Analysis of the New Class of Cellular Automata and Its Application in VLSI Testing

by

Lin Sun

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

MASTER OF SCIENCE

in the Department of Computer Science

Supervisor: Dr. J. C. Muzio

ABSTRACT

A new class of cellular automata is introduced and its properties studied. These one-dimensional neighbor-five cellular automata are proposed as pseudorandom pattern generators to be applied in built-in self-test. Based on a theoretical analysis, the transition capability of the new class of cellular automata is larger than that of linear hybrid cellular automata. In order to evaluate their randomness properties, Knuth's randomness tests are employed; the patterns generated by the new class of cellular automata are shown to achieve a slightly better randomness than those generated by linear hybrid cellular automata and much better randomness than those generated by linear feedback shift registers. For testing combinational and sequential faults over a set of standard benchmark circuits, experimental results demonstrate that, in sequential circuits, the pseudorandom pattern generators produced by the proposed cellular automata outperform the conventional generators produced by linear hybrid cellular automata and linear feedback shift registers.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CA	Cellular Automata
LHCA	Linear Hybrid Cellular Automata
LFSR	Linear Feedback Shift Register
ALFSR	Automata Linear Feedback Register
LFSM	Linear Finite State Machine
NCCA	New Class of Cellular Automata
ICs	Integrated Circuits
VLSI	Very Large Scale Integration
BIST	Built-In Self-Test
DFT	Design for Test

Acknowledgement

I would like to thank my supervisor, Dr. Jon Muzio, for his invaluable guidance, support, patience and understanding through my time as a graduate student.

I would also like to thank my committee members Dr. John Ellis and Dr. Michael Miller, for their comments and suggestions.

I am grateful to everyone in the Digital System Design Lab for their kind help, especially for Jiexia Zhu, jing Zhong.

Finally, I would like to express my deep gratitude to my parents and my brother for their unending supporting, love and encouragement through my studies.

Chapter 1

Introduction

1.1 Motivation

Advances in Very Large Scale Integration (VLSI) technology have resulted in the ability to produce Integrated Circuits (ICs) with over one hundred million transistors on a chip. The problem of testing such complex circuits in a cost-effective way has been and remains a major concern. Built-In Self-Test (BIST) is a general approach to the testing of ICs. A widely accepted method for BIST is to use a pseudorandom pattern generator and a data compactor. It is well known that Linear Feedback Shift Registers (LFSR) are commonly used for pseudorandom pattern generators and data compactors. However, recent studies [15, 21] prove that Linear Hybrid Cellular Automata (LHCA) are superior to LFSR in VLSI testing.

LHCA are the simplest class of Cellular Automata (CA). CA were proposed first by John von Neumann in the 1940's, and were used to model self-reproducing organisms. In 1983 and 1984, Stephen Wolfram published papers [19, 20], which are milestones for studying cellular automata in the engineering and science fields. Now CA have been applied in different research fields such as VLSI testing, error correcting codes, cryptography, parallel computing and computer graphics.

Cellular automata are mathematical models for complicated natural systems. They consist of a series of identical components (also called 'cells '), each with a finite set of possible values. The value of each cell is determined by the previous value of its neighbors and/or itself. The relation of its neighbors and/or itself forms different and complex structures such as a 1-dimensional string (e.g. LHCA), a 2-dimensional grid (e.g. 2-by-n CA [7]), or a 3-dimensional structure of cells. Based on different properties of different structures, this thesis introduces a New Class of Cellular Automata (NCCA), in which the value of each cell is dependent on the previous value of the nearest four cells or dependent on the previous value of the nearest four cells and itself, as pseudorandom pattern generators in VLSI testing.

The main objective of this thesis is to evaluate the performance of this new class of

CA as pseudorandom pattern generators. This has not been done before, thus we set the notation, define their computation rules, and apply the necessary mathematical background for a complete theoretical analysis. Then we derive a recursive relation to compute the characteristic polynomial of an NCCA; analyze the transition properties of NCCA, which are used as the metrics of effectiveness of pseudorandom pattern generators for testing sequential faults, and also derive the maximum number of an NCCA's transitions. We compare NCCA with their corresponding LFSR and LHCA. All this work is accomplished through the generation of maximum-length NCCA and by performing Knuth's empirical tests [14] for evaluating the pseudorandom properties of the patterns generated from these NCCA. Furthermore, standard benchmark circuits are simulated to perform a feasibility study on the behavior of NCCA, LFSR and LHCA as pseudorandom test pattern generators.

1.2 Outline of Thesis

The main goal of this thesis is to formally introduce a New Class of CA (NCCA), to study their properties, and to compare them with the corresponding LFSR and LHCA.

In Chapter 2, the background material relevant to this thesis is briefly introduced. The definition and mathematical characteristics of Linear Finite State Machines (LFSMs) are reviewed. LFSR and CA with an emphasis on LHCA are discussed together with the notation and mathematical background. In this chapter, we also present some background on built-in self-test especially for pseudorandom pattern generators, and also discuss its applications.

In Chapter 3, a new class of cellular automata (NCCA) is proposed. Its definition and notation, as well as the computation rules for each cell are presented. Then we focus on the properties of NCCA: first, we introduce their transition matrix and characteristic polynomial; second, we derive a recursive relation to obtain the characteristic polynomial as well as the structure for the minimum-cost primitive NCCA for degrees 1 through 100; third, we analyze the transition properties of NCCA and compare them with those of LHCA; this

chapter concludes with a discussion of the application of NCCA in built-in self-test.

In Chapter 4, Knuth's empirical (or randomness) tests [14] are introduced. We employ these tests on the sequences generated by the maximum-length LFSR, LHCA and NCCA in order to compare their randomness, and the experimental results are presented.

In Chapter 5, the standard benchmark circuits ISCAS'85 [11] and ISCAS'89 [9, 10] are also briefly introduced. We perform a feasibility study on the performance of the pseudorandom pattern generators produced by the maximum-length NCCA, compared to those based on the corresponding maximum-length LFSR and LHCA by simulating a standard benchmark circuit with the generators and evaluating their fault coverage. Experimental results are presented and analyzed.

In Chapter 6, the main results in this thesis are summarized and possible future work is discussed.

Chapter 2

Linear Finite State Machines

In this chapter, several LFSMs are introduced. The important concepts about LFSM are described, which include the notation used and the mathematical background, such as the definition of linear finite state machines, the characteristic polynomial of a LFSM and primitive polynomials, etc.

One of the most frequently used forms of LFSM is LFSR. Type I and Type II LFSR are introduced in section 2.2.

Another important form of LFSMs, LHCA are introduced. First, the general Cellular Automata (CA) are introduced, including their space structure and neighborhood relation, then a special linear CA, LHCA, are discussed. In Chapter 3, a new class of CA is introduced, and the transition properties of binary sequences produced by LHCA and NCCA are analyzed in greater detail.

Finally, some fundamental concepts of built-in self-test, with an emphasis on pseudo-random testing are discussed.

2.1 Definitions

The general definitions of LFSM and some of mathematical characteristics are introduced below:

Definition 2.1.1 [18] A machine M is a linear finite state machine if:

1) the state space S_M of M, the input space I_M, and the output space Y_M are each vector spaces over the appropriate finite field (here a Galois Field of order 2, GF(2)).

2) let the vector q_i denote the current state of the machine, the vector u_i denote the inputs to the machine, and the vector y_i denote the outputs of the machine. The next state q_i^+ of M is defined by:

$$q_i^+ = \lambda q_i + P u_i$$

and output is defined by:

$$y_i^+ = T q_i + Q u_i$$

where λ , P, T and Q are matrices of the appropriate size over the finite field (here GF(2)), and λ is called the transition matrix.

If the finite machine has no external input u_i, that is, the second term is omitted from the above next state and output equations, it is called an Autonomous Linear Finite State Machine. So the next state q_i^+ of M is defined by:

$$q_i^+ = \lambda q_i$$

and output is defined by:

$$y_i^+ = T q_i$$

In this thesis, we only consider autonomous LFSM with the underlying field GF(2).

Definition 2.1.2 Any LFSM is uniquely represented by a transition matrix λ and every transition matrix has a characteristic polynomial. The characteristic polynomial Δ of a LFSM is defined by:

$$\Delta = |x\mathrm{I} + \lambda|$$

where I is an identity matrix, x is an indeterminate, and $xI + \lambda$ is called the characteristic matrix of the LFSM.

Definition 2.1.3 If the sequence generated by an n-cell LFSM has period $2^n - 1$, then it is called a maximum-length sequence.

Definition 2.1.4 [3] The characteristic polynomial associated with an n-cell LFSM, which has period $2^n - 1$, is called a primitive polynomial.

If the characteristic polynomial of an $n \times n$ state matrix of an autonomous LFSM is primitive (which is called a primitive LFSM), the machine cycles through all $2^n - 1$ non-zero states.

The maximum-length sequence produced by a LFSM that has a primitive characteristic polynomial is indeed the property that one wants to exploit. The most common LFSMs

used in VLSI testing are LFSR and LHCA.

2.2 Linear Feedback Shift Registers

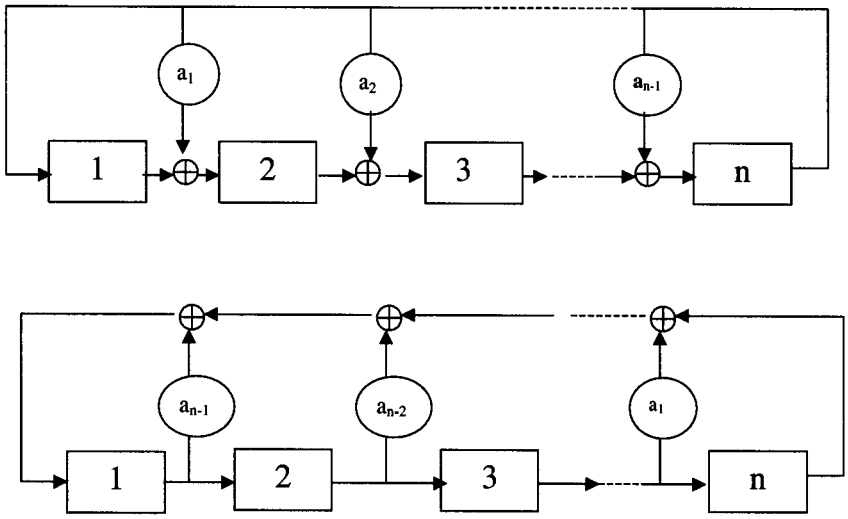

Figure 2.1. *Linear Feedback Shift Register;(top) internal-XOR ALFSR; (bottom) external-XOR ALFSR*

The most commonly used class of pseudorandom pattern generators in BIST are Autonomous Linear Feedback Shift Registers (ALFSRs). An ALFSR consists of a series of delay elements (D flip flops) with no external inputs and with all feedback functions provided by means of XOR gates [16] as illustrated in Figure 2.1.

