.

LC Calculator: This module calculates the linear complexity of the sequence S. It employs the Berlekamp-Massey algorithm to determine LC [6] [15].

Balance Calculator: This module calculates the number of 0s and 1s in S. It subtracts the number of 0s from the number of 1s to produce B.

Run Calculator: This module calculates the number of runs of lengths one, two, three, ..., twelve. The runs are calculated for both 0s and 1s. R is the array of runs of lengths $1, 2, 3, \ldots, 12$.

AC Calculator: This module calculates the autocorrelation of S.

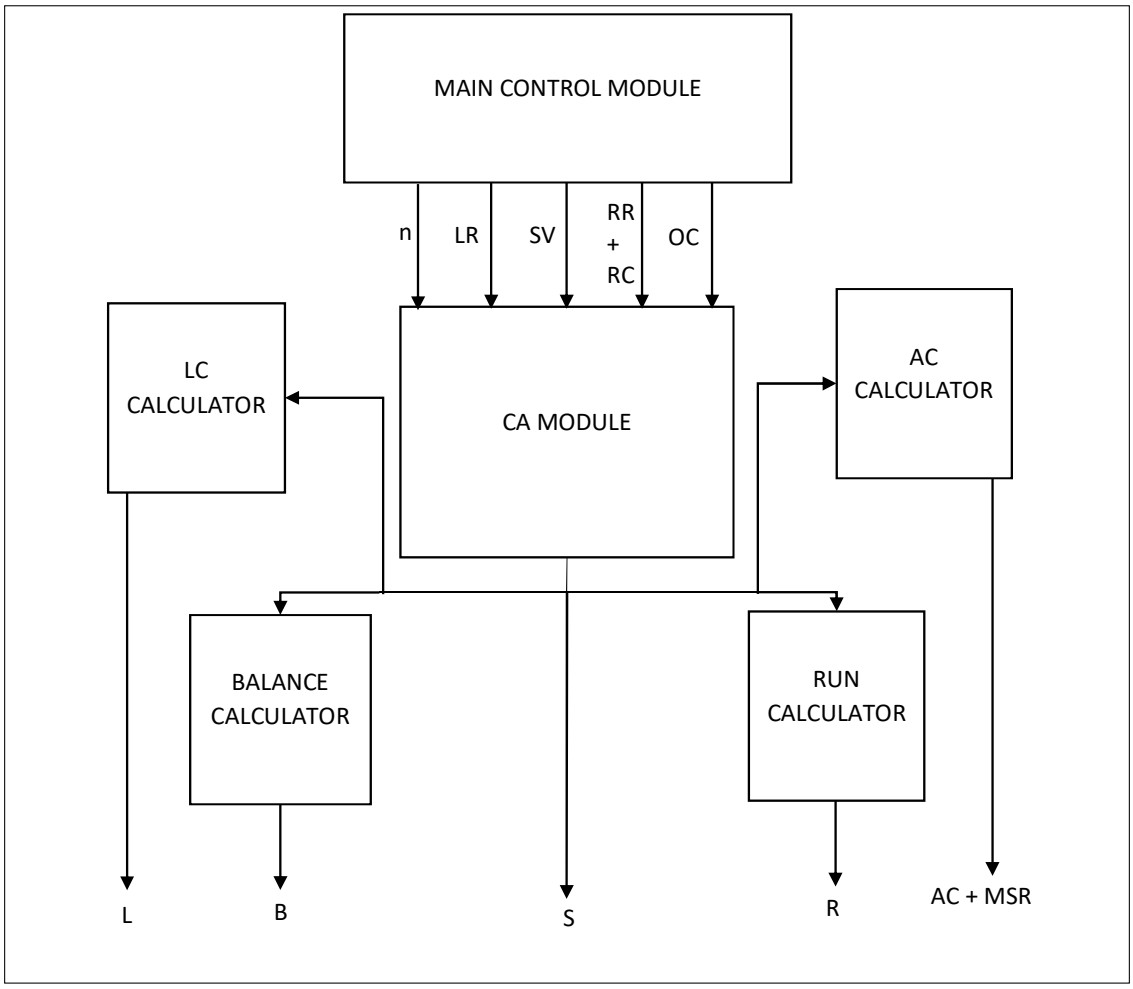

Figure 2.2: Modules of the 1D CA evaluation system.

2.2 Filtering Criteria

The properties of the generated sequences are evaluated and the set of best linear rules (LR) in combination with non-linear rules (RR) and their corresponding cell positions (RC) are obtained. This selection is based on criteria which were designed with reference to the properties of m-sequences and the randomness tests defined in [13]. These criteria are given below.

1. Linear complexity is used as the initial filtering criteria because the objective in this thesis is to obtain sequences with high linear complexity. Only those LR, RR and RC combinations which have $LC \geq n$ for small n (3 and 4) and $LC \geq 2^n/4$ for $n > 4$, are considered. This is because large linear complexity is the goal of this thesis.

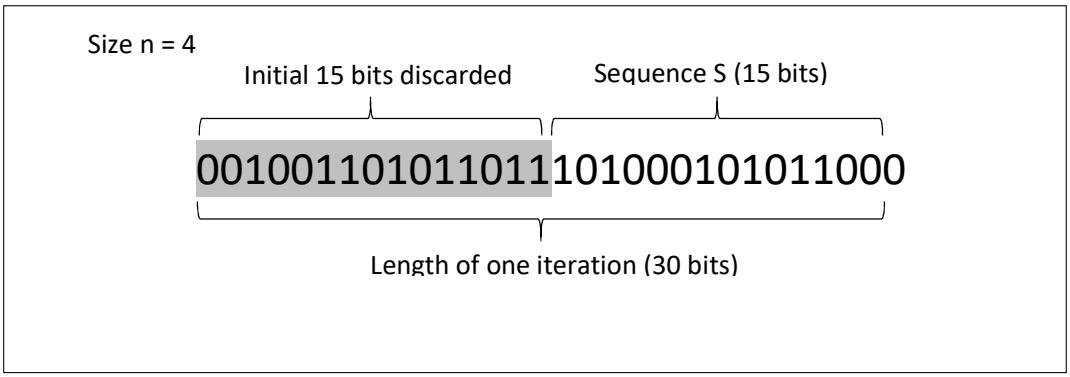

Figure 2.3: Sequence generated by a CA of size $n = 4$.

2. Only those rules which maintain a relatively unchanged LC of no more than $\pm 10\%$ for all 2^n start values (SV) are chosen. The objective here is to obtain good sequences irrespective of the initial state of the CA, which is similar to m-sequences.

3. The sequences obtained after steps 1 and 2 are evaluated using the tests provided in [13]. The two tests used for balance and run are given below.

   (a) **Frequency Test for Balance:** In this test each bit of a sequence is assigned a value -1 or +1 (0 = -1 and 1 = +1). The sum of the values is calculated as

   $$X_N = \sum_{m=0}^{N-1} 2s[m] - 1$$

   where $s[m]$ is the mth bit of S and $N = 2^n - 1$ is the length of S. This is used to calculate the complementary error function

   $$P_B = \text{erfc}(|X_N|/(\sqrt{2N}))$$

   A balance threshold value of $P_{BTh} = 0.01$ is used to filter the parameters LR, RR and RC, so the combination is kept if $P_B > P_{BTh}$ [13].

   (b) **Run Test:** In this test the ratio of the number of 1s to the length of the sequence is calculated as

   $$\pi = \sum_{m=0}^{N-1} s[m]/N$$

where $s[m]$ is the mth bit of S. If the condition

$$|\pi - 1/2| < X_N/\sqrt{N}$$

is not satisfied, then the test fails. Then the test statistic

$$V_N(obs) = \sum_{m=0}^{N-2} v[m] + 1$$

is calculated, where $v[m] = 0$ if the $(m+1)$th bit is the same as the mth bit and $v[m] = 1$, otherwise. The corresponding complementary error function

$$P_R = \text{erfc}\left(\frac{|V_N(obs) - 2N\pi(1-\pi)|}{2\sqrt{2N}\pi(1-\pi)}\right)$$

is calculated. A run threshold value of $P_{RTh} = 0.01$ is used to filter the parameters so that the combination is kept if $P_R > P_{RTh}$ [13].

4. To evaluate the autocorrelation of the sequences, the maximum sidelobe is considered. The autocorrelation for an m-sequence has a mainlobe of magnitude N and sidelobes of magnitude 1, as shown in (1.5). However, the sequences generated by the CA might produce larger sidelobes. The MSR is used to evaluate this. A threshold of $MSR_{Th} = 0.3$ is used because it is low and produces a small set of results. Parameters that generate sequences with MSR greater than MSR_{Th} are discarded.

The filtering process described above has two stages. The first or initial stage consists of steps 1 and 2 where the filtering process is based on linear complexity. The second or final stage consists of steps 3 and 4, where properties of randomness (balance, run and autocorrelation) form the criteria for filtering the sequences. The filtered results for CA sizes $n = 3, 4, 5$ and 6 were obtained using the above criteria, and the properties of the sequences obtained were compared to those of the m-sequences. These results are used to determine those values of LR, RR and RC that produce pseudorandom sequences with high linear complexity and good randomness. These parameters are then applied to CA sizes $n > 6$. Only single replacement of linear rules with a non-linear rule in the CA is investigated in this thesis. Multiple replacements of linear rules is left for future work.

Chapter 3

Results and Analysis

The 1D CA evaluation system was first used to analyze single cell replacements of linear rules with non-linear rules in CAs of sizes $n = 3, 4, 5$ and 6 and then for $n > 6$. The outputs were captured in Excel files. During the initial stage of the filtering process, certain observations were made, and they are elaborated below.

3.1 Initial Observations

1. Even non-linear rules produce all-zero sequences (0000...) with the initial state $SV = 0$. Examples of some even rules with $SV = 0$ for $n = 3, 4, 5$ and 6 are given in Table 3.1. This occurs because the LSB bit of an even rule is 0, so when the neighborhood inputs are 000, the next state of the cell is 0. The linear rules of the other cells in the CA are either Rule 90 or 150, which are even. Hence, the next state of every cell will always be 0 when $SV = 0$.