Figure 2.1 shows the two categories of ALFSRs, which are called external-XOR ALFSR and internal-XOR ALFSR. Let $s = (s_1, s_2, ..., s_n)$ be the present state and $s^+ = (s_1^+, s_2^+, ..., s_n^+)$ be the next state. For internal-XOR ALFSR, we find:

$s_1^+ = s_n$

$s_i^+ = s_{i-1} + a_{i-1}s_n \qquad$ for $i = 2, 3, ..., n.$

According to Definition 2.1.2, the next state s^+ can be evaluated by $s^+ = \lambda s$. Thus, the

behavior of the n-cell internal-XOR ALFSRs is described by the $n \times n$ transition matrix λ:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & a_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & a_{n-2} \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 1 & a_{n-1} \end{bmatrix}$$

For the external-XOR ALFSR, we also find:

$s_i^+ = s_{i-1}$ for $i = 2, 3, ..., n$.

$s_1^+ = a_{n-1}s_1 + a_{n-2}s_2 + ... + a_{n-i}s_i + ... + a_1 s_{n-1} + s_n$

and the behavior of n-cell external-XOR ALFSRs is described by an $n \times n$ transition matrix λ:

$$\begin{bmatrix} a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} & a_{n-5} & \cdots & \cdots & a_3 & a_2 & a_1 & 1 \\ 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 1 & 0 \end{bmatrix}$$

Both of the transition matrices lead to a degree n characteristic polynomial $\triangle$:

$\triangle = 1 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_{n-1} x^{n-1} + x^n$

According to Definition 2.1.3 and Definition 2.1.4, if the polynomial $\triangle$ is primitive, then the ALFSR represented by the polynomial is a maximum-length ALFSR.

In VLSI testing, the Linear Feedback Shift Register (LFSR) is widely applied as a data compactor [3] and a pseudorandom pattern generator.

2.3 Cellular Automata

In recent VLSI testing developments, it has been proposed that test pattern generators based on cellular automata (CA) may be superior to those based on LFSR [21]. CA were proposed first by John von Neumann in the late 1940's [1], and were used for self-reproducing organism models. In 1983 and 1984, Stephen Wolfram published his papers[19, 20] , which are considered to be milestones for studying cellular automata in the computer science field.

CA are a realization of a finite state machine. A cellular automata consists of a regular uniform array, with a discrete variable at each cell [19]. They can be characterized by four features: the states of the cell, geometry, the neighborhood of a cell, and the transition rule.

2.3.1 Cell States

Assume that the cells of a cellular automata are in one of a finite number of possible states at any point of time. When these cells can have different state sets, the CA are called a polygeneous CA. However, the characteristics of CA are very complex, and in VLSI testing, CA are considered over the field GF(2), that is, the state space has only two elements, 0 and 1.

2.3.2 Geometry

An array of CA can be 1 dimensional, 2 dimensional or more than 2 dimensional. The greater the number of dimensions, the more complex are the geometries of CA. However, the geometry of CA depends not only on the dimension but also on the boundary conditions. In the finite array, different boundary conditions can be defined. So far we only consider the quiescent boundary condition, in which the extreme cells are considered to be adjacent to cells in some pre-specified state whose value does not change during the computation. For the linear CA, the quiescent boundary condition is the null boundary condition in which value of pre-specified state is zero. In this thesis, all of the CA considered are null boundary CA.

2.3.3 Neighborhood

According to the geometry of multiple-dimensional cellular automata, there are complex neighborhoods that can be generally defined as two kinds: local and global neighborhoods. Usually the neighborhoods are defined by the relation between inputs and outputs, that is, a cell takes its input from its input neighborhood and its state is available to the cells of its output neighborhood. Local neighborhood refers to the relation where a cell is solely influenced by its nearest neighbors, for example, the von Neumann (orthogonal) neighborhood and the Moore (unit cube) neighborhood (see Figure 2.2). Wolfram[19] proposed a local neighborhood for 1 dimensional neighborhood-three CA that is depicted in Figure 2.2. Global neighborhoods address the relation where a cell is influenced by not only its nearest neighbors but also more distant neighbors.

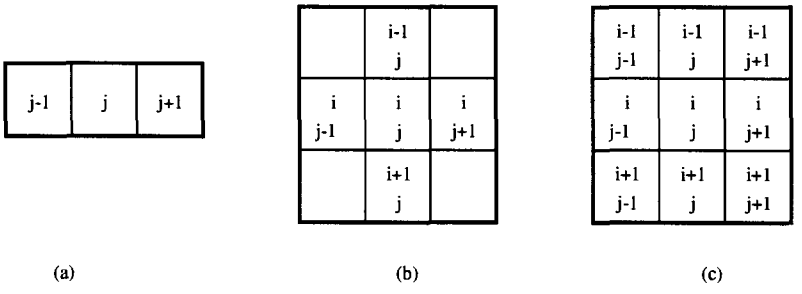

Figure 2.2. *Examples of the most frequently used neighborhoods in 1-dimensional and 2-dimensional CA:* (*a*) *1-dimensional von Neumann neighborhood or Wolfram neighborhood* (*b*) *2-dimensional von Neumann neighborhood* (*c*) *the Moore neighborhood*

2.3.4 Rule

The rules of a CA refer to the algorithms used to compute the successor states. Usually a rule can be expressed as a function by which the next state of a cell depends on the present states of k neighborhood cells and possibly its own present state. For different

neighborhoods or geometries, there exist many complex and different rules. In this thesis, only 1-dimensional CA are considered.

For 1-dimensional CA, let s_i be the current state of the cell i; the next state s_i^+ of the cell i can be represented as a function of the present state of cells i, $i-1$, $i-2$, ... , $i-r$ (left neighbors), and of cells $i+1$, $i+2$, ... , $i+k$ (right neighbors);

$$s_i^+ = f\,(s_{i-r}, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{i+k})$$

$$i,\ r,\ k \text{ are integers.}$$

where f is known as a rule. Thus, a rule is a function which is used to describe how the next state of a cell changes in response to its current state and those of its neighbors.

In the 1-dimensional CA, Wolfram proposes to use the local, distance 3 neighborhood for the cell rules. According to Wolfram's theory, every cell of a 1-dimensional CA has a relationship with only two nearest neighbors; namely the cell to the left and the cell to the right. There are 2^{2^3} possible rules when the next state of a cell is dependent on the current states of its two neighbors' or dependent on the current states of its two neighbors' and its own. In this case, rule of 1-dimensional CA can be defined as:

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

In order to define Wolfram's rules clearly, we can use a transition table, similar to a truth table. In such a table, the first line lists the eight possible states of the cell and its two adjacent cells as indicated by 3-bit binary numbers. In the second row, a rule can be described by an eight-digit binary number $(r_7 \ldots r_0)$. The r_i is 1 iff $f(a,b,c) = 1$ for abc as the logic states shown in the first line of Table 2.1 and is 0 otherwise. The last row shows the decimal numbers associated with the corresponding binary bit in the second line. The rule table for two examples is shown in Table 2.2.

From the example, rule 90 can be expressed by the 8-bit binary number 01011010, because the binary number " 01011010 " is represented by the decimal number " 90 ". The same condition is for rule 150.

Logic States	111	110	101	100	011	010	001	000
Binary Rule	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0
Decimal Rule	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table 2.1. *Wolfram's Rule Table*

	111	110	101	100	011	010	001	000
Rule 90	0	1	0	1	1	0	1	0
90 =	0	2^6	0	2^4	2^3	0	2^1	0
Rule 150	1	0	0	1	0	1	1	0
150 =	2^7	0	0	2^4	0	2^2	2^1	0

Table 2.2. *Wolfram's Rule Table of rule 90 and 150*

2.3.5 Linear Hybrid Cellular Automata

Though there are 256 rules in 1-dimensional neighborhood-three CA, there exist 8 linear rules. In this thesis, we only consider the two linear rules: rule 90 and rule 150. The class of 1-dimensional CA determined by the two rules are called Linear Hybrid Cellular Automata (LHCA) and their structure is shown in Figure 2.3

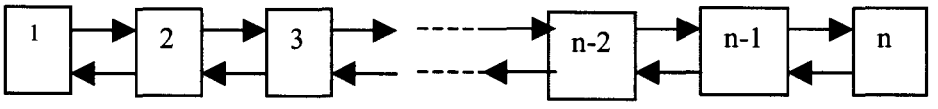

Figure 2.3. *1-Dimensional LHCA*

Rule 90 and rule 150 can be formally written by the following expression:

$$s_i^+ = s_{i-1} + d_i s_i + s_{i+1}$$

where $d_i = 1$ implies rule 150, and $d_i = 0$ implies rule 90 and " + " is over GF(2).

The current state of an n-cell LHCA is represented by the vector $s = [\, s_1, \, s_2, \ldots, \, s_n]$. The next state of the LHCA is represented by the vector $s^+ = [\, s_1^+, \, s_2^+, \ldots, \, s_n^+]$. Since the

next state function is a linear operator, the rule function can be expressed by the following $n \times n$ transition matrix λ:

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

and the next state can be obtained by:

$$s^+ = \lambda s$$

Given an LHCA with n cells, let Δ_k denote the characteristic polynomial of the LHCA formed by removing cells $k+1$, ..., $n-1$, n, thus, the characteristic polynomial of the original LHCA is Δ_n. As a result, the CA recursive relation[17] can be stated:

$\Delta_{-1} = 0$

$\Delta_0 = 1$

$\Delta_k = (x + d_k)\Delta_{k-1} + \Delta_{k-2}\ (1 \leq k \leq n)$

This recursive relation provides an efficient algorithm to compute the characteristic polynomial of a CA. In [6], an algorithm is presented to obtain a CA that has a given characteristic polynomial by using this recursive relation.

2.4 Built-In Self-Test

As a solution to increasingly complex digital circuits, BIST is being adopted as a preferred test strategy. BIST is a design technique in which parts of a circuit are used to test the circuit itself[3].