2. The linear complexity (LC) of the sequences remains approximately the same when the observed cell (OC) (including end cells) is varied with all other parameters fixed. Examples of this behavior for $n = 3, 4, 5$ and 6 are given in Tables 3.2, 3.3, 3.4, 3.5, respectively. In each of the tables, LR, RR, RC and SV are constant and only OC is varied. A deviation in LC occurs only for one or two values of OC with a maximum deviation of $\pm 10\%$.

3. Linear rule combinations (LR) that produce m-sequences (based on primitive polynomials) always have at least one non-linear rule replacement that produces high linear complexity (LC) [6]. For fixed values of SV and RC, and LR based

on primitive polynomials, the maximum complexity occurs at a different RR for each CA size $n = 3, 4, 5, 6, 7$ and 8, and in each case the maximum complexity is close to $2^n/2$. Table 3.6 provides examples with maximum complexity for values of n upto 8 for $SV = 3$ and $RC = 2$. The table also contains the associated primitive polynomials for each case. Figure 3.1 is a plot of these LC values which shows that the maximum complexity approximately doubles with an increase in CA size of one.

4. Complementary rules produce the same sequences when the CA is reversed. For example, for $n = 4$, rule 30 with $LR = 5$ (0101) and $RC = 3$, produces the same sequence as its complement rule 86 with $LR = 10$ (1010) and $RC = 2$. Note that the reverse of $LR = 5$ is $LR = 10$. The reversed position of $RC = 3$ is $RC = 2$ for $n = 4$, as shown in Figure 3.2. Table 3.7 gives examples of complementary rules producing the same sequences.

5. Based on the position of the randomized cell (RC), there are sets of *duplicate LRs*. This means that a non-linear rule (RR) for a particular RC will produce the same sequence for two different LRs, with all other parameters fixed. For example, for $n = 4$ and $RR = 59$, $RC = 2$ will produce the same sequence for $LR = 9$ and 13. This example and others are shown in Table 3.8. This occurs because with a single replacement, a non-linear rule replaces either a rule 90 (0) or a rule 150 (1). For example, both $LR = 9$ (1001) and $LR = 11$ (1011) produce the combination $10X1$, where X signifies the non-linear rule at $RC = 3$. Hence, they are duplicate LRs for $RC = 3$.

As mentioned in Section 2.2, the results from the initial stage of the filtering process were further filtered (final stage) based on the criteria for balance, run and autocorrelation, i.e.

$$\begin{aligned} P_B &> 0.01 \\ P_R &> 0.01 \\ MSR &< 0.3 \end{aligned} \tag{3.1}$$

to obtain the final results.

n	OC	RC	RR	LR	SV	LC	B	R	AC	S
3	1	2	30	101	000	0	-7	0,0,0,0,0,0, ...	7,7,7,7,7,7,7	0000000
4	2	2	106	0110	0000	0	-15	0,0,0,0,0,0, ...	15,15,15,15,15,15,15, ...	0000000...
5	3	2	174	00111	00000	0	-31	0,0,0,0,0,0, ...	31,31,31,31,31,31,31, ...	0000000...
6	5	2	212	110001	000000	0	-63	0,0,0,0,0,0, ...	63,63,63,63,63,63,63, ...	0000000...

Table 3.1: All-zero Sequences Produced by Even Rules for $SV = 0$ and $n = 3, 4, 5, 6$

OC	RC	RR	LR	SV	LC	B	R	AC	S
1	2	15	001	010	3	1	1,3,0,0,0,0, ...	7,-1,-5,3,3,-5,-1	0110011
2	2	15	001	010	3	1	1,3,0,0,0,0, ...	7,-1,-5,3,3,-5,-1	1100110
3	2	15	001	010	4	-3	4,0,1,0,0,0, ...	7,-1,-1,3,3,-1,-1	0100010

Table 3.2: Linear Complexity versus Observed Cell for $n = 3$, $RR = 15$ and $RC = 2$

3.2 Filter Results for $n = 3$

After the initial stage of the filtering process, LR, RC and RR parameters for $n = 3$ were obtained that produce sequences with $LC \geq 3$. $SV = 0$ for even RR was ignored during the filtering process because this produces all-zero sequences (initial observation 1), so for even RR only $SV = 1$ to 7 were considered. To reduce the amount of data, the initial stage of the filtering process was done only for $OC = 2$, considering that the linear complexity remains approximately the same for all values of OC (initial observation 2). Table 3.9 shows the LR, RC and RR parameters from the initial stage of the filtering process for $n = 3$.

It can be seen in Table 3.9 that the sets of rules are repeated. This is because, for single rule replacements, $1, 3$ and $4, 6$ are two pairs of duplicate LRs (initial obsevation 5). Further, the rules for $LR = 4$ are complementary to the those for $LR = 1$, because 4 (100) is the reverse of 1 (001) (initial observation 4). Similarly, $LR = 3$ and $LR = 6$ have complementary sets of non-linear rules. Tables 3.10 and 3.11 LC, B, R and AC values produced with the initial filtered parameters $LR = 3$, $RR = 99$, and $LR = 6$, $RR = 18$, respectively. In both tables it can be seen that the maximum linear complexity is $LC = 4$. The sequences produced by the parameters in Table 3.9 were put through the final stage of the filtering process. It is observed that no parameters satisfy all the criteria given in (3.1). However, Table 3.12 gives the parameters LC, RC and RR that satisfy the criteria for P_B and P_R, but do not satisfy the MSR criterion for all values of SV. The LC, P_B, P_R and MSR values for $LR = 1$, $RR = 122$ and $LR = 4$, $RR = 183$ are shown in Tables 3.13 and 3.14, respectively. These tables give the results for all values of SV. In Table 3.13, it can

OC	RC	RR	LR	SV	LC	B	R	AC	S
1	2	169	0011	1000	7	-1	4,1,3,0,0,0, ...	15,-1,-1,-5,-1,-5, ...	0001011...
2	2	169	0011	1000	7	1	4,1,3,0,0,0, ...	15,-1,-1,-5,-1,-5, ...	0010111...
3	2	169	0011	1000	6	1	3,3,2,0,0,0, ...	15,-1,-5,-1,-1,-1, ...	1001110...
4	2	169	0011	1000	7	-1	6,0,3,0,0,0, ...	15,-1,-1,-5,-1,-5, ...	1000101...

Table 3.3: Linear Complexity versus Observed Cell for $n = 4$, $RR = 169$ and $RC = 2$

OC	RC	RR	LR	SV	LC	B	R	AC	S
1	3	146	00110	00001	16	-3	12,2,5,0,0,0, ...	31,-5,3,-9,15,-5,-1, ...	0010001...
2	3	146	00110	00001	16	-1	14,1,5,0,0,0, ...	31,-9,7,-9,15,-9,3, ...	0100010...
3	3	146	00110	00001	16	-3	20,0,1,2,0,0, ...	31,-13,15,-9,3,3,-9, ...	1010100...
4	3	146	00110	00001	16	3	2,3,6,0,1,0, ...	31,7,-9,-13,-1,3,-5, ...	0001110...
5	3	146	00110	00001	16	3	4,2,6,0,1,0, ...	31,7,-9,-13,-1,3,-5, ...	1000111...

Table 3.4: Linear Complexity versus Observed Cell for $n = 5$, $RR = 146$ and $RC = 3$

be seen that the sequences have $MSR = 0.43$ (≥ 0.3) for all SV except $SV = 4$ and 6. For $SV = 4$ and 6, the MSR criterion is satisfied, but $LC = 3$, which is the same as that of an m-sequence. This is also seen in Table 3.14. The other parameters in Table 3.12 produce similar results.

3.3 Filter Results for $n = 4$

As in the case of $n = 3$, the initial stage of the filtering process for $n = 4$ was done for $OC = 2$ to reduce the amount of data, and results for $LC \geq 4$ were obtained. The resulting sequences were then put through the final stage of the filtering process with the criteria given in (3.1). The results obtained show that some combinations of LR, RC and RR satisfy all the filtering criteria for all SV, and they are given in Table 3.15. Similar to Tables 3.9 and 3.12, Table 3.15 has duplicate LRs. They are $LR = 8, 10$ and $LR = 9, 11$ for $RC = 3$. Tables 3.16 and 3.17 give the LC, P_B, P_R and MSR values for $LR = 8$, $RC = 38$, $RR = 86$ and $LR = 9$, $RC = 3$, $RR = 178$, respectively. The maximum linear complexity from the tables is $LC = 9$. The P_B and MSR are 0.8 and 0.2, respectively for all SVs. In Table 3.16, P_R is either 0.43 or 0.78 depending on the value of SV.

OC	RC	RR	LR	SV	LC	B	R	AC	S
1	3	89	010000	001001	32	1	11,8,6,2,2,0, ...	63,7,-9,-5,-5,-1,-9, ...	0010001...
2	3	89	010000	001001	32	1	11,8,6,2,2,0, ...	63,7,-9,-5,-5,-1,-9, ...	0100010...
3	3	89	010000	001001	32	11	19,5,3,5,1,0, ...	63,-1,11,-9,11,-1,19, ...	1110111...
4	3	89	010000	001001	34	-11	19,6,3,1,1,0, ...	63,-1,11,-1,7,7,11, ...	1000101...
5	3	89	010000	001001	30	11	19,4,2,5,2,0, ...	63,-1,11,-9,11,-1,19, ...	1111101...
6	3	89	010000	001001	31	11	19,4,2,5,2,0,...	63,-1,11,-9,11,-1,19, ...	0111110...

Table 3.5: Linear Complexity versus Observed Cell for $n = 6$, $RR = 89$ and $RC = 3$

n	Primitive Polynomial	LR	RR	Max LC
3	$x^3 + x + 1$	110	18	4
4	$x^4 + x + 1$	1010	230	9
5	$x^5 + x^4 + x^2 + x + 1$	10000	86	17
	$x^5 + x^2 + 1$	01111	91	16
6	$x^6 + x + 1$	011000	135	34
	$x^6 + x^5 + x^2 + x + 1$	100000	230	33
7	$x^7 + x^3 + x^2 + x + 1$	1000111	27	64
	$x^7 + x^3 + 1$	0111010	75	64
8	$x^8 + x^4 + x^3 + x^2 + 1$	00000110	75	129
	$x^8 + x^6 + x^5 + x^3 + 1$	11110000	86	128

Table 3.6: Maximum Complexity versus Size for $SV = 3$ and $RC = 2$

3.4 Filter Results for $n = 5$ and $n = 6$

As for $n = 3$ and 4, the initial stage of the filtering process for $n = 5$ was done for $OC = 2$ to reduce the amount of data. The sequences with $LC \geq 2^n/4 = 8$ obtained from the inital stage were put through the final stage of the filtering process with the criteria given in (3.1). It was observed that no parameters satisfy all these criteria. However, some satisfy the criteria for P_B and P_R, but not the criterion for MSR. The corresponding parameters LR, RC and RR are given in Table 3.18. Each combination produces a sequence with $MSR \geq 0.3$ for a few values of SV. This can be seen in the results for $LR = 3$, $RC = 4$ and $RR = 91$ given in Table 3.20. In this table, the sequences for $SV = 11, 18, 23$ and 31 have $MSR = 0.35$ (≥ 0.3), but $MSR < 0.3$ for all other values of SV. This behavior is also observed for all other parameters in Table 3.18, where the MSR criterion is not satisfied for a few values of SV.

The same procedure as with $n = 3, 4$ and 5 was applied for $n = 6$. The final filtered parameters after both stages of the filtering process are given in Table 3.19. These satisfy all the criteria in (3.1) and have $LC \geq (2^n/4) = 16$. The results for

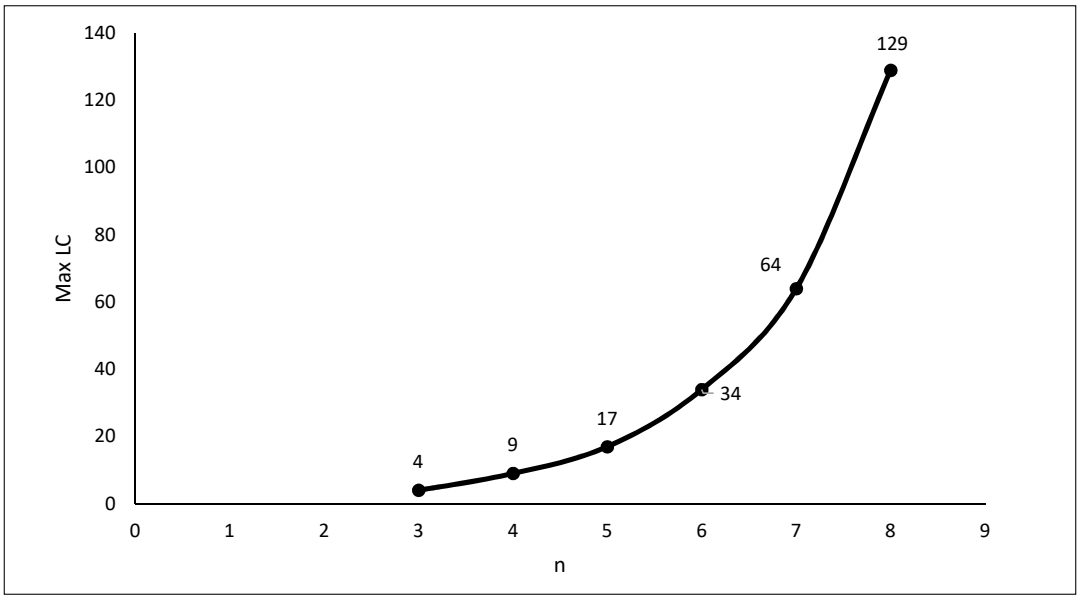

Figure 3.1: Maximum linear complexity using linear rules based on primitive polynomials versus size.

n	OC	RC	RR	LR	SV	LC	B	R	AC	S
3	1	2	35	011	001	4	-3	3,2,0,0,0,0, ...	7,-1,3,-1,-1,3,-1	0010100
	3	2	49	110	100	4	-3	3,2,0,0,0,0, ...	7,-1,3,-1,-1,3,-1	0010100
4	3	3	107	1010	0111	7	3	10,1,1,0,0,0, ...	15,-9,7,-5,7,-5,3, ...	1010111...
	2	2	121	0101	1110	7	3	10,1,1,0,0,0, ...	15,-9,7,-5,7,-5,3, ...	1010111...

Table 3.7: Sequences produced by Complementary Rules for $n = 3, 4$

$LR = 46$, $RC = 2$ and $RR = 86$ are given in Tables 3.21 and 3.22, respectively, for all values of SV.

3.5 Observations for n up to 6 and Results for $n = 7$ and 8

From the results in the previous sections, the following observations were made.

1. Odd-sized CAs have more non-linear rules that with single replacement produce sequences that satisfy the filtering criteria than even-sized CAs. However, the filtered parameters for odd-sized CA have inconsistent behavior with respect to MSR. They satisfy the criteria for P_B and P_R in (3.1), but $MSR < 0.3$ is not satisfied for all values of SV. This can be seen in Tables 3.13, 3.14 and 3.20.

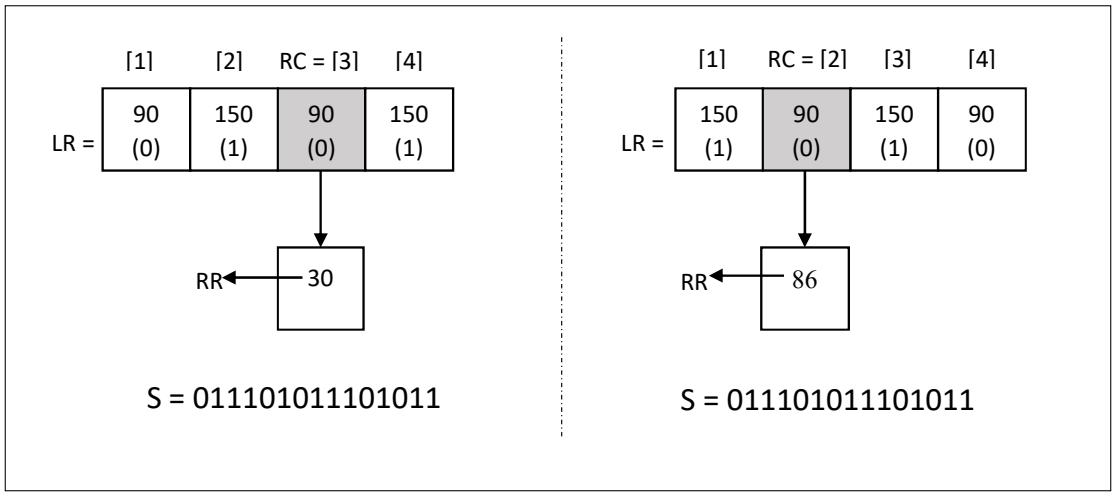

Figure 3.2: An example of complementary rules and a reversed CA.

OC	RC	RR	LR	SV	LC	B	R	AC	S
2	2	59	1001	0111	9	1	13,1,0,0,0,0, ...	15,-13,11,-9,7,-5,3, ...	1010101...
2	2	59	1101	0111	9	1	13,1,0,0,0,0, ...	15,-13,11,-9,7,-5,3, ...	1010101...
3	3	79	0111	1011	4	9	3,2,0,2,0,0, ...	15,3,3,3,3,15,3, ...	1101111...
3	3	79	0101	1011	4	9	3,2,0,2,0,0, ...	15,3,3,3,3,15,3, ...	1101111...
3	2	146	1110	0100	7	1	4,1,3,0,0,0, ...	15,-1,-1,-5,-1,-5,-1, ...	1110100...
3	2	146	1010	0100	7	1	4,1,3,0,0,0 ...	15,-1,-1,-5,-1,-5,-1, ...	1110100...

Table 3.8: Sequences produced by Duplicate Linear Rules for $n = 4$

On the other hand, the filtered results of even-sized CA satisfy all the filtering criteria, which can be seen in Tables 3.16, 3.17, 3.21 and 3.22.

2. In all the final filtered results, the LRs are either based on primitive polynomials or are duplicates of those LRs. The algorithm in [11] was used to determine that the LRs are based on primitive polynomials. For example $n = 4$, $LR = 10$ and 11 are based on the primitive polynomials $x^4 + x + 1$ and $x^4 + x^3 + 1$, respectively. $LR = 8$ and 9 are the duplicate LRs of 10 and 11, respectively for $RC = 3$. Table 3.23 summarizes all the final filtered LRs obtained in the previous sections and their associated primitive polynomials.

The second observation is used to obtain filtered rules for CA sizes $n = 7$ and $n = 8$. Instead of doing a search for filtered results with all 2^n LRs, it is restricted to LRs based on primitive polynomials. The algorithm in [11] was used to obtain the LRs from primitive polynomials. The filtered parameters for $n = 7$ and $n = 8$ are given

LR	RC	RR
1	2	18, 19, 35, 75, 82, 83, 99, 122, 123, 135, 146, 147, 163, 182, 183
3	2	18, 19, 35, 75, 82, 83, 99, 122, 123, 135, 146, 147, 163, 182, 183
4	2	18, 19, 26, 27, 49, 57, 89, 122, 123, 146, 147, 149, 177, 182, 183
6	2	18, 19, 26, 27, 49, 57, 89, 122, 123, 146, 147, 149, 177, 182, 183

Table 3.9: Initial Filtered Parameters LR, RC and RR for $n = 3$

OC	RC	RR	LR	SV	LC	B	R	AC	S
2	2	99	011	000	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1001010
2	2	99	011	001	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0101001
2	2	99	011	010	4	-3	3,2,0,0,0,0, ...	7,-1,3,-1,-1,3,-1	0010100
2	2	99	011	011	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1010010
2	2	99	011	100	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0100101
2	2	99	011	101	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0101001
2	2	99	011	110	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0101001
2	2	99	011	111	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1010010

Table 3.10: Results for $LR = 3$, $RC = 2$ and $RR = 99$ for $n = 3$

in Tables 3.24 and 3.25, respectively. These tables also contain the primitive polynomials on which the the LRs are based. Table 3.26 shows results for $LR = 43$, $RC = 6$ and $RR = 18$ for $n = 7$. As mentioned in the first observation, $n = 7$ is odd and does not satisfy the MSR criterion for all SV, which is seen in the table. The LC is between 63 and 67, which are on the order of $2^n/2 = 64$. Table 3.27 shows results for $LR = 93$, $RC = 4$ and $RR = 225$ for $n = 8$. All filtering criteria in (3.1) are satisfied in the results given in this table and also with all other filtered parameters for $n = 8$. The LC is between 124 and 130, which are on the order of $2^n/2 = 128$.