The principle of BIST is shown in Figure 2.4. BIST employs many techniques used in

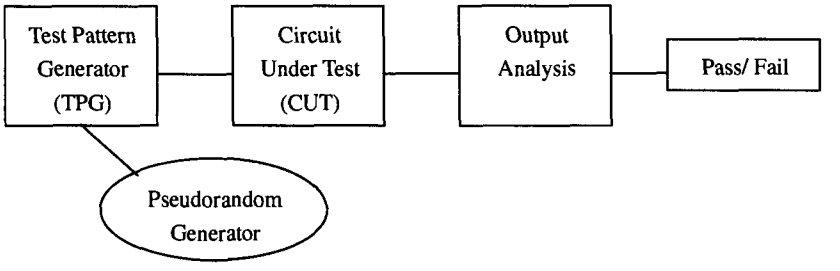

Figure 2.4. *BIST architecture*

integrating the test resources on the chip including a test pattern generator and a signature analyzer. The test pattern generator provides a test input sequence to the circuit under test (CUT). The output analysis compares the output sequence with the expected sequence and defines a " fail/pass " test output.

Among the different BIST approaches, the pseudorandom test is widely favored due to its associated low physical integrated circuits area overhead in manufacture. The pseudo-random sequence applied to the circuit may be generated by an LFSR or an LHCA as pseudorandom pattern generators.

Today, BIST techniques for combinational circuits are well established, whereas BIST techniques for sequential circuits are not yet mature. The main difficulty in implementing sequential BIST is that some internal faults in sequential circuits are highly resistant to pseudorandom patterns.[4] A preferred solution to overcome this problem is to use the new pseudorandom pattern generators or to modify the pseudorandom pattern generator.

This thesis focuses solely on a new pseudorandom pattern generator NCCA, which may lead to a higher degree of randomness and a more efficient pseudorandom pattern generator in BIST than a LFSR and a LHCA. In the following chapters, the NCCA are tested by Knuth randomness tests and evaluated by conducting fault simulation experiments using the ISCAS'85 benchmarks and ISCAS'89 benchmarks circuits, and the results are compared with those obtained using LFSR and LHCA as pseudorandom pattern generators.

2.5 Summary

This chapter provides background materials on the following topics: the definition of Linear Finite State Machines (LFSMs); two of the most important and special forms of LFSMs, namely Linear Feedback Shift Registers (LFSR) and Linear Hybrid Cellular Automata (LHCA). In the last section, Built-in Self-test (BIST) is discussed together with the application of these LFSMs.

Chapter 3

The New Class of Cellular Automata

In this chapter, a new class of cellular automata (NCCA) is introduced and analyzed. In Section 3.1, we define NCCA with its notation and introduce the two simple rules. Section 3.2 defines the NCCA's transition matrix, characteristic matrix and characteristic polynomial. In Section 3.3, the general recursive relation, shown to be an important factor to study NCCA, is presented and proven. Section 3.4 lists the low-cost characteristic primitive polynomials for 1-cell NCCA through 100-cell NCCA. In Section 3.5, the transition property of NCCA is explored and compared with LHCA and LFSR to provide a theoretical basis for the experiments discussed in Chapter 5. Section 3.6 summarizes the materials concerning NCCA in VLSI testing applications.

3.1 The New Class of Cellular Automata (NCCA)

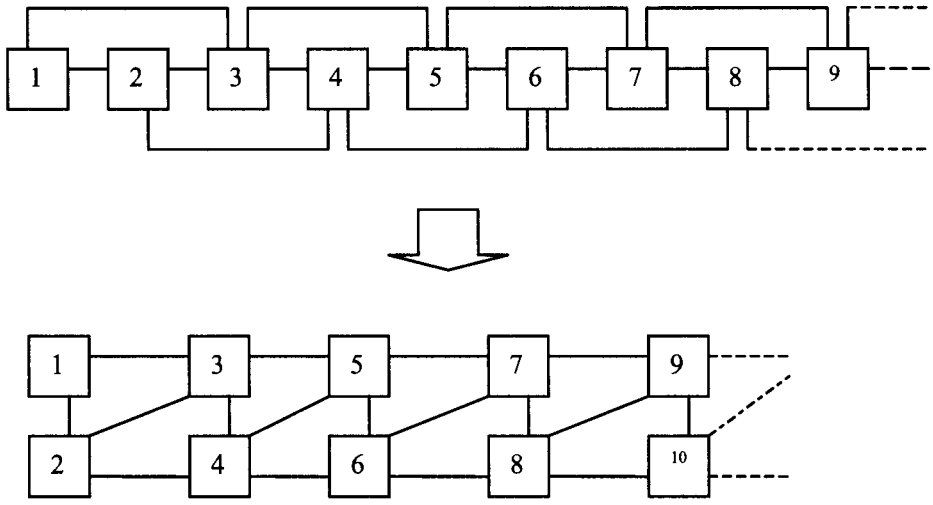

Figure 3.1. *The New Class of Cellular Automata*

NCCA are, in fact, 1-dimensional 5-neighborhood CA. Actually, NCCA (Figure 3.1) are an extension of the 1 dimensional CA presented by Wolfram which are 3-neighborhood

CA. Alternately, they can be considered as a simple 2-dimensional CA. For the 3-neighborhood CA, it is well known that there are 256 rules. For the NCCA, however, there exists a much larger number of more complex rules (2^{2^5}). This occurs because the next state of any cell is dependent on the states of the four closest neighbors and/or its own in the current states.

Only two simple linear rules are considered because these rules can be considered as the direct generalization of those used in LHCA, a kind of CA with local neighborhoods, a symmetric and regular structure, and more importantly, they also lead to the primitive LFSM.

Rule 0: $s_i^+ = s_{i-2} + s_{i-1} + s_{i+1} + s_{i+2}$

Rule 1: $s_i^+ = s_{i-2} + s_{i-1} + s_i + s_{i+1} + s_{i+2}$

Rule 0 implies that the next state of a cell depends on all of its four closest neighbors' current states; rule 1 implies that the next state of a cell depends on all of its four closest neighbors' and its own current state.

The NCCA considered are all null boundary, so the boundary conditions of the NCCA are defined:

$$s_0 = 0 \qquad s_{-1} = 0$$
$$s_{n+1} = 0 \qquad s_{n+2} = 0$$

3.2 Transition Matrix

In GF(2), rule 1 and rule 0 can be defined by the expression:

$$s_{i+1}^+ = s_{i-2} + s_{i-1} + d_i s_i + s_{i-1} + s_{i+2}$$

where $d_i = 1$ implies rule 1, $d_i = 0$ implies rule 0, and "+" is over GF(2). The general form of an n-cell NCCA transition matrix λ is expressed as:

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

The characteristic matrix A of a NCCA is defined by:

$$A = Ix + \lambda \tag{3.1}$$

thus

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

where I is an identity matrix and x is a variable.

The characteristic polynomial Δ of a NCCA is defined by:

$$\Delta = det(A) \tag{3.2}$$

where det(A) computes the determinant of the matrix A.

3.3 Recursive Relation

The recursive relation of NCCA is to provide an efficient algorithm to compute characteristic polynomials of NCCA.

Definition 3.3.1 For an n-cell NCCA, Δ_{ik} is defined to be the characteristic polynomial of the partial CA consisting of cells i through k. If $i = 1$, one can define that Δ_k equals Δ_{1k}, thus, the characteristic polynomial of the original NCCA is Δ_n.

Theorem 3.3.1 Given an NCCA with n cells:

$$\begin{aligned} \Delta_{-3} &= \Delta_{-2} = \Delta_{-1} = 0 \\ \Delta_0 &= 1 \\ \Delta_k &= (x + d_k) * \Delta_{k-1} + \Delta_{k-2} + (x + d_{k-1}) * \Delta_{k-3} + \Delta_{k-4} \quad (1 \leq k \leq n) \end{aligned} \tag{3.3}$$

where d_i is the rule of the i-th cell, $1 \leq i \leq k$ or $[d_1 d_2 ... d_k]$ is the rule vector of the k-cell NCCA.

Proof.

To prove the general case, assume that A is the characteristic matrix of a k-cell NCCA with the rule vector $[d_1 d_2 ... d_k]$;

$$A = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & x+d_{k-2} & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 & x+d_{k-1} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 1 & x+d_k \end{bmatrix}$$

Simplifying det(A) by expansion along the last row,

$det(A) = (x + d_k) * det(B) + det(C) + det(D)$

where:

$$B = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & x+d_{k-2} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 & x+d_{k-1} \end{bmatrix},$$

and

$$C = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & x+d_{k-2} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

and

$$D = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 0 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & x+d_{k-1} & 1 \end{bmatrix}.$$

Now, B is the characteristic matrix of the NCCA by removing the last cell, and so

$det(B) = \Delta_{k-1}$.

The det(C) and det(D) are expanded along the last column. So

$det(C) = det(C_1) + det(C_2)$

$det(D) = det(D_1) + det(D_2)$

where:

$$C_1 = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 1 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & x+d_{k-2} \end{bmatrix},$$

and

$$C_2 = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 1 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix},$$

and

$$D_1 = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & x+d_{k-1} \end{bmatrix},$$

and

$$D_2 = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Now, the matrix C_1 is the characteristic matrix of the NCCA by removing the last two cells, and so $det(C_1) = \Delta_{k-2}$.

The det(C_2), det(D_1) and det(D_2) are expanded again through the last column, or the last row,

$det(C_2) = det(C_{21}) + det(C_{22})$

$det(D_1) = det(D_{11}) * (x + d_{k-1}) + det(D_{12})$

$det(D_2) = det(D_{21}) + det(D_{22})$

where

$$C_{21} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} \end{bmatrix},$$

and

$$C_{22} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 0 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 1 & 1 & 1 \end{bmatrix},$$

and

$$D_{11} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} \end{bmatrix},$$

and

$$D_{12} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix},$$

and

$$D_{21} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 1 & 1 & x+d_{k-3} \end{bmatrix}.$$

and

$$D_{22} = \begin{bmatrix} x+d_1 & 1 & \cdots & 0 & 0 & 0 \\ 1 & x+d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & x+d_{k-5} & 1 & 1 \\ 0 & 0 & \cdots & 1 & x+d_{k-4} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}.$$

from those, one can derive the following:

$det(C_{21}) = det(D_{11}) = det(D_{21}) = \Delta_{k-3}$

$det(D_{12}) = \Delta_{k-4}$.

It is obvious that C_{22} is the transpose of the matrix D_{22},

so $det(C_{22}) = det(D_{22})$.

All operations are over GF(2), which leads to $det(C_{22}) + det(D_{22}) = 0$.

Thus

$det(A) = (x + d_k) * det(B) + det(C) + det(D)$

$= (x + d_k) * \Delta_{k-1} + det(C_1) + detC_2 + det(D_1) + det(D_2)$

$= (x + d_k) * \Delta_{k-1} + \Delta_{k-2} + det(C_{21}) + det(C_{22}) + det(D_{11}) * (x + d_{k-1}) + det(D_{12}) +$

$det(D_{21}) + det(D_{22})$

$= (x + d_k) * \Delta_{k-1} + \Delta_{k-2} + (x + d_{k-1}) * \Delta_{k-3} + \Delta_{k-4}$.

■

Example 3.3: For a 4-cell NCCA with a rule vector [1100].

Using(3.3), we get:

$$\begin{aligned}
\Delta_0 &= 1 \\
\Delta_1 &= x + 1 \\
\Delta_2 &= (x + d_2) * \Delta_1 + \Delta_0 = (x + 1) * (x + 1) + 1 = x^2 \\
\Delta_3 &= (x + d_3) * \Delta_2 + \Delta_1 + (x + d_2) * \Delta_0 = x * x^2 + x + 1 + x + 1 = x^3 \\
\Delta_4 &= (x + 0) * \Delta_3 + \Delta_2 + (x + 0) * \Delta_1 + \Delta_0 \\
&= x * x^3 + x^2 + x * (x + 1) + 1 = x^4 + x + 1
\end{aligned}$$

According to (3.2), the following characteristic polynomial is also obtained:

$$\Delta_4 = \begin{vmatrix} x+1 & 1 & 1 & 0 \\ 1 & x+1 & 1 & 1 \\ 1 & 1 & x & 1 \\ 0 & 1 & 1 & x \end{vmatrix} = x^4 + x + 1$$

Obviously, the computation results for the characteristic polynomial are the same by the two different methods. Note that when the number of NCCA's cells increases, the recursive algorithm using (3.3) is simpler and more effective than that using (3.2).

3.4 Minimum-Cost Primitive NCCA

In VLSI testing, the primitive NCCA are desirable as pseudorandom pattern generators because they can generate the maximum-length sequence. For all practical purposes, the minimum-cost hardware of primitive NCCA (called the minimum-cost primitive NCCA) should be sought and employed. Specifically, the minimum-cost primitive NCCA have the minimal number of rule 1 among all of the primitive NCCA because the structure of rule 1 is a little more complex than that of rule 0 in evaluation and implementation.
The procedure for finding the minimal-cost primitive NCCA is described in Algorithm 3.4

Algorithm 3.4

define an n $\times$ n NCCA transition matrix λ as in Section 3.2;
initialization (let rule vector $d = [0\ 0\ ...\ 0]$);
repeat:
    j=1;
    select j rule-1 cells by a sequence of all combinations of j that can be taken from n;
    compute the characteristic polynomial Δ defined in (3.2);
    if Δ is primitive **then** return rule vector $[d_1\ d_2\ ...\ d_n]$ and **break**;
    j=j+1;
end repeat

We used Maple 7, which is specially designed for fundamental mathematics computation, and obtained the results shown in Table 3.1 and Table 3.2 which list the minimum-cost

primitive NCCA for degrees 1 through 100.

Degree n	Positions of Rule 1 Cells	Degree n	Positions of Rule 1 Cells
1	1	51	24
2	1	52	1,20
3	-	53	2,48
4	1,2	54	43, 54
5	2	55	5,55
6	2	56	32
7	1,2	57	3,47
8	1,2,6,8	58	1,3
9	3	59	3,36
10	1,2,4	60	18,60
11	3	61	3, 61
12	2,3	62	1 ,33
13	1,8	63	1, 23
14	2	64	7, 28
15	1,11	65	1, 10
16	1,9	66	1, 54
17	1,3	67	4, 12
18	2,8	68	1, 12
19	7	69	1, 42
20	2	70	4, 37
21	1,11	71	3
22	1,3,16	72	11
23	2,8	73	7, 8

Table 3.1. *Minimum-Cost Primitive NCCA, for Degrees 1 through 100*

Degree n	Positions of Rule 1 Cells	Degree n	Positions of Rule 1 Cells
24	8	74	8
25	4	75	9
26	1,6	76	1, 26
27	1,10	77	1, 20
28	7,16	78	3, 56
29	1,27	79	1, 62
30	2	80	4, 71
31	1,9	81	2, 22
32	3,22	82	1, 44
33	1,23	83	1, 3
34	13,16	84	13
35	9	85	1, 35
36	3,31	86	17
37	7	87	1, 18
38	6,8	88	7, 38
39	1,17	89	28
40	7,16	90	4, 14
41	9,29	91	2, 52
42	4,41	92	16
43	7,21	93	1, 53
44	13	94	4, 43
45	1,28	95	46
46	3,19	96	3, 7
47	18,46	97	1, 12
48	11,48	98	1, 48
49	23,46	99	3, 90
50	10	100	8, 88

Table 3.2. *Minimum-Cost Primitive NCCA, for Degrees 1 through 100 (Continued)*

The first and third columns are the degree n (or n-cell NCCA); the second and fourth columns are the positions of rule 1 of the NCCA vector. For example, for degree 8, we have 1, 2, 6, 8 which imply the 8-cell NCCA with the rule vector [1 1 0 0 0 1 0 1] or the 8-cell NCCA with rule 1 in cells 1, 2, 6 and 8, and rule 0 elsewhere. Note that for 3-cells, there does not exist any primitive NCCA since it is a cyclic LHCA. Note that the algorithm 3.4 is an exhaustive search technique and due to time, we only obtain all of the minimum-cost primitive NCCA for degree 1 through 100.

3.5 The Transition Properties

A good pseudorandom pattern generator in BIST must be able to test the faults not only in combinational circuits but also in sequential circuits. In combinational circuits, in general, the greater the number of the different patterns, the higher is the fault coverage. But, in sequential circuits, because a fault requires a pair of patterns to be tested, fault coverage is dependent on the number of different patterns as well as the ordering of these patterns, that is, the capability of testing sequential faults depends on the number of distinct transitions. Thus the number of transitions is used as a criterion to evaluate whether the generator is likely to have a good fault coverage for sequential faults.

Definition 3.5.1 For a given n-bit vector $(s_1, s_2, ..., s_n)$, $s_i = \{0, 1\}$, $1 \leq i \leq n$, a k-bit subvector of the vector is defined by

$$subvector_{[p,\ k]} = (s_p, s_{p+1}, ..., s_{p+k-1}) \tag{3.4}$$

where $1 \leq p \leq n$ and $1 \leq k \leq n + 1 - p$.

Definition 3.5.2 For a given sequence $< S >= S_1, S_2, ..., S_m$, the composition of $subvector_{[p,\ k]}$ of S_j and $subvector_{[p,\ k]}$ of S_{j+1}, where $1 \leq j < m$, is defined as a transition.

Theorem 3.5.1[22] Consider any n-cell LFSM test vector generator with the maximum cycle length $(2^n - 1)$. Let $F_{(LFSM,\ p,\ k)}$ be the maximum number of distinct transitions of k-bit subvector$_{[p,k]}$, produced by the LFSM, where $1 \leq p \leq n$ and $1 \leq k \leq n + 1 - p$. In this case, we have

$$\text{upper bound: } F_{(LFSM,\ p,\ k)} \leq \begin{cases} 2^{2k}, & 1 \leq k < \lceil n/2 \rceil \\ 2^n - 1, & \lceil n/2 \rceil \leq k \leq n \end{cases}$$

$$\text{lower bound: } F_{(LFSM,\ p,\ k)} \geq \begin{cases} 2^k, & 1 \leq k < n \\ 2^n - 1, & k = n \end{cases}$$

The detailed proof is presented in [22].

Consider an n-cell primitive LFSM. Let $F_{[LFSM,\ p,\ k]}$ be the maximum number of distinct transitions of k-bit subvector$_{[p,\ k]}$, generated by the LFSM, where $1 \leq p \leq n$ and $1 \leq k \leq n + 1 - p$. For an LHCA, we have the following theorem.

Theorem 3.5.2 [21, 22] If the LFSM is a LHCA, then:

$$F_{[LHCA,\ p,\ k]} = \begin{cases} 2^n - 1, & (k \geq n - 1)\ or\ (p = 2\ and\ k = n - 2) \\ 2^{k+1}, & ((k < n - 1)\ and\ (p = 1\ or\ p + k - 1 = n)) \\ 2^{k+2}, & otherwise \end{cases}$$

The detailed proof is presented in [22]. However, the theorem is not completely correct. Because when $k = 1$ and $1 \leq p \leq n$, there only exist 4 distinct transitions: 0-0, 0-1,1-0, 1-1.

A similar Theorem, giving the transition properties of NCCA is presented below:

Theorem 3.5.3 If the LFSM is a NCCA, then:

$$F_{(NCCA,\,p,\,k)} = \begin{cases} 2^{k+1}, & k = 1 \text{ and } n > 2 \\ 2^{k+2}, & (k = 2) \text{ and } (n > 4), \text{ or} \\ & ((2 < k < n-2) \text{ and } (p = 1 \text{ or } p+k-1 = n)) \\ 2^{k+3}, & ((k = 3) \text{ and } (p \neq 1 \text{ and } p+k-1 \neq n) \text{ and } (n > 6)), \text{ or} \\ & ((3 < k < n-3) \text{ and } (p = 2 \text{ or } p+k = n)) \\ 2^n - 1, & (k \geq n-1), \text{ or} \\ & ((k = n-2) \text{ and } (n \geq 4)), \text{ or} \\ & ((k = n-3) \text{ and } (p = 2 \text{ or } p = 3) \text{ and } (n \geq 6)), \text{ or} \\ & ((k = n-4) \text{ and } (p = 3) \text{ and } (n \geq 8)) \\ 2^{k+4}, & \text{otherwise} \end{cases}$$

Proof.

Let $(s_p,\ s_{p+1},\ \ldots\ , s_{p+k-1})$ be a k-cell substate and $(s_p^+,\ s_{p+1}^+,\ \ldots\ , s_{p+k-1}^+)$ be the corresponding next substate.