Even though the odd-sized CAs do not satisfy the $MSR < 0.3$ criterion for all SV, it can be seen that the maximum MSR decreases with increase in size n. The maximum MSR for $n = 3, 5$ and 7 are 0.43, 0.35 and 0.31, respectively (Tables 3.13, 3.20 and 3.26). For $n = 7$, the maximum $MSR = 0.31$ is very close to the threshold ($MSR_{Th} = 0.3$). Thus, it can be predicted that the same parameters for large odd-sized CAs ($n > 8$) will satisfy all the filtering criteria, including MSR. This is confirmed in Section 3.7.

OC	RC	RR	LR	SV	LC	B	R	AC	S
2	2	18	110	001	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0101001
2	2	18	110	010	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1010010
2	2	18	110	011	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1010010
2	2	18	110	100	3	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	1001010
2	2	18	110	101	4	-1	5,1,0,0,0,0, ...	7,-5,3,-1,-1,3,-5	0100101
2	2	18	110	110	3	-3	3,2,0,0,0,0, ...	7,-1,3,-1,-1,3,-1	0010100
2	2	18	110	111	4	-3	3,2,0,0,0,0, ...	7,-1,3,-1,-1,3,-1	0010100

Table 3.11: Results for $LR = 6$, $RC = 2$ and $RR = 18$ for $n = 3$

LR	RC	RR
1	2	122, 123, 182, 183
3	2	122, 123, 182, 183
4	2	122, 123, 182, 183
6	2	122, 123, 182, 183

Table 3.12: Final Filtered Parameters LR, RC and RR for $n = 3$

3.6 Comparison with m-sequences

This section compares m-sequences with the sequences produced by CAs with a single non-linear rule replacement as given in the tables in the previous sections. This comparison is done for $n = 3, 4, 5, 6, 7$ and 8.

1. For $n = 3$, Table 3.28 gives the results for the m-sequence generated with linear rule combination $LR = 6$ (110) and initial state $SV = 1$ (001), and the CA sequence generated by a single replacement of non-linear rule $RR = 183$ at cell position $RC = 2$ with the same SV. It can be seen that the linear complexity of the CA sequence is 4, which is higher than that of the m-sequence. The CA sequence has balance 3 and run 3,2,0,0,0,0, ..., which are not as good as those of the m-sequence, but satisfy the criteria in (3.1). The autocorrelation of the CA sequence is 7,-1,3,-1,-1,3,-1, which has $MSR = 0.43$. Table 3.29 gives a similar comparison with $SV = 4$ (100). This CA sequence has balance 1 and autocorrelation 7,-1,-1,-1,-1,-1,-1, which are equal to those of an m-sequence. In fact, this CA sequence is an m-sequence.

2. For $n = 4$, Table 3.30 gives the results for the m-sequence generated with $LR = 10$ (1010) and $SV = 2$ (0010) and the CA sequence generated by a single replacement of $RR = 86$ at $RC = 3$ with the same LR and SV. The

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	2	122	001	001	4	0.26	0.05	0.43	1101011
2	2	122	001	010	4	0.26	0.05	0.43	1110101
2	2	122	001	011	4	0.26	0.05	0.43	1010111
2	2	122	001	100	3	0.7	0.22	0.14	0111010
2	2	122	001	101	4	0.26	0.05	0.43	1101011
2	2	122	001	110	3	0.7	0.22	0.14	0101110
2	2	122	001	111	4	0.26	0.05	0.43	1011101

Table 3.13: Results for $LR = 1$, $RC = 2$ and $RR = 122$ for $n = 3$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	2	183	100	000	4	0.26	0.05	0.43	1101011
2	2	183	100	001	4	0.26	0.05	0.43	1101011
2	2	183	100	010	4	0.26	0.05	0.43	1010111
2	2	183	100	011	4	0.26	0.05	0.43	1011101
2	2	183	100	100	3	0.7	0.22	0.14	0111010
2	2	183	100	101	3	0.7	0.22	0.14	0111010
2	2	183	100	110	4	0.26	0.05	0.43	1110101
2	2	183	100	111	3	0.7	0.22	0.14	0101110

Table 3.14: Results for $LR = 4$, $RC = 2$ and $RR = 183$ for $n = 3$

CA sequence has linear complexity 8, which is twice that of the m-sequence. The balance is 1 and run is 4,1,3,0,0,0, ..., which are excellent and comparable to those of the m-sequence. The autocorrelation is 15,-1,-1,-3,3,3,3,..., which results in $MSR = 0.2$ as compared to $MSR = 0.07$ for the m-sequence. Similar results were obtained for the sequences produced with all other values of SV.

3. For $n = 5$, $LR = 7$ and $SV = 1$, the m-sequence is compared to the CA sequence generated with a single replacement of $RR = 107$ at $RC = 4$ and the results are given in Table 3.31. The linear complexity of the CA sequence is $LC = 2^n/2 = 16$. The balance is 3 and run is 6,3,2,2,1,0, ... which are close to those of the m-sequence. The autocorrelation is 31,3,-1,-9,-5,-1,3, ..., which is not as good as the m-sequence because the largest sidelobe -9 leads to an MSR of 0.29, but it satisfies the MSR criterion. Table 3.32 gives a similar comparison with $SV = 2$. However, in this table the CA sequence has $MSR = 0.35$, which does not satisfy the criterion in (3.1). The balance is 5 and run is 5,4,3,1,1,0, ... which are worse than those with $SV = 1$, but still satisfy the criteria for P_B and P_R.

LR	RC	RR
8	3	86
9	3	178
10	3	86
11	3	178

Table 3.15: Final Filtered Parameters LR, RC and RR for $n = 4$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	3	86	1000	0001	9	0.8	0.43	0.2	0111000...
2	3	86	1000	0010	8	0.8	0.78	0.2	1110001...
2	3	86	1000	0011	8	0.8	0.78	0.2	1011100...
2	3	86	1000	0100	7	0.8	0.78	0.2	0001010...
2	3	86	1000	0101	8	0.8	0.78	0.2	0111001...
2	3	86	1000	0110	7	0.8	0.43	0.2	1001110...
2	3	86	1000	0111	7	0.8	0.43	0.2	1100010...
2	3	86	1000	1000	8	0.8	0.78	0.2	1110011...
2	3	86	1000	1001	7	0.8	0.78	0.2	1010111...
2	3	86	1000	1010	7	0.8	0.43	0.2	0010101...
2	3	86	1000	1011	8	0.8	0.43	0.2	0101011...
2	3	86	1000	1100	8	0.8	0.43	0.2	1100111...
2	3	86	1000	1101	7	0.8	0.43	0.2	1000101...
2	3	86	1000	1110	7	0.8	0.78	0.2	0101110...
2	3	86	1000	1111	9	0.8	0.78	0.2	0011100...

Table 3.16: Results for $LR = 8$, $RC = 3$ and $RR = 86$ for $n = 4$

4. For $n = 6$, Table 3.33 gives the results for the m-sequence generated with $LR = 45$ and $SV = 7$ and the CA sequence generated with a single replacement of $RR = 86$ at $RC = 3$. The linear complexity of the CA sequence is $2^n/2 = 32$. The balance is 1 and run is 18,7,4,2,1,1, ..., which are excellent and equal to those of the m-sequence. The autocorrelation is 63,-1,-1,-1,-1,-1,7, ... with $MSR = 0.17$, which is good, but not as good as that of the m-sequence (0.02). Similar results were obtained for the sequences produced with all other values of SV.

5. For $n = 7$, $LR = 43$ and $SV = 6$, the m-sequence is compared to the CA sequence generated with single replacement of $RR = 18$ at $RC = 6$, and the results are given in Table 3.34. As with $n = 3, 4, 5$ and 6, the linear complexity of the CA sequence is $2^n/2 = 64$. The balance is 1 and the run is 37,15,7,2,5,1,

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	3	178	1001	0001	7	0.8	0.43	0.2	0100101...
2	3	178	1001	0010	8	0.8	0.78	0.2	1001111...
2	3	178	1001	0011	6	0.8	0.78	0.2	1001010...
2	3	178	1001	0100	6	0.8	0.78	0.2	0010100...
2	3	178	1001	0101	8	0.8	0.78	0.2	0011111...
2	3	178	1001	0110	7	0.8	0.43	0.2	1110010...
2	3	178	1001	0111	7	0.8	0.43	0.2	1001001...
2	3	178	1001	1000	8	0.8	0.78	0.2	1111100...
2	3	178	1001	1001	9	0.8	0.78	0.2	1010011...
2	3	178	1001	1010	6	0.8	0.43	0.2	0101001...
2	3	178	1001	1011	8	0.8	0.43	0.2	0111110...
2	3	178	1001	1100	8	0.8	0.43	0.2	1111001...
2	3	178	1001	1101	8	0.8	0.43	0.2	1100100...
2	3	178	1001	1110	8	0.8	0.78	0.2	0010010...
2	3	178	1001	1111	8	0.8	0.78	0.2	0100111...

Table 3.17: Results for $LR = 9$, $RC = 3$ and $RR = 178$ for $n = 4$

LR	RC	RR
1	4	91, 167
3	4	91, 167
5	4	107, 151
7	4	107, 151
16	2	91, 181
24	2	91, 181

Table 3.18: Final Filtered Parameters LR, RC and RR for $n = 5$

..., which are excellent and comparable to those of the m-sequence. The autocorrelation is 127,-5,3,-5,-5,-9, 35, ..., which is not as good as the m-sequence because the largest sidelobe 35 leads to an MSR of 0.28 as compared to 0.01 for an m-sequence. However, it still satisfies the MSR criterion. The corresponding CA sequence with $SV = 4$ has $MSR = 0.31$, which does not satisfy the criterion.