1. For the case of $k = 1$ and $n > 2$, for any substate s_p, we can find at most 4 distinct transitions: 0-0, 0-1, 1-0, 1-1, so $F_{(NCCA,\,p,\,k)}$ is $2^{k+1}(= 4)$.
2. For the case of $k = 2$ and $n > 4$, for any substate $(s_p,\ s_{p+1})$, according to Theorem 3.5.1, we can find at most 16 distinct transitions: 00-00, 00-01, 00-10, 00-11, 01-00, 01-01, 01-10, 01-11, 10-00, 10-01, 10-10, 10-11, 11-00, 11-01, 11-10, 11-11, so $F_{(NCCA,\,p,\,k)}$ is $2^{k+2}(= 16)$.
3. For the case of $((2 < k < n-2)$ *and* $(p = 1))$, according to the NCCA's rules in Section 3.1, the next substate $(s_1^+,\ s_2^+,\ \ldots\ , s_k^+)$ is dependent on $s_1,\ s_2,\ \ldots\ , s_k, s_{k+1}$, and s_{k+2}. The new members s_{k+1}, and s_{k+2} can provide four possibilities to affect the next substate. so the total number of all distinct transitions for the 2^k different k-bit substates is at most 2^{k+2}. Hence, $F_{(NCCA,\,p,\,k)}$ is 2^{k+2}. For the case of $((2 < k < n-2)$ *and* $(p+k-1 = n))$, the proof is similar.

4. For the case of $((k = 3)$ and $(p \neq 1$ *or* $p + k - 1 \neq n)$ and $(n > 6))$, based on Theorem 3.5.1, the upper bound of all distinct transitions for the 2^3 different 3-bit substate is 2^{3+3}. Hence, $F_{(NCCA,\ p,\ k)}$ is 2^{k+3}.

5. For the case of $((3 < k < n-3)$ and $(p = 2))$, the next substate $(s_2, \ldots, s_k)$ is determined by $s_1, s_2, \ldots, s_k, s_{k+1}, s_{k+2}$. The three new elements added, s_k, s_{k+1}, s_{k+2}, can provide 2^{k+3} distinct transitions for k-bit substates, so $F_{(NCCA,\ p,\ k)}$ is 2^{k+3}. The proof for the case of $((3 < k < n-3)$ and $(p + k = n))$ is similar.

6. For the case of $k \geq n - 1$, according to Theorem 3.5.1, when $\lceil n/2 \rceil \leq k \leq n$, the upper bound of transitions is $2^n - 1$. So is the case with $((k = n - 2)$ and $(n \geq 4))$, the case with $((k = n - 3)$ and $(p = 2$ *or* $p = 3)$ and $(n \geq 6))$ and the case with $((k = n - 4)$ and $(p = 3)$ and $(n \geq 8))$.

7. Otherwise, according to the NCCA's rules in Section 3.1, the next substate $(s_p, s_{p+1}, \ldots, s_{p+k-1})$ is determined by $s_{p-2}, s_{p-1}, s_p, s_{p+1}, \ldots, s_{p+k-1}, s_{p+k}, s_{p+k+1}$. The four new elements $s_{p-2}, s_{p-1}, s_{p+k}$ and s_{p+k+1} can provide 2^4 possibilities to affect the next substate, so the total number of all distinct transitions for the 2^k different k-bit substates is at most 2^{k+4}. Hence, $F_{(NCCA,\ p,\ k)}$ is 2^{k+4}.

■

Based on Theorems (3.5.2) and (3.5.3), we can easily find that the maximum-length NCCA potentially have more transitions than the maximum-length LHCA. The experimental results, which use transition test [23] to count the number of all the transitions for the specific subvectors, are shown in Table 3.3.

Note: All the NCCA are the minimal-cost primitive NCCA listed in Table 3.2. All the LHCA are also the minimal-cost LHCA listed in [8]. In row 2 the first integer 'p' stands for the starting position of LHCA and NCCA and the second integer 'k' the length of subvectors of LHCA and NCCA; e.g., $2, 4$ implies $\text{subvector}_{[2,4]}$. Row 3 is the number of all transitions for k-bit subvectors of NCCA. Row 4 is the number of all transitions for k-bit subvectors of LHCA.

	the Number of Transitions of 5-Cell LFSMs														
p, k	1,1	2,1	3,1	4,1	5,1	1,2	2,2	3,2	4,2	1,3	2,3	3,3	1,4	2,4	1,5
NCCA	4	4	4	4	4	16	16	16	16	31	31	31	31	31	31
LHCA	4	4	4	4	4	8	16	16	8	16	31	16	31	31	31

(a)

	the Number of Transitions of 6-Cell LFSMs													
p, k	1,2	2,2	3,2	4,2	5,2	1,3	2,3	3,3	4,3	1,4	2,4	3,4	1,5	2,5
NCCA	16	16	16	16	16	32	63	63	32	63	63	63	63	63
LHCA	8	16	16	16	8	16	32	32	16	32	63	32	63	63

(b)

	the Number of Transitions of 7-Cell LFSMs											
p, k	1,3	2,3	3,3	4,3	5,3	1,4	2,4	3,4	4,4	1,5	2,5	3,5
NCCA	32	64	64	64	32	64	127	127	64	127	127	127
LHCA	16	32	32	32	16	32	64	64	32	64	127	64

(c)

	the Number of Transitions of 8-Cell LFSMs								
p, k	1,4	2,4	3,4	4,4	5,4	1,5	2,5	3,5	4,5
NCCA	64	128	255	128	64	128	255	255	128
LHCA	32	64	64	64	32	64	128	128	64

(d)

	the Number of Transitions of 9-Cell LFSMs										
p, k	1,4	2,4	3,4	4,4	5,4	6,4	1,5	2,5	3,5	4,5	5,5
NCCA	64	128	256	256	128	64	128	256	511	256	128
LHCA	32	64	64	64	64	32	64	128	128	128	64

(e)

Table 3.3. *Transitions of NCCA and LHCA*

From Table 3.3, we notice:

1. When ($k = 1$ and $n > 2$), the maximum number of transitions is 4 for NCCA and LHCA, e.g., in Table 3.3(a), all of 1-bit subvectors.
2. When ($k = 2$ and $n > 4$), the maximum number of transitions is 16 for NCCA , but the number of transitions for LHCA is 8 (when $p = 1$ or $p + k - 1 = n$), or 16 (when $p \neq 1$ or $p + k - 1 \neq n$, e.g., in Table 3.3(a), all of 2-bit subvectors, in Table 3.3(b), all of 2-bit subvectors.
3. When (($2 < k < n - 2$) and ($p = 1$ or $p + k - 1 = n$)), the maximum number of NCCA's transitions is 2^{k+2}, but the maximum number of LHCA's transitions is 2^{k+1}, e.g., in Table 3.3 (e), the subvector$_{[1,4]}$, the subvector$_{[6,4]}$, the subvector$_{[1,5]}$ and subvector$_{[5,5]}$.
4. When (($k = 3$) and ($p \neq 1$ and $p + k - 1 \neq n$) and ($n > 6$)), the maximum number of NCCA's transitions is 64 (2^{k+3}), but the maximum number of LHCA's transitions is 32 (2^{k+2}), e.g., in Table 3.3 (c), subvector$_{[2,3]}$, subvector$_{[3,3]}$, and subvector$_{[4,3]}$
5. When $k \geq n-1$, the maximum number for both machines' transitions is $2^n - 1$, e.g., in Table 3.3(a), the subvector$_{[1,5]}$; in Table 3.3(b), the subvector$_{[1,5]}$ and subvector$_{[2,5]}$.
6. When $k = n - 2$ and $n \geq 4$, the maximum number of both machines' transitions is $2^n - 1$, e.g., in Table 3.3(a), the subvector$_{[2,3]}$, in Table 3.3(b), subvector$_{[2,4]}$, Table 3.3(c), the subvector$_{[2,5]}$,etc.
7. When $k = n - 3$ and ($p = 2$ *or* $p = 3$) and $n \geq 6$, the maximum number of NCCA's transitions is $2^n - 1$, but the maximum number of LHCA's transitions is 2^{k+2}, e.g., in Table 3.3(c), the subvector$_{[2,4]}$ and subvector$_{[3,4]}$; in Table 3.3(d), the subvector$_{[2,5]}$ and subvector$_{[3,5]}$.
8. For $k = n - 4$ *and* $p = 3$ and $n \geq 8$,the maximum number of NCCA's transitions is $2^n - 1$, but the maximum number of LHCA's transitions is 2^{k+2}, e.g., in Table 3.3(d), subvector$_{[3,4]}$.
9. Otherwise, the maximum number of NCCA's transitions is 2^{k+4}, but the maximum

number of LHCA's transitions is 2^{k+2}, e.g., in Table 3.3(e), the subvector$_{[3,4]}$ and subvector$_{[4,4]}$.

In [21, 22], it was proven that the number of transitions for an LHCA is more than that of an LFSR on average. From Theorems 3.5.2 and 3.5.3, we can conclude that the number of transitions for an NCCA is more than that of an LHCA on average. Therefore, NCCA have the largest number of transitions amongst these three particular types of LFSMs on average. For testing sequential circuits, NCCA used as a pseudorandom pattern generator may have better fault coverage than LHCA and LFSR and the results of these experiments are shown in Chapter 5.

3.6 VLSI Testing Applications

Similar to LHCA and LFSR, NCCA are also used as pseudorandom pattern generators in BIST. In order to prove whether the sequence generated by NCCA has enough pseudo-randomness, NCCA should be compared with the corresponding LHCA and LFSR used commonly as pseudrandom generators. In Chapter 4, Knuth's random tests are used to compare the three types of LFSMs, and their experimental results are analyzed. In chapter 5, the fault coverage for the ISCAS'85 and ISCAS'89 benchmark circuits is evaluated.

3.7 Summary

This chapter analyzes the characteristics of the NCCA in detail. First, it proposes a new class of CA, and obtains its characteristic matrix as well as proves a recursive relation to derive the characteristic polynomial. The transition behavior of NCCA is discussed in detail, and it is shown to be better than that of the other machines.