6. For $n = 8$, Table 3.35 gives the results for the m-sequence generated with $LR = 93$ and $SV = 30$ and the CA sequence generated with single replacement of $RR = 225$ at $RC = 4$. The linear complexity of the CA sequence is 127, which is close to $2^n/2 = 128$. The balance is 1 and the run is 56,30,18,7,6,2 ..., which are excellent and close to those of the m-sequence. The autocorrelation

LR	RC	RR
37	3	86
39	2	186
45	3	86
46	2	86
55	2	186
62	2	86

Table 3.19: Final Filtered Parameters LR, RC and RR for $n = 6$

is 255,15,-1,-1,11,39, ... with $MSR = 0.17$, which is good, but not as good as that of the m-sequence (0.005). Similar results were obtained for the sequences produced with all other values of SV.

3.7 Results for $n > 8$

Table 3.36 shows the results for $n = 9$ produced with parameters $LR = 305$, $RC = 8$ and $RR = 163$. $LR = 305$ was chosen based on the primitive polynomial $x^9 + x^5 + 1$. The linear complexity (LC) is between 251 and 257, which is similar to $2^n/2 = 256$. P_B and P_R have a minimum value of 0.32 for all SV, and the maximum MSR is 0.22, so the filtering criteria are satisfied, as predicted for odd-sized CAs in Section 3.5.

Table 3.37 shows the results for $n = 10$ produced with parameters $LR = 1008$, $RC = 4$ and $RR = 154$. $LR = 1008$ was chosen based on the primitive polynomial $x^{10} + x^7 + 1$. The linear complexity (LC) is between 510 and 515, which is similar to $2^n/2 = 512$. P_B has a minimum value of 0.22 and P_R is greater than 0.5 for all SV. The maximum MSR is 0.15 for all SV. Thus, all the filtering criteria are satisfied as predicted for even-sized CAs in Section 3.5.

Table 3.38 shows the results for $n = 11$ produced with parameters $LR = 706$, $RC = 10$ and $RR = 86$. $LR = 706$ was chosen based on the primitive polynomial $x^{11} + x^2 + 1$. The linear complexity (LC) is between 1023 and 1027, which is similar to $2^n/2 = 1024$. P_B and P_R are greater than 0.9 for all SV, and the maximum MSR is 0.16, so the filtering criteria are satisfied, as predicted for odd-sized CAs in Section 3.5.

Table 3.39 shows the results for $n = 12$ produced with parameters $LR = 634$,

$RC = 3$ and $RR = 99$. $LR = 634$ was chosen based on the primitive polynomial $x^{12} + x^7 + x^3 + x + 1$. The linear complexity (LC) is between 2045 and 2049, which is similar to $2^n/2 = 2048$. P_B has a minimum value of 0.3 and P_R is greater than 0.9 for all SV. The MSR is 0.23 for all SV. Thus, all the filtering criteria are satisfied as predicted for even-sized CAs in Section 3.5.

3.8 Execution Time

The 1D CA evaluation system was executed on a computer with an Intel i5-4210U (2 cores) CPU @ 1.70GHz, 4 GB RAM and a 64-bit Linux Ubuntu 14.04 operating system. Table 3.40 gives the execution times in seconds for $n = 3, 4, 5$ and 6. The execution time for size n is dependant on the number of iterations, which are also given in the table. The search for filtered results for these values of n was done with all 2^n LRs (2.1). Table 3.41 gives the execution times for $n = 7$ and 8. The search for filtered results was done only for LRs based on primitive polynomials. There are 2 LRs for every primitive polynomial [11] and $\phi(2^n - 1)/n$ primitive polynomials of degree n, where $\phi()$ is Euler's totient function [16]. The number of iterations is then

$$2 \times \phi(2^n - 1)/n \times 2^n \times n \times 248 \times (n - 2)$$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	4	91	00011	00000	16	0.37	0.68	0.22	0000111...
2	4	91	00011	00001	16	0.86	0.58	0.22	0101011...
2	4	91	00011	00010	16	0.21	0.79	0.22	0001110...
2	4	91	00011	00011	16	0.59	0.55	0.22	1010110...
2	4	91	00011	00100	16	0.1	0.48	0.22	1101111...
2	4	91	00011	00101	16	0.1	0.94	0.22	0011101...
2	4	91	00011	00110	16	0.59	0.82	0.22	0101100...
2	4	91	00011	00111	16	0.37	0.97	0.22	1011000...
2	4	91	00011	01000	16	0.59	0.55	0.22	1010110...
2	4	91	00011	01001	16	0.1	0.75	0.22	0111011...
2	4	91	00011	01010	16	0.1	0.27	0.22	1011111...
2	4	91	00011	01011	16	0.1	0.27	0.35	1111101...
2	4	91	00011	01100	16	0.59	0.9	0.22	0110000...
2	4	91	00011	01101	16	0.2	0.21	0.22	1110101...
2	4	91	00011	01110	16	0.37	0.68	0.22	0000111...
2	4	91	00011	01111	16	0.37	0.97	0.22	1011000...
2	4	91	00011	10000	16	0.1	0.48	0.22	1101111...
2	4	91	00011	10001	16	0.2	0.38	0.22	0111110...
2	4	91	00011	10010	16	0.1	0.27	0.35	1111010...
2	4	91	00011	10011	16	0.37	0.97	0.22	1000011...
2	4	91	00011	10100	16	0.37	0.68	0.22	0000111...
2	4	91	00011	10101	16	0.59	0.34	0.22	1010101...
2	4	91	00011	10110	16	0.59	0.82	0.22	0101100...
2	4	91	00011	10111	16	0.05	0.55	0.35	1110111...
2	4	91	00011	11000	16	0.59	0.9	0.22	0110000...
2	4	91	00011	11001	16	0.1	0.75	0.22	0111011...
2	4	91	00011	11010	16	0.1	0.94	0.22	0011101...
2	4	91	00011	11011	16	0.2	0.79	0.22	0001110...
2	4	91	00011	11100	16	0.59	0.55	0.22	1010110...
2	4	91	00011	11101	16	0.37	0.28	0.22	1101010...
2	4	91	00011	11110	16	0.37	0.97	0.22	1100001...
2	4	91	00011	11111	16	0.05	0.55	0.35	1110111...

Table 3.20: Results for $LR = 3$ $RC = 4$ and $RR = 91$ for $n = 5$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	2	86	101110	000001	32	0.9	0.7	0.17	0001010...
2	2	86	101110	000010	32	0.9	0.7	0.17	0010100...
2	2	86	101110	000011	32	0.9	0.7	0.17	0011011...
2	2	86	101110	000100	33	0.9	0.7	0.17	0111110...
2	2	86	101110	000101	32	0.9	0.7	0.17	0110111...
2	2	86	101110	000110	31	0.9	0.7	0.17	0100110...
2	2	86	101110	000111	32	0.9	0.7	0.17	0101001...
2	2	86	101110	001000	32	0.9	0.9	0.17	1000001...
2	2	86	101110	001001	31	0.9	0.9	0.17	1001101...
2	2	86	101110	001010	32	0.9	0.9	0.17	1011001...
2	2	86	101110	001011	32	0.9	0.9	0.17	1010010...
2	2	86	101110	001100	32	0.9	0.9	0.17	1101110...
2	2	86	101110	001101	35	0.9	0.9	0.17	1100111...
2	2	86	101110	001110	32	0.9	0.9	0.17	1111101...
2	2	86	101110	001111	32	0.9	0.9	0.17	1110001...
2	2	86	101110	010000	35	0.9	0.7	0.17	1001111...
2	2	86	101110	010001	32	0.9	0.7	0.17	1000110...
2	2	86	101110	010010	32	0.9	0.7	0.17	1011100...
2	2	86	101110	010011	31	0.9	0.7	0.17	1010000...
2	2	86	101110	010100	32	0.9	0.7	0.17	1110001...
2	2	86	101110	010101	32	0.9	0.7	0.17	1111011...
2	2	86	101110	010110	32	0.9	0.7	0.17	1101111...
2	2	86	101110	010111	32	0.9	0.7	0.17	1100011...
2	2	86	101110	011000	29	0.9	0.9	0.17	0101100...
2	2	86	101110	011001	32	0.9	0.9	0.17	0100100...
2	2	86	101110	011010	33	0.9	0.9	0.17	0111110...
2	2	86	101110	011011	31	0.9	0.9	0.17	0110010...
2	2	86	101110	011100	31	0.9	0.9	0.17	0000010...
2	2	86	101110	011101	31	0.9	0.9	0.17	0001110...
2	2	86	101110	011110	31	0.9	0.9	0.17	0011010...
2	2	86	101110	011111	32	0.9	0.9	0.17	0010000...

Table 3.21: Results for $LR = 46$, $RC = 2$ and $RR = 86$ for $n = 6$ (Part 1)

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	2	86	101110	100000	31	0.9	0.9	0.17	1101011...
2	2	86	101110	100001	31	0.9	0.9	0.17	1100100...
2	2	86	101110	100010	31	0.9	0.9	0.17	1110100...
2	2	86	101110	100011	33	0.9	0.9	0.17	1111100...
2	2	86	101110	100100	35	0.9	0.9	0.17	1011001...
2	2	86	101110	100101	32	0.9	0.9	0.17	1010110...
2	2	86	101110	100110	31	0.9	0.9	0.17	1001001...
2	2	86	101110	100111	32	0.9	0.9	0.17	1000010...
2	2	86	101110	101000	32	0.9	0.7	0.17	0011101...
2	2	86	101110	101001	32	0.9	0.7	0.17	0010101...
2	2	86	101110	101010	33	0.9	0.7	0.17	0000101...
2	2	86	101110	101011	32	0.9	0.7	0.17	0001010...
2	2	86	101110	101100	32	0.9	0.7	0.17	0111010...
2	2	86	101110	101101	32	0.9	0.7	0.17	0110101...
2	2	86	101110	101110	31	0.9	0.7	0.17	0101011...
2	2	86	101110	101111	32	0.9	0.7	0.17	0100001...
2	2	86	101110	110000	30	0.9	0.7	0.17	1010110...
2	2	86	101110	110001	31	0.9	0.7	0.17	1011111...
2	2	86	101110	110010	32	0.9	0.7	0.17	1001000...
2	2	86	101110	110011	32	0.9	0.7	0.17	1000111...
2	2	86	101110	110100	32	0.9	0.7	0.17	1110111...
2	2	86	101110	110101	32	0.9	0.7	0.17	1111000...
2	2	86	101110	110110	32	0.9	0.7	0.17	1100011...
2	2	86	101110	110111	31	0.9	0.7	0.17	1101000...
2	2	86	101110	111000	34	0.9	0.9	0.17	0011111...
2	2	86	101110	111001	32	0.9	0.9	0.17	0010011...
2	2	86	101110	111010	32	0.9	0.9	0.17	0001101...
2	2	86	101110	111011	32	0.9	0.9	0.17	0000101...
2	2	86	101110	111100	31	0.9	0.9	0.17	0101100...
2	2	86	101110	111101	31	0.9	0.9	0.17	0100000...
2	2	86	101110	111110	35	0.9	0.9	0.17	0110011...
2	2	86	101110	111111	32	0.9	0.9	0.17	0111000...