Chapter 4

Knuth's Tests for Pseudo-random Sequences

In this chapter, Knuth's commonly used empirical tests [12, 13] are adopted to evaluate the pseudorandom behavior of the patterns generated from the maximum-length NCCA. These are compared with those generated from the maximum-length LHCA and LFSR. This chapter covers the following: first, random numbers, and pseudorandom pattern generators are defined; second, a key and basic chi-square test is introduced; third, the empirical tests are discussed, as suggested in Knuth [14]; finally the tests are employed to evaluate the patterns generated by NCCA, LFSR, and LHCA, and the statistical results for the empirical tests are presented and explained.

4.1 Definitions

A random event is an event that occurs at a given time by way of chance, that is, there is no specific pattern, purpose, or objective. The term random number implies that numbers are chosen at random. Random numbers are a valuable source of data for testing the correctness and effectiveness in scientific fields including VLSI testing, computer algorithms and simulation.

A truly random number generator is a non-deterministic process that produces a sequence of numbers with an outcome that cannot be determined before it actually happens. In practice, a pseudorandom number generator is usually proposed, which takes as its input a (short) sequence of numbers (seeds). This generator is not truly random because the process that produces the sequence is deterministic and repeatable.

However, pseudorandom number generators are used only when their randomness satisfies the requirements posed by their actual application. In order to assess the qualities of the pseudorandom number generators properly, the randomness tests proposed by Knuth [14] are employed as an acceptance criterion.

Knuth's empirical tests can be applied to a sequence:

$$< U_n >= U_0,\ U_1,\ U_2,\ \cdots$$

where each U_i is a real number between 0 and 1.[14] Some of these tests are designed

mainly for integer-valued sequences. In which case, the sequence $< U >$ is converted into an integer-valued sequence $< Y >$:

$$< Y_n >= Y_0,\ Y_1,\ Y_2,\ \cdots$$

where each

$$Y_i = \lfloor dU_i \rfloor, \quad \text{d is an integer value.}$$

This is a sequence of integers $< Y_n >$ that appears to be independently and uniformly distributed ranged from 0 to $d - 1$. The size of the integer d is chosen conveniently, and note that the value of d should be large enough so that the test is meaningful, but not so large that the test becomes impracticably difficult to carry out.[14]

For the pseudorandomness of sequences of binary patterns, in which each element is represented by ' 1 ' and ' 0 ', it seems more appropriate to test N successive n-bit binary sequences by adopting the following steps [12, 13]:

1. The n-bit binary-value sequences are converted into the corresponding decimal-valued sequences;
2. These decimal-valued sequences are further converted into the decimal-valued sequences whose elements range from 0 to d-1 by taking them modulo d where d is chosen conveniently and not more than $N/10$ [14];
3. Knuth's empirical tests are applied to the sequences;
   - numbers in the sequences are grouped and counted; these are called the observed results;
   - given a distribution for a test, the expected results can be computed;
   - the observed and expected results are compared to check their difference and the chi-square test is used for the comparison.
4. Repeat steps 2 to 4 for different values of the modulo d.

In this thesis, NCCA, LHCA and LFSR are used as generators. If sequences produced by one of these generators pass a certain number of empirical tests, the generators can be considered as good pseudorandom generators.

4.2 Chi-square Test

The chi-square test [14] is the most famous and common of all statistical tests and is a basic method underlying many other tests. It is performed to test whether the observed frequencies (values) differ significantly from the expected frequencies (values), and is used for the goodness-of-fit test.

Before introducing the chi-square test, we define the following parameters, which are also used in Section 4.3.

1. Let N be the length of a sequence;
2. Let m to be the number of independent observations made from the sequences when executed by the different tests, e.g., the outcome of one observation has absolutely no effect on the outcome of any of the others. For most tests, m is less than N, as explained later;
3. Assume k to be the number of categories and every observation falls into one of the k categories.
4. Let p_i be the probability of the category i where $1 \leq i \leq k$, which is computed in accordance with the different distribution for the different tests and discussed later.
5. Let Y_i be the total number of observations that fall into the category i, which is counted during the process of the empirical tests.

The calculation of this form of the chi-square test requires four steps:

1. Computing the expected value.

   For any category i, the expected value is equal to mp_i. According to a suggestion in [14], the chi-square test is valid only when m is large enough so that mp_i is five or more.

2. Applying the chi-square formula.

   For any category i, subtract mp_i from Y_i, square the result and divide by mp_i. Finally perform the calculation for every category and sum the results.

$$V = \frac{(Y_1 - mp_1)^2}{mp_1} + \frac{(Y_2 - mp_2)^2}{mp_2} + \cdots + \frac{(Y_k - mp_k)^2}{mp_k}$$

The equation is commonly written as:

$$V = \sum_{(i=1)}^{k} \frac{(Y_i - mp_i)^2}{mp_i} \tag{4.1}$$

3. Calculating the degrees of freedom ν.

   The chi-square value is not interpretable directly but must be compared with a table of the chi-square distribution, such as Table 4.1 and Table 4.2, which gives values of " the chi-square distribution with ν degrees of freedom " for various values of ν. The degrees of freedom ν is equal to $k - 1$ [14], where k is the number of categories. In this thesis, the values of ν listed in Table 4.1 and Table 4.2 are less than or equal to 100.

4. Using the chi-square table.

   A chi-square table, which in effect is built into statistical software packages, provides critical values. The rows and columns of the table are indexed by the degrees of freedom ν and the probability P. If the table entry in row ν under the column P is x, it means , " the quantity V in (4.1) will be less than or equal to x with approximate probability P, if m is large enough ". For example, in Table 4.1, the value of x for the row with $\nu = 8$ and $P = 5\%$ is equal to 2.733, this means that the computed statistic V is greater than 2.733 with the probability 5%.

In this thesis, according to a suggestion in Knuth [14], for all degrees of freedom ν,

1. A sequence is considered as a pseudorandom sequence if the value of its computed statistic V lies between the values listed in the columns $P = 5\%$ and $P = 95\%$;
2. A sequence is rejected as a pseudorandom sequence if the value of its computed statistic V is less than or equal to the values of column $P = 5\%$ or larger than or equal to the values of column $P = 95\%$.

Degrees of ν	P = 5%	P = 95%	Degrees of ν	P =5%	P = 95%
1	0.004	3.841	26	15.379	38.885
2	0.103	5.991	27	16.151	40.113
3	0.352	7.815	28	16.928	41.337
4	0.711	9.488	29	17.708	42.557
5	1.145	11.070	30	18.493	43.773
6	1.635	12.592	31	19.281	44.985
7	2.167	14.067	32	20.072	46.194
8	2.733	15.507	33	20.867	47.400
9	3.325	16.919	34	21.664	48.602
10	3.940	18.307	35	22.465	49.802
11	4.575	19.675	36	23.269	50.998
12	5.226	21.026	37	24.075	52.192
13	5.892	22.362	38	24.884	53.384
14	6.571	23.685	39	25.695	54.572
15	7.261	24.996	40	26.509	55.758
16	7.962	26.296	41	27.326	56.942
17	8.672	27.587	42	28.144	58.124
18	9.390	28.869	43	28.965	59.304
19	10.117	30.144	44	29.787	60.481
20	10.851	31.410	45	30.612	61.656
21	11.591	32.671	46	31.439	62.830
22	12.338	33.924	47	32.268	64.001
23	13.091	35.172	48	33.098	65.171
24	13.848	36.415	49	33.930	66.339
25	14.611	37.652	50	34.764	67.505

Table 4.1. *Selected Percentage Points of the Chi-square Distribution[2]*

Degrees of ν	P = 5%	P = 95%	Degrees of ν	P = 5%	P = 95%
51	35.600	68.669	76	56.920	97.351
52	36.437	69.832	77	57.786	98.484
53	37.276	70.993	78	58.654	99.617
54	38.116	72.153	79	59.522	100.749
55	38.958	73.311	80	60.391	101.879
56	39.801	74.468	81	61.261	103.010
57	40.646	75.624	82	62.132	104.139
58	41.492	76.778	83	63.004	105.267
59	42.339	77.931	84	63.876	106.395
60	43.188	79.082	85	64.749	107.522
61	44.038	80.232	86	65.623	108.648
62	44.889	81.381	87	66.498	109.773
63	45.741	82.529	88	67.373	110.898
64	46.595	83.675	89	68.249	112.022
65	47.450	84.821	90	69.126	113.145
66	48.305	85.965	91	70.003	114.268
67	49.162	87.108	92	70.882	115.390
68	50.020	88.250	93	71.760	116.511
69	50.879	89.391	94	72.640	117.632
70	51.739	90.531	95	73.520	118.752
71	52.600	91.670	96	74.401	119.871
72	53.462	92.808	97	75.282	120.990
73	54.325	93.945	98	76.164	122.108
74	55.189	95.081	99	77.046	123.225
75	56.054	96.217	100	77.929	124.342

Table 4.2. *Selected Percentage Points of the Chi-square Distribution (Continued)[2]*

4.3 Empirical Tests

In this section, we give a brief explanation of the equidistribution test, the serial test, the poker-t test, the gap test, the run test, and the permutation test of Knuth's empirical tests. Please refer to [13, 14] for a more detailed description. We perform the empirical tests on the successive integer sequences, each of length N, and for each test we assume m observations, k categories and the probability p_i of each category i, $1 \leq i \leq k$. The user manual for the empirical tests, coded in C, is given in appendix B.

4.3.1 Equidistribution Test

The equidistribution test is used to test if a sequence is a uniform distribution. We perform the test on an integer sequence ranging from 0 to $d-1$, and there are d categories: 0, 1, ..., $d-1$. Then we count the number of times that a member of the sequence falls into each category. The number should be approximately the same for each category if the sequence is uniform.

In this test, the number of observations is $m = N$.
The number of categories is $k = d$.
The expected probability for each category i is $p_i = \frac{1}{d}$.

4.3.2 Serial Test

The serial test is a higher dimensional version of the equidistribution test. Here successive pairs, tuples, quadruples, and so on are taken from the sequence and tested for uniform distribution. For the successive pairs, the test is called a serial-pair test or the serial-2 test, which is adopted in this thesis.

- serial pair: successive pairs of numbers (Y_{2j}, Y_{2j+1}) from the integer sequences ranging from 0 and $d-1$. There are d^2 categories: $(0,0)$, $(0,1)$, ..., $(0, d-$

1), $(1,0)$, ..., $(1,d-1)$, ..., $(d-1,d-1)$. We count the number of each category and check if its distribution is uniform.

In the serial-pair test, for an N integer number sequence,
the number of observations is $m = N/2$.
The number of categories is $k = d^2$.
The expected probability for each category i is $p_i = \frac{1}{d^2}$.