Table 3.22: Results for $LR = 46$, $RC = 2$ and $RR = 86$ for $n = 6$ (Part 2)

n	LR	Primitive Polynomial	RC	Duplicate LR
3	1	$x^3 + x^2 + 1$	2	3
	6	$x^3 + x + 1$	2	4
4	10	$x^4 + x + 1$	3	8
	11	$x^4 + x^3 + 1$	3	9
5	1	$x^5 + x^4 + x^2 + x + 1$	4	3
	7	$x^5 + x^4 + x^3 + x + 1$	4	5
	16	$x^5 + x^4 + x^2 + x + 1$	2	24
6	37	$x^6 + x^5 + x^2 + x + 1$	3	45
	39	$x^6 + x^4 + x^2 + x + 1$	2	55
	46	$x^6 + x^4 + x^3 + x + 1$	2	62

Table 3.23: Filtered Parameters, Associated Primitive Polynomials and Duplicate LRs

Primitive Polynomial	LR	RC	RR
$x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$	14	3	18
$x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$	33	3	58
		6	50
$x^7 + x^4 + x^3 + x^2 + 1$	43	6	18, 122, 182
$x^7 + x^3 + x^2 + x + 1$	71	5	154
		6	122, 182
$x^7 + x + 1$	77	6	126
$x^7 + x^6 + x^5 + x^2 + 1$	84	4	30

Table 3.24: Filtered Parameters LR, RC and RR for $n = 7$

Primitive Polynomial	LR	RC	RR
$x^8 + x^7 + x^6 + x + 1$	93	4	225
		7	225
$x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$	201	2	182, 186

Table 3.25: Filtered Parameters LR, RC and RR for $n = 8$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	6	18	0101011	0000001	64	0.93	0.66	0.29	1001101...
2	6	18	0101011	0000010	65	0.93	0.32	0.31	1001110...
2	6	18	0101011	0000011	64	0.93	0.32	0.29	0011011...
2	6	18	0101011	0000100	64	0.93	0.66	0.31	0110110...
2	6	18	0101011	0000101	64	0.93	0.32	0.31	0011100...
2	6	18	0101011	0000110	64	0.93	0.32	0.28	1000001...
2	6	18	0101011	0000111	64	0.79	0.32	0.28	1011011...
2	6	18	0101011	0001000	64	0.93	0.94	0.29	0010100...
2	6	18	0101011	0001001	63	0.93	0.94	0.29	0111000...
2	6	18	0101011	0001010	64	0.93	0.66	0.31	1101100...
2	6	18	0101011	0001011	64	0.79	0.32	0.29	0001000...
2	6	18	0101011	0001100	64	0.79	0.32	0.28	0110111...
2	6	18	0101011	0001101	63	0.93	0.32	0.28	0000010...
2	6	18	0101011	0001110	64	0.79	0.42	0.29	0101010...
2	6	18	0101011	0001111	64	0.66	0.66	0.29	0001111...
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	6	18	0101011	1110000	64	0.93	0.53	0.29	1000101...
2	6	18	0101011	1110001	64	0.53	0.76	0.29	1011001...
2	6	18	0101011	1110010	67	0.79	0.53	0.24	0100111...
2	6	18	0101011	1110011	64	0.79	0.42	0.29	0010110...
2	6	18	0101011	1110100	67	0.79	0.42	0.28	1010011...
2	6	18	0101011	1110101	65	0.66	0.64	0.29	0101100...
2	6	18	0101011	1110110	63	0.79	0.78	0.29	0010110...
2	6	18	0101011	1110111	61	0.66	0.52	0.28	1101001...
2	6	18	0101011	1111000	64	0.93	0.32	0.28	0001010...
2	6	18	0101011	1111001	65	0.66	0.32	0.28	0111100...
2	6	18	0101011	1111010	63	0.93	0.32	0.28	1110110...
2	6	18	0101011	1111011	64	0.53	0.9	0.29	0110010 ...
2	6	18	0101011	1111100	64	0.79	0.32	0.29	0101101...
2	6	18	0101011	1111101	67	0.79	0.66	0.24	1001111...
2	6	18	0101011	1111110	65	0.93	0.93	0.31	0001110...
2	6	18	0101011	1111111	64	0.93	0.52	0.29	0101010...

Table 3.26: Results for $LR = 43$, $RC = 6$ and $RR = 18$ for $n = 7$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	4	225	01011101	00000000	130	0.49	0.43	0.22	0001100
2	4	225	01011101	00000001	128	0.66	0.96	0.18	0111011
2	4	225	01011101	00000010	127	0.42	0.37	0.18	1100001
2	4	225	01011101	00000011	127	0.42	0.37	0.20	1011111
2	4	225	01011101	00000100	127	0.42	0.37	0.17	1000110
2	4	225	01011101	00000101	127	0.42	0.37	0.25	1000011
2	4	225	01011101	00000110	130	0.49	0.43	0.22	0011101
2	4	225	01011101	00000111	129	0.75	0.42	0.18	0010100
2	4	225	01011101	00001000	127	0.42	0.37	0.18	0011000
2	4	225	01011101	00001001	129	0.75	0.42	0.17	1010001
2	4	225	01011101	00001010	130	0.49	0.43	0.17	0101000
2	4	225	01011101	00001011	125	0.85	0.66	0.18	0010100
2	4	225	01011101	00001100	128	0.66	0.96	0.18	0110000
2	4	225	01011101	00001101	128	0.66	0.96	0.18	0011101
2	4	225	01011101	00001110	126	0.66	0.55	0.18	1001010
2	4	225	01011101	00001111	128	0.66	0.96	0.17	0100011
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	4	225	01011101	11110000	128	0.66	0.96	0.23	1100101
2	4	225	01011101	11110001	128	0.66	0.96	0.18	0001100
2	4	225	01011101	11110010	128	0.66	0.96	0.18	0000001
2	4	225	01011101	11110011	128	0.66	0.96	0.18	1111110
2	4	225	01011101	11110100	129	0.75	0.42	0.23	1110110
2	4	225	01011101	11110101	124	0.57	0.32	0.20	1001010
2	4	225	01011101	11110110	128	0.66	0.96	0.18	0111110
2	4	225	01011101	11110111	128	0.66	0.96	0.18	1100101
2	4	225	01011101	11111000	127	0.42	0.37	0.18	1101100
2	4	225	01011101	11111001	127	0.42	0.37	0.22	1000001
2	4	225	01011101	11111010	128	0.66	0.96	0.22	1011101
2	4	225	01011101	11111011	127	0.42	0.37	0.18	1110111
2	4	225	01011101	11111100	127	0.42	0.37	0.22	0001100
2	4	225	01011101	11111101	129	0.75	0.42	0.23	1011001
2	4	225	01011101	11111110	128	0.66	0.96	0.18	1101100
2	4	225	01011101	11111111	128	0.66	0.96	0.20	0111011

Table 3.27: Results for $LR = 93$, $RC = 4$ and $RR = 225$ for $n = 8$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	110	001	3	1	0.7	2,1,1,0,0,0, ...	0.66	7,-1,-1,-1,-1,-1,-1,-1	0.14	1110010
2	2	183	110	001	4	3	0.26	3,2,0,0,0,0, ...	0.05	7,-1,3,-1,-1,3,-1	0.43	1101011

Table 3.28: Comparison of an m-sequence with the CA for $n = 3$, $LR = 6$, $SV = 1$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	110	100	3	1	0.7	2,1,1,0,0,0, ...	0.66	7,-1,-1,-1,-1,-1,-1,-1	0.14	1011100
2	2	183	110	100	3	1	0.7	4,0,1,0,0,0, ...	0.22	7,-1,-1,-1,-1,-1,-1,-1	0.14	0111010

Table 3.29: Comparison of an m-sequence with the CA for $n = 3$, $LR = 6$, $SV = 4$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	1010	0010	4	1	0.8	4,2,1,1,0,0, ...	0.78	15,-1,-1,-1,-1,-1,-1,-1,...	0.07	1101010...
2	3	86	1010	0010	8	1	0.8	4,1,3,0,0,0, ...	0.78	15,-1,-1,-3,3,3,3,...	0.2	1110001...

Table 3.30: Comparison of an m-sequence with the CA for $n = 4$, $LR = 10$, $SV = 4$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	00111	00001	5	1	0.86	8,6,2,0,1,0, ...	0.59	31,-1,-1,-1,-1,-1,-1,-1, ...	0.03	0011010...
2	4	107	00111	00001	16	3	0.79	6,3,2,2,1,0, ...	0.62	31,3,-1,-9,-5,-1,3, ...	0.29	1110000...

Table 3.31: Comparison of an m-sequence with the CA for $n = 5$, $LR = 7$, $SV = 1$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	00111	00010	5	1	0.86	8,6,2,0,1,0, ...	0.58	31,-1,-1,-1,-1,-1,-1,-1, ...	0.03	0101010...
2	4	107	00111	00010	15	5	0.37	5,4,3,1,1,0, ...	0.68	31,3,-5,-11,-1,11,7, ...	0.35	1110011...