4.3.3 Poker-t Test

The poker-t test is also said to be a higher dimensional distribution. We generate N integers in [0, $d-1$], divide them into successive t-tuples and count the number r, where r is the number in the range from 1 to t, of distinct integers represented in each tuple. For example if $t = 3$, $d = 3$ and the sequence is: 0, 1, 1, 2, 2, 2, 0, 1, 2,..., then the number of distinct integers obtained in the first three 3-tuple are 2, 1, and 3. We compare the results with the expected distribution for random samples from the uniform distribution.

In this test, the number of observations is $m = \lfloor \frac{N}{t} \rfloor$.
The number of categories is $k = t$.
The expected probability for each category i is $p_i = \frac{d(d-1)...(d-i+1)}{d^t} \left\{ {t \atop i} \right\}$,
where $\left\{ {t \atop i} \right\}$ is the Stirling number of the second kind.

In this thesis, we apply the poker test for $t = 3$, that is, poker-3.

4.3.4 Gap Test

The gap test is commonly used on real number sequences in Knuth[14], but it can be used on integer number sequences. We generate N integer numbers in [0, $d-1$] and select two suitable integers a and b where $0 \leq a < b \leq d$. The gap is the length of the segments with no element in the given interval $[a,\ b)$, e.g., for the integer sequence: Y_j, $Y_{j+1}, Y_{j+2}, ..., Y_{j+r}$, if Y_{j+r} lies between a and b, but the other Y's do not, the gap is

r. We then count the number of different gaps and examine if the number is binomially distributed.

In this test, the number of observations m is dependent on the actual sequence data.

The number of categories is $k = G + 1$, where G is the maximal gap.

The expected probability for each category i is:

$p_i = p * (1-p)^i, \quad 0 \leq i < G$

$p_i = (1-p)^i, \quad i = G$

where p is given by $\frac{(b-a)}{d}$.

Example 4.3.4: Suppose $a = 3$, $b = 7$, and $d = 10$. Let a sample input sequence be 1, 2, 8, 3, 9, 1, 2, 7, 8, 4, 5, 6, 7. We note that ' 3 ' is the first occurrence of a number in $[3,\ 7)$ and thus forms the first gap; the next gap is formed by ' 5 ' and so on with a final result as:

$$\overbrace{1,2,8,3}^{3}\ \overbrace{9,1,2,7,8,4}^{5}\ \overbrace{5}^{0}\ \overbrace{6}^{0}\ \overbrace{7}^{1}$$

the number of categories k is 6, and the maximal Gap G is 5.

4.3.5 Run Test

The run test examines the monotonicity of the sequence as in increasing or decreasing relation. The run is defined as the length of an increasing (a decreasing) succession of numbers preceded and followed by a decreasing (an increasing) number. For example, we generate the sequence which is determined by the number of ways of assigning elements from $\{0, \ldots, d-1\} : Y_j, Y_{j+1}, Y_{j+2}, \ldots, Y_{j+r-1}, Y_{j+r}$ and, if $Y_j < Y_{j+1} < Y_{j+2} < \ldots < Y_{j+r-1}$, and $Y_{j+r-1} \geq Y_{j+r}$, the run is r. At the same time, we discard the number Y_{j+r} that follows a run before starting the new run, thus, the adjacent run is independent. We count the number of different runs and plot its distribution. In the thesis, we perform the increasing test.

In this test, the number of observation m can not be determined until the test is actually performed.

The number of categories is $k = R + 1$, where R is the maximal run.
The expected probability for each category i is $p_i = \frac{1}{i!} - \frac{1}{(i+1)!} \quad 1 \leq i < R$,

4.3.6 Permutation-t Test

The permutation-t test is used to check if each possible permutation occurs about equally often. We generate an N integer sequence in [0, $d-1$] and divide the sequence into the t-tuple sequence. Assume there are distinct numbers in each t-tuple, then we can rank each number in each tuple: the smallest number is ranked as 1, the second smallest number is ranked as 2, ..., the largest number is ranked t, so there are $t!$ possible permutations, for example, if $t = 3$, then the following orders are possible: (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1). We count the number of time each permutation occurs and investigate its distribution.

In this test, the number of observations is $m = \lfloor \frac{N}{t} \rfloor$.
The number of categories is $k = t!$.
The expected probability for each category i is $p_i = \frac{1}{t!}$.

As we notice, k and p_i described above are valid only if the elements in each tuple are distinct. However, in practice there could exist the same elements in the same tuples, e.g., for the permutation-3 test, there should be 13 possible permutation: (1 ,1 ,1), (1, 1, 2), (1, 2, 1), (1, 2, 2), (1, 2, 3), (1, 3, 2), (2, 1, 1), (2, 1, 2), (2, 1, 3), (2, 2, 1), (2, 3, 1), (3, 1, 2) and (3, 2, 1), so we modify k and p_i in order to handle the tuples which may have the same elements and get the following results:

In this test, assume that the test divides the sequence into successive t-tuples which have r distinct numbers ($0 < r \leq t$), then

The number of categories is $k = \sum_{r=1}^{t} \{ {t \atop r} \} r!$
The expected probability for each category i is $p_i = \frac{\binom{d}{r}}{d^t}$

Here we adopt permutation-3 test.

4.4 Knuth's Empirical Tests Results for LFSMs

All of the above empirical tests are applied to the sequences generated by the maximum-length LFSR , LHCA and NCCA. During the process of the tests, we set up:

1. The length of the sequences (N) is 10,000;
2. Several different modulo d values are chosen conveniently and adopted. Here all of primes that are larger than 2 and less than 75 are used in the the empirical tests except the serial-pair test. For the serial-pair test, the modulo d values are the primes no more than 10 because if d is more than 10, the number of categories is (d^2) which should cause very small value or no observation for some categories, which leads to invalid chi-square values;
3. For a particular test and modulo d, we compute the value of the test statistic V by using the chi-square formula 4.1, and compare it with the chi-square tables 4.1 and 4.2. If V falls between the values listed in the columns P = 5% and P = 95%, the sequence is considered as passing the test for a particular modulo; or else, the sequence fails the test for a particular modulo;
4. Every generator is initialized to three different starting points: 0000...0001, 1111...1111, and a random binary number which is the same for the same length machines. If more than 1 out of 3 sequences pass some particular test for some particular modulo d, the sequences produced by the generator can pass the particular test with the modulo d and is marked as " P "; or else, the sequences produced by the generator fail the test with the modulo d and are marked as " F ".
5. "weight" is the count of P's in a column. For any particular test, if the value of weight is more than 10, the sequence passes the test; otherwise, it fails the test.
6. If the sequences produced by a generator can pass at least 4 out of the 6 empirical tests, the generator, which produces the sequence, is considered as a good pseudo-random generator.

In this thesis, All of the generators we used are maximum-length LFSR, LHCA and NCCA and their polynomials are listed below:

- 16 bit LFSMs

  LFSR: $x^{16} + x^{12} + x^{11} + x^7 + x^4 + x + 1$

  LHCA: $x^{16} + x^{12} + x^{11} + x^7 + x^4 + x + 1$

  NCCA: $x^{16} + x^{12} + x^{11} + x^7 + x^4 + x + 1$

- 17 bit LFSMs

  LFSR: $x^{17} + x^{13} + x^{10} + x^8 + x^6 + x^5 + x^2 + x + 1$

  LHCA: $x^{17} + x^{13} + x^{10} + x^8 + x^6 + x^5 + x^2 + x + 1$

  NCCA: $x^{17} + x^{13} + x^{10} + x^8 + x^6 + x^5 + x^2 + x + 1$

- 18 bit LFSMs

  LFSR: $x^{18} + x^7 + 1$

  LHCA: $x^{18} + x^{15} + x^{12} + x^{10} + x^8 + x^2 + 1$

  NCCA: $x^{18} + x^{15} + x^{13} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$

- 19 bit LFSMs

  LFSR: $x^{19} + x^{14} + x^2 + x + 1$

  LHCA: $x^{19} + x^{18} + x^{12} + x^{11} + x^{10} + x^8 + x^3 + x^2 + 1$

  NCCA: $x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$

- 28 bit LFSMs

  LFSR: $x^{28} + x^9 + 1$

  LHCA: $x^{28} + x^{27} + x^{26} + x^{25} + x^{24} + x^{23} + x^{21} + x^{19} + x^{18} + x^{17} + x^{16} + x^7 + x^5 + x^3 + x^2 + x + 1$

  NCCA: $x^{28} + x^{24} + x^{22} + x^{19} + x^{18} + x^{17} + x^{15} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + 1$

- 31 bit LFSMs

LFSR: $x^{31} + x^6 + 1$

LHCA: $x^{31} + x^{30} + x^8 + x^4 + 1$

NCCA: $x^{31} + x^{26} + x^{25} + x^{22} + x^{21} + x^{15} + x^{13} + x^{12} + x^{11} + x^{10} + x^3 + x + 1$

- 35 bit LFSMs

LFSR: $x^{35} + x^2 + 1$

LHCA: $x^{35} + x^{34} + x^{32} + x^{28} + x^{27} + x^{26} + x^{24} + x^{19} + x^{18} + x^{16} + x^3 + x^2 + 1$

NCCA: $x^{35} + x^{34} + x^{33} + x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{27} + x^{26} + x^{23} + x^{21} +$
$x^{20} + x^{18} + x^{17} + x^{15} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$

Note the length of LFSMs is selected according to the ISCAS'89 benchmark circuits because the sequences generated by these generators are used as inputs on the benchmark circuits in chapter 5. In applications, the minimum-cost LFSMs are commonly used as pseudorandom generators, so we select the minimum-cost 18-bit, 19-bit, 28-bit, 31-bit and 35-bit LFSMs which are listed in Table 3.1, Table 3.2 and [8] as generators and employ Knuth's empirical tests to evaluate them. How about another case: the generators produced by different LFSMs which have the same polynomials? So here we adopt the same polynomials for 16-bit and 17-bit LFSMs in order to compare the three LFSMs more comprehensively.