Table 3.32: Comparison of an m-sequence with the CA for $n = 5$, $LR = 7$, $SV = 2$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	101101	000111	6	1	0.9	18,7,4,2,1,1,...	0.7	63,-1,-1,-1,-1,-1,-1,-1, ...	0.02	0101010...
2	3	86	101101	000111	32	1	0.9	18,7,4,2,1,1, ...	0.7	63,-1,-1,-1,-1,-1,-1,7, ...	0.17	0111111...

Table 3.33: Comparison of an m-sequence with the CA for $n = 6$, $LR = 45$, $SV = 7$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	0101011	0000110	7	1	0.92	32,18,8,3,2,1, ...	0.79	127,-1,-1,-1,-1,-1,-1,-1, ...	0.01	0011110...
2	6	18	0101011	0000110	64	-1	0.93	37,15,7,2,5,1, ...	0.32	127,-5,3,-5,-5,-9, 35, ...	0.28	1000001...

Table 3.34: Comparison of an m-sequence with the CA for $n = 7$, $LR = 43$, $SV = 6$

OC	RC	RR	LR	SV	LC	B	P_B	R	P_R	AC	MSR	S
2	NA	NA	01011101	00011110	8	1	0.95	66,31,16,8,4,2 ...	0.85	255,-1,-1,-1,-1,-1,-1,-1, ...	0.005	0011110...
2	4	225	01011101	00011110	127	1	0.95	56,30,18,7,6,2 ...	0.82	255,15,-1,-1,11,39, ...	0.17	0001100...

Table 3.35: Comparison of an m-sequence with the CA for $n = 8$, $LR = 93$, $SV = 30$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	8	163	100110001	000000000	256	0.89	0.32	0.12	1000101...
2	8	163	100110001	000000001	253	0.76	0.51	0.12	0011110...
2	8	163	100110001	000000010	256	0.82	0.32	0.21	0001011...
2	8	163	100110001	000000011	252	0.76	0.57	0.12	0111101...
2	8	163	100110001	000000100	251	0.69	0.57	0.12	1111011...
2	8	163	100110001	000000101	256	0.82	0.32	0.11	0010111...
2	8	163	100110001	000000110	254	0.82	0.69	0.12	1111001...
2	8	163	100110001	000000111	256	0.56	0.98	0.10	1100000...
2	8	163	100110001	000001000	250	0.76	0.63	0.12	1110111...
2	8	163	100110001	000001001	256	0.76	0.32	0.10	1010011...
2	8	163	100110001	000001010	257	0.69	0.9	0.13	1110101...
2	8	163	100110001	000001011	256	0.82	0.32	0.21	0101111...
2	8	163	100110001	000001100	255	0.82	0.32	0.13	0000100...
2	8	163	100110001	000001101	254	0.82	0.69	0.12	1110010...
2	8	163	100110001	000001110	256	0.63	0.96	0.11	1000001...
2	8	163	100110001	000001111	255	0.45	0.32	0.10	1010010...
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	8	163	100110001	111110000	255	0.96	0.32	0.10	1111000...
2	8	163	100110001	111110001	256	0.63	0.32	0.10	0101101...
2	8	163	100110001	111110010	257	0.69	0.32	0.10	1001010...
2	8	163	100110001	111110011	256	0.63	0.32	0.22	0010100...
2	8	163	100110001	111110100	256	0.63	0.95	0.22	0100001...
2	8	163	100110001	111110101	256	0.57	0.32	0.22	0101101...
2	8	163	100110001	111110110	255	0.57	0.32	0.12	1001010...
2	8	163	100110001	111110111	255	0.32	0.44	0.12	1001011...
2	8	163	100110001	111111000	256	0.69	0.57	0.12	1010001...
2	8	163	100110001	111111001	256	0.63	0.7	0.12	0111011...
2	8	163	100110001	111111010	256	0.63	0.32	0.12	1000111...
2	8	163	100110001	111111011	256	0.63	0.83	0.12	1100111...
2	8	163	100110001	111111100	256	0.76	0.32	0.10	1000100...
2	8	163	100110001	111111101	255	0.96	0.32	0.11	1000010...
2	8	163	100110001	111111110	257	0.69	0.44	0.11	0101001...
2	8	163	100110001	111111111	255	0.63	0.45	0.12	1011001...

Table 3.36: Results for $LR = 305$, $RC = 8$ and $RR = 163$ for $n = 9$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	4	154	1111110000	0000000001	513	0.33	0.53	0.10	1001001...
2	4	154	1111110000	0000000010	512	0.33	0.57	0.10	0010010...
2	4	154	1111110000	0000000011	512	0.27	0.75	0.10	0011011...
2	4	154	1111110000	0000000100	512	0.51	0.72	0.10	0100110...
2	4	154	1111110000	0000000101	511	0.3	0.57	0.10	0100100...
2	4	154	1111110000	0000000110	515	0.33	0.75	0.10	1001011...
2	4	154	1111110000	0000000111	512	0.25	0.75	0.10	0110111...
2	4	154	1111110000	0000001000	512	0.27	0.61	0.10	1001001...
2	4	154	1111110000	0000001001	510	0.68	0.93	0.11	0010111...
2	4	154	1111110000	0000001010	512	0.51	0.77	0.10	1001100...
2	4	154	1111110000	0000001011	511	0.6	0.63	0.11	1000100...
2	4	154	1111110000	0000001100	513	0.22	0.7	0.10	0011011...
2	4	154	1111110000	0000001101	511	0.25	0.75	0.10	1101111...
2	4	154	1111110000	0000001110	512	0.73	0.97	0.11	1111000...
2	4	154	1111110000	0000001111	514	0.36	0.8	0.10	0010111...
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	4	154	1111110000	1111110000	511	0.47	0.81	0.10	1111111...
2	4	154	1111110000	1111110001	513	0.68	0.73	0.15	1000101...
2	4	154	1111110000	1111110010	513	0.36	0.8	0.10	0111011...
2	4	154	1111110000	1111110011	512	0.55	0.63	0.10	1110001...
2	4	154	1111110000	1111110100	512	0.47	0.54	0.10	0111010...
2	4	154	1111110000	1111110101	512	0.3	0.7	0.10	1010100...
2	4	154	1111110000	1111110110	512	0.73	0.68	0.10	0001000...
2	4	154	1111110000	1111110111	512	0.73	0.88	0.11	1101011...
2	4	154	1111110000	1111111000	513	0.68	0.68	0.10	1101101...
2	4	154	1111110000	1111111001	513	0.33	0.71	0.10	1010000...
2	4	154	1111110000	1111111010	511	0.6	0.97	0.10	0111001...
2	4	154	1111110000	1111111011	510	0.98	0.98	0.15	1100110...
2	4	154	1111110000	1111111100	511	0.68	0.82	0.10	0001111...
2	4	154	1111110000	1111111101	512	0.33	0.75	0.10	1101001...
2	4	154	1111110000	1111111110	512	0.33	0.75	0.10	0000011...
2	4	154	1111110000	1111111111	511	0.33	0.66	0.10	0101001...

Table 3.37: Results for $LR = 1008$, $RC = 4$ and $RR = 154$ for $n = 10$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	10	86	01011000010	00000000001	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000000010	1023	0.9	0.95	0.13	0000000...
2	10	86	01011000010	00000000011	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000000100	1023	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000000101	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000000110	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000000111	1022	0.9	0.95	0.16	0000000...
2	10	86	01011000010	00000001000	1024	0.98	0.95	0.06	0000001...
2	10	86	01011000010	00000001001	1024	0.98	0.95	0.06	0000001...
2	10	86	01011000010	00000001010	1024	0.98	0.95	0.06	0000011...
2	10	86	01011000010	00000001011	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000001100	1023	0.98	0.95	0.13	0000010...
2	10	86	01011000010	00000001101	1026	0.98	0.95	0.06	0000010...
2	10	86	01011000010	00000001110	1024	0.98	0.95	0.06	0000000...
2	10	86	01011000010	00000001111	1025	0.9	0.95	0.06	0000000...
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	10	86	01011000010	11111110000	1024	0.98	0.95	0.06	1011111...
2	10	86	01011000010	11111110001	1023	0.9	0.95	0.16	1011111...
2	10	86	01011000010	11111110010	1025	0.98	0.95	0.08	1011111...
2	10	86	01011000010	11111110011	1024	0.98	0.95	0.06	1011111...
2	10	86	01011000010	11111110100	1023	0.98	0.95	0.06	1011110...
2	10	86	01011000010	11111110101	1025	0.98	0.95	0.06	1011110...
2	10	86	01011000010	11111110110	1025	0.98	0.95	0.06	1011110...
2	10	86	01011000010	11111110111	1023	0.9	0.95	0.16	1011110...
2	10	86	01011000010	11111111000	1025	0.9	0.95	0.05	1011100...
2	10	86	01011000010	11111111001	1025	0.9	0.95	0.05	1011100...
2	10	86	01011000010	11111111010	1024	0.98	0.95	0.06	1011100...
2	10	86	01011000010	11111111011	1027	0.95	0.95	0.08	1011100...
2	10	86	01011000010	11111111100	1023	0.9	0.95	0.16	1011100...
2	10	86	01011000010	11111111101	1024	0.98	0.95	0.06	1011100...
2	10	86	01011000010	11111111110	1024	0.98	0.95	0.06	1011101...
2	10	86	01011000010	11111111111	1022	0.9	0.95	0.16	1011101...

Table 3.38: Results for $LR = 706$, $RC = 10$ and $RR = 86$ for $n = 11$

OC	RC	RR	LR	SV	LC	P_B	P_R	MSR	S
2	3	99	001001111010	000000000000	2046	0.3	0.98	0.23	1011111...
2	3	99	001001111010	000000000001	2049	0.98	0.92	0.23	1111110...
2	3	99	001001111010	000000000010	2048	0.3	0.98	0.23	1000101...
2	3	99	001001111010	000000000011	2047	0.45	0.9	0.23	1000100...
2	3	99	001001111010	000000000100	2047	0.96	0.96	0.23	1011111...
2	3	99	001001111010	000000000101	2048	0.32	0.98	0.23	0110001...
2	3	99	001001111010	000000000110	2045	0.3	0.98	0.23	1000101...
2	3	99	001001111010	000000000111	2049	0.45	0.9	0.23	1000100...
2	3	99	001001111010	000000001000	2046	0.3	0.98	0.23	1011111...
2	3	99	001001111010	000000001001	2049	0.98	0.92	0.23	1111110...
2	3	99	001001111010	000000001010	2048	0.3	0.98	0.23	1000101...
2	3	99	001001111010	000000001011	2047	0.45	0.9	0.23	1000100...
2	3	99	001001111010	000000001100	2047	0.96	0.96	0.23	1011111...
2	3	99	001001111010	000000001101	2048	0.32	0.98	0.23	0110001...
2	3	99	001001111010	000000001110	2045	0.3	0.98	0.23	1000101...
2	3	99	001001111010	000000001111	2049	0.45	0.9	0.23	1000100...
⋮	⋮	⋮	⋮	⋮	⋮				⋮
2	3	99	001001111010	111111110000	2046	0.3	0.98	0.23	1011111...
2	3	99	001001111010	111111110001	2049	0.98	0.92	0.23	1111110...
2	3	99	001001111010	111111110010	2048	0.3	0.98	0.23	1000101...
2	3	99	001001111010	111111110011	2047	0.45	0.9	0.23	1000100...
2	3	99	001001111010	111111110100	2047	0.96	0.96	0.23	1011111...
2	3	99	001001111010	111111110101	2048	0.32	0.98	0.23	0110001...
2	3	99	001001111010	111111110110	2045	0.3	0.98	0.23	1000101...
2	3	99	001001111010	111111110111	2048	0.95	0.9	0.23	1000100...
2	3	99	001001111010	111111111000	2049	0.45	0.9	0.23	1000100...
2	3	99	001001111010	111111111001	2046	0.3	0.98	0.23	1011111...
2	3	99	001001111010	111111111010	2049	0.98	0.92	0.23	1111110...
2	3	99	001001111010	111111111011	2048	0.3	0.98	0.23	1000101...
2	3	99	001001111010	111111111100	2047	0.45	0.9	0.23	1000100...
2	3	99	001001111010	111111111101	2047	0.96	0.96	0.23	1011111...
2	3	99	001001111010	111111111110	2048	0.32	0.98	0.23	0110001...
2	3	99	001001111010	111111111111	2045	0.3	0.98	0.23	1000101...

Table 3.39: Results for $LR = 634$, $RC = 3$ and $RR = 99$ for $n = 12$

n	Iterations	Time (s)
3	47616	4.64
4	507904	64.51
5	3809280	756.79
6	24379392	6914.15

Table 3.40: Execution Times for $n = 3, 4, 5$ and 6

n	Iterations	Time (s)
7	39997440	16257.02
8	97517568	54499.74

Table 3.41: Execution Times for $n = 7$ and 8

Chapter 4

Conclusions

The objective of this thesis was to use cellular automata (CA) to produce pseudo-random sequences with high linear complexity and good randomness. An evaluation system was developed to test sequences generated by one dimensional (1D) CAs which contain combinations of linear rules (rules 90 and 150) with single cells replaced by a non-linear rule (rules 0 to 255, except 0, 60, 90, 102, 150, 170, 204 and 240). Initially, an extensive investigation was done on CAs of sizes $n = 3, 4, 5$ and 6. A two stage process was implemented for filtering sequences generated by these CAs. The first stage involved filtering sequences that had a linear complexity of $LC \geq n$ for $n = 3$ and 4, and $LC \geq 2^n/4$ for $n > 4$, irrespective of the initial state (SV) of the CA. In the second stage, the filtered sequences from the first stage were further filtered based on the criteria for balance, run and autocorrelation (randomness properties). The criteria for balance and run were $P_B > 0.01$ and run $P_R > 0.01$, respectively for all values of SV. P_B and P_R are complementary error functions calculated using the sequences. The calculations of P_B and P_R were adopted from the statistical test suite for random and pseudorandom number generators for cryptographic applications by the National Institute of Standards and Technology (NIST) and the U.S. Department of Commerce [13]. For autocorrelation, the filtering criterion was a maximum sidelobe ratio of $MSR < 0.3$. MSR is the ratio of the largest sidelobe of the autocorrelation to the mainlobe.

The results of the filtering process indicated that for even-sized CA, linear rules based on m-sequences can always be combined with a non-linear rule to generate sequences that satisfy all the filtering criteria. However, odd-sized CA with a similar combination of rules generate sequences that satisfy the filtering criteria for linear

complexity, balance and run, but do not always satisfy the criterion for autocorrelation as with some SV the max MSR was greater than 0.3. This result was also obtained with the filtered sequences of CA sizes $n = 7$ and 8. The CA sequences for $n = 3, 4, 5, 6, 7$ and 8 were then compared with m-sequences. The linear complexity of the CA sequences was approximately $2^{n/2}$, which is high compared to the linear complexity of n for m-sequences. The balance and run for both even and odd CAs were close to those of the m-sequences. The autocorrelation for even-sized CAs satisfied the filtering criterion, but was not as good as that of the m-sequences, as expected. For odd-sized CAs, the MSR did not satisfy the filtering criterion for all values of SV. However, the maximum MSR decreased with increasing n. For $n = 9$ and 11, $MSR < 0.3$ was satisfied for all values of SV, and all the filtering criteria were satisfied. In conclusion, CAs can be used to generate pseudorandom sequences of high linear complexity and good randomness.

4.1 Future Work

Cellular automata for pseudorandom sequence generation has good scope for future work. First, the sequences generated by the 1D CA evaluation system can be tested for other properties of randomness such as closure, recurrence, window and shift [4]. To do this, the algorithms for these properties will need to be incorporated as additional modules into the evaluation system, similar to those for balance, run and autocorrelation. The filtering criteria can be designed using the tests given in [13]. For example, the non-overlapping and overlapping template matching tests provide good methods for filtering sequences with respect to the window property.

This thesis analyzed sequences generated by a single replacement of a linear rule by a non-linear rule. Multiple non-linear rule replacements can be considered in the future. The evaluation system will need to be modified. Specifically, the parameters in the evaluation system associated with the non-linear rules (RR) and their respective cell positions (RC) will need to be converted from single values to arrays to accommodate multiple replacements. A concern here is that the computational complexity for multiple replacements is on the order of z^l, where l is the number of replacements and z is the computational complexity of the single replacement algorithm. The complexity of the multiple replacement algorithm can be reduced by removing the iterations for all LRs that are duplicates.

Finally, a similar CA evaluation system can be designed for two dimensional (2D) CAs and used to analyze the linear complexity [17] and randomness of the generated sequences. The rules (linear and non-linear) for each cell in the 2D system would be one of the 2^9 Wolfram rules [8] because the neighborhood here is 4, 6 or 9 depending on if the cell is in the corner, edge or centre, respectively.

Bibliography

[1] Oxford Dictionary of English, Oxford University Press, 3rd Edition, 2010.

[2] M.G. Kendall and B.B. Smith, "Randomness and Random Sampling Numbers," *Journal of the Royal Statistical Society*, vol. 101, no. 1, pp. 147-160, 1938.

[3] A. J. Menezes *et al.*, Handbook of Applied Cryptography, CRC Press, pp. 39-40, 2001.

[4] A. Mitra, "On the Properties of Pseudo Noise Sequences with a Simple Proposal of Randomness Test," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 2, no. 9, 2008.

[5] S. Mishra, R.R. Tripathi and D.K. Tripathi, "Implementation of Configurable Linear Feedback Shift Register in VHDL," *International Conference on Emerging Trends in Electrical Electronics and Sustainable Energy Systems*, 2016.

[6] Robert Scurr, "Sequences and Cellular Automata," ENEL 427 Final Report, University of Canterbury, New Zealand, 1998.

[7] R.E. Ziemer and R.L. Peterson, Digital Communications and Spread Spectrum Systems, Macmillan Publishing Company, pp. 385-386, 1985.

[8] S. Wolfram, A New Kind of Science, Wolfram Media, 2002.

[9] S. Wolfram, Cellular Automata and Complexity: Collected Papers, Westview Press, 1st Edition, 1994.

[10] S. Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics*, vol. 7, pp. 123-169, 1986.

[11] K. Cattell and J.C. Muzio, "Synthesis of One Dimensional Linear Hybrid Cellular Automata," *IEEE Transactions on Computer Aided Design*, vol. 15, no. 3, pp. 325-335, 1996.

[12] K. Cattell and J.C. Muzio, "Analysis of One Dimensional Linear Hybrid Cellular Automata over GF(q)," *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 782-792, 1996.

[13] A. Rukhin *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," rev. 1a, National Institute of Standards and Technology (NIST) and U.S. Department of Commerce, 2010.

[14] M. K. Simon *et al.*, Spread Spectrum Communications, Computer Science Press, vol. 1, pp. 262-353, 1985.

[15] E. Casey, "Berlekamp-Massey Algorithm," REU Summer Report, University of Minnesota, Minneapolis, MN, 2000.

[16] E. R. Berlekamp, Algebraic Coding Theory, New York: McGraw-Hill, 1968.

[17] P. Boyle, "Sequences and Cellular Automata," ENEL 427 Final Report, University of Canterbury, New Zealand, 1999.

[18] P.D. Hortensius *et al.*, "Cellular Automata-based Pseudorandom Number Generators for Built-in Self-Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 842-859, 1989.

[19] S.A. Fredricsson, "Pseudo-Randomness Properties of Binary Shift Register Sequences," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 115-120, 1975.

[20] M.L. Bushnell and V. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Springer, 2004.

[21] P.H. Bardell, "Analysis of Cellular Automata Used as Pseudorandom Pattern Generators," *Proceedings of the IEEE International Test Conference*, pp. 762-767, 1990.

[22] M. Serra *et al.*, "The Analysis of One Dimensional Linear Cellular Automata and Their Aliasing Properties," *IEEE Transactions on Computer Aided Design*, vol. 9, no. 7, pp. 767-778, 1990.