Then we employ the Knuth Tests using the testbench [23] on the above LFSMs, and we list the experimental results in the following tables:

From the tables, we conclude:

- Except for the equidistribution test, all the LFSRs fail all of Knuth's empirical tests. Based on the tests, LFSRs can not be considered as good pseudorandom generators.
- Both of the two linear cellular automata pass more than 50 percent of the Knuth's empirical tests except the run up test, e.g., for the gap tests, the weights for both machines for each test and each length are more than 15, so they pass the gap test, and therefore appear to be good pseudorandom generators.
- If analyzed in more detail, NCCA is obviously a little better from a randomness

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N
MOD	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C
	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C
d	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
5	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	P	P	P	F	P	F	F	F	F	F	P	P	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	F	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
29	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
31	P	F	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
37	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	P	F	F	P	P	F	P	F	F	P	P
43	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
47	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
53	F	P	P	-	-	-	F	F	P	P	P	P	F	P	P	F	P	P
59	P	F	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
61	P	P	F	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
71	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
73	P	P	F	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
weight	19	18	18	0	3	2	0	7	9	3	20	20	0	20	19	0	19	20

Table 4.3. *Knuth Tests Results for 16-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N
MOD	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C
	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C
d	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
5	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	F	P	P	F	P	P	F	F	F	P	P	P	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	F
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
23	P	P	F	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
29	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
31	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
37	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	P	F	F	P	P	F	P	P	F	P	P
43	P	P	P	-	-	-	F	P	P	F	P	P	F	P	F	F	P	F
47	P	P	P	-	-	-	F	P	P	F	P	P	F	F	P	F	P	F
53	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
59	F	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
61	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
71	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
73	P	P	P	-	-	-	F	P	F	P	P	P	F	P	P	F	P	P
weight	16	20	19	0	3	3	0	8	8	5	20	20	0	19	19	0	20	17

Table 4.4. *Knuth Tests Result for 17-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N
MOD	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C
	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C
d	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	F	P	F	P	P
5	P	F	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	P	P	P	F	P	P	P	F	F	F	P	P	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	F	P	P	-	-	-	P	F	F	P	P	P	F	P	P	F	P	P
29	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
31	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
37	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
43	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
47	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
53	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
59	P	P	P	-	-	-	F	P	P	F	P	P	F	F	P	F	P	P
61	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
71	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
73	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
weight	18	19	20	0	3	3	2	9	10	6	20	20	0	18	20	0	20	20

Table 4.5. *Knuth Tests Results for 18-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N
MOD	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C
	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C
d	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
5	F	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	P	F	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	P	P	P	-	-	-	f	F	F	F	P	P	F	P	P	F	P	P
29	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
31	P	P	P	-	-	-	F	P	F	P	P	P	F	P	P	F	P	P
37	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	P	F	F	P	P	F	F	P	F	P	P
43	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
47	P	P	P	-	-	-	F	P	P	F	F	P	F	P	P	F	P	P
53	P	P	P	-	-	-	F	P	F	F	P	P	F	P	P	F	P	P
59	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
61	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
71	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
73	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
weight	18	20	19	0	3	3	0	10	7	2	19	20	0	19	20	0	20	20

Table 4.6. *Knuth Tests Results for 19-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
MOD	L F S d R	L H C R	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	P	F	F	P	P
5	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	P	P	P	-	-	-	f	F	F	F	P	P	F	P	P	F	P	F
29	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
31	P	F	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
37	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	F
43	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
47	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
53	P	P	P	-	-	-	F	P	F	F	P	P	F	P	P	F	F	P
59	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
61	P	P	P	-	-	-	F	P	P	F	P	P	F	P	F	F	P	P
67	P	P	P	-	-	-	F	P	F	F	F	P	F	P	P	F	P	P
71	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
73	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
weight	19	19	20	0	3	3	0	8	8	0	19	20	0	20	18	0	19	18

Table 4.7. *Knuth Tests Results for 28-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N	L	L	N
MOD	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C	F	H	C
	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C	S	C	C
d	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A	R	A	A
3	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
5	P	P	P	F	P	P	F	F	F	F	P	P	F	P	P	F	P	P
7	P	P	P	F	P	P	F	F	F	F	P	F	F	P	P	F	P	P
11	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
17	P	F	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	F	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
29	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
31	P	P	P	-	-	-	F	F	P	F	P	P	F	P	P	F	P	P
37	F	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
41	F	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
43	P	P	P	-	-	-	F	F	P	F	F	P	F	P	P	F	P	P
47	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
53	P	P	P	-	-	-	F	F	F	P	P	P	F	P	P	F	P	P
59	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
61	P	P	P	-	-	-	F	P	F	F	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
71	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
73	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
weight	16	19	20	0	3	3	0	6	7	2	19	19	0	20	20	0	20	20

Table 4.8. *Knuth Tests Results for 31-bit LFSM*

Test	Equidistri			Serial-2			Run up			Gap			Poker-3			Permut-3		
MOD d	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A	L F S R	L H C A	N C C A
3	F	F	P	F	P	P	F	F	F	F	F	P	F	F	P	F	F	P
5	P	P	P	F	F	P	F	F	F	F	P	P	F	P	P	F	F	P
7	F	P	P	F	F	P	F	F	F	F	F	F	F	P	P	F	P	P
11	F	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
13	P	P	P	-	-	-	F	F	F	F	P	P	F	F	P	F	F	P
17	F	F	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
19	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
23	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
29	P	P	F	-	-	-	F	F	F	F	P	P	F	F	P	F	P	P
31	P	P	P	-	-	-	F	F	F	F	P	P	F	P	P	F	P	P
37	P	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
41	P	P	P	-	-	-	F	P	F	F	P	F	F	P	P	F	P	P
43	P	F	P	-	-	-	F	P	P	F	F	P	F	P	P	F	F	P
47	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
53	F	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	F	P
59	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
61	F	P	P	-	-	-	F	P	P	P	P	P	F	P	P	F	P	P
67	P	P	P	-	-	-	F	P	P	F	P	P	F	P	P	F	P	P
71	P	P	P	-	-	-	F	P	P	F	P	F	F	P	P	F	P	P
73	F	P	P	-	-	-	F	P	P	P	F	P	F	P	P	F	P	P
weight	13	17	19	0	1	3	0	10	9	3	16	17	0	17	20	0	15	20

Table 4.9. *Knuth Tests Results for 35-bit LFSM*

perspective than an LHCA because over all of the Knuth tests, NCCA has the largest weight for each test among the three LFSMs , e.g., the totals of the weights from all of the experiments for the equidistribution test are 135, 132, 119 for NCCA, LHCA and LFSR respectively.

Based on the above criteria, we can conclude that the generators produced by NCCA and LHCA have much better pseudorandomness properties than those for LFSR. However, we restricted our work to only a small number of LFSMs because larger-sized binary sequence can cause overflow when the sequences are converted into the corresponding decimal-valued sequences. In order to obtain more general conclusions, many larger size LFSMs should be tested.

The test results are sensitive to several parameters that are normally chosen without much mathematical justification, such as a and b in the gap test. There are several other tests that are also applied for the testing of pseudorandomness properties of binary sequences, such as the spectral test [14], which is very complex and beyond the scope of this thesis. Further study for the LFSMs is in chapter 5.

4.5 Summary

This chapter employs Knuth's empirical tests suggested in [14] to evaluate the pseudorandomness of the sequences generated by the maximum-length LFSR, LHCA and NCCA. From the experimental results, the sequences generated by LFSR fail the pseudorandom tests whereas those generated by the two CA pass the tests; and the sequences generated by NCCA have a little better pseudorandomness than those generated by LHCA.

Chapter 5

Testing Applications

In this chapter, we briefly introduce benchmark circuits, then compare the performance of different BIST generators for the testing of the benchmark circuits using the stuck-at fault model. Fault simulation is used to evaluate the performance of these generators on the basis of the ISCAS'85 and ISCAS'89 benchmark circuits.

5.1 Benchmark Circuits

How to evaluate the pseudorandom generators in BIST? A set of benchmark circuits is used to compare the performance of different kinds of test generators and fault simulators. In this thesis we only consider the benchmark circuit sets ISCAS'85 and ISCAS'89, since they have been widely accepted and employed by many researchers. Using these simulation results such as fault coverage, designers have the choice of modifying or evaluating the design before it runs into the practical manufacture.

5.1.1 ISCAS'85 Benchmark Circuits

The ISCAS'85 benchmark set consists of ten combinational circuits provided at the International Conference "1985 International Symposium on Circuits and Systems". The detailed information can be found in [5].

Table 5.1 [5] lists the complete profile of the ten combinational circuits. The columns in the table correspond to circuit name, circuit function, the number of primary inputs and outputs, number of gates and the total number of faults (Collapsed Faults) in the ISCAS'85 benchmark circuits.

The Non-Redundant ISCAS '85 benchmark circuits, commonly called the ISCAS'85nr benchmark circuits, is a modified version of the ISCAS '85 benchmark circuits which removes the logic redundancies from the ISCAS'85 benchmark circuits. Therefore, ISCAS'85nr benchmark circuits are functionally equivalent to the original circuits. Here we employ the ISCAS'85nr benchmark circuits. Table 5.2 contains the profile of only those circuits which have been used in this thesis to compare the quality of NCCA as generators

Circuit Name	Circuit Function	Primary Inputs	Primary Outputs	Number of Gates	Collapsed Faults
c6288	16-bit Multiplier	32	32	2406	7744
c1908	16-bit ECAT	33	25	880	1879
c432	Priority Decoder	36	7	160	524
c499[1]	32-bit ECAT	41	32	202	758
c1355[1]	32-bit ECAT	41	32	546	1574
c3540	ALU and Control	50	22	1669	3428
c880	ALU and Control	60	26	383	942
c5315	ALU and Selector	178	123	2307	5350
c2670	ALU and Control	233	140	1193	2747
c7552	ALU and Control	207	108	3512	7550

Table 5.1. *ISCAS 85 Benchmark Circuits*

with LFSR and LHCA. Table 5.2 describes the known circuit functions [5] and the number of stuck-at faults of the ISCAS'85nr benchmark circuits.

Circuit Name	Circuit Function	No. of Stuck-at Faults
C6288nr	16-bit Multiplier	7710
C1908nr	16-bit ECAT	1879
C432nr	Priority Decoder	520
C499nr	32-bit ECAT	750
C1355nr	32-bit ECAT	1566
C3540nr	ALU and Control	3297

Table 5.2. *ISCAS 85nr Benchmark Circuits*

5.1.2 ISCAS'89 Benchmark Circuits

The ISCAS'89 benchmark circuits consist of thirty one sequential circuits also provided at International Conference "1989 International Symposium on Circuits and Systems". The detailed information can be found in [9, 10].

Owing to the limit of the minimum-cost primitive NCCA (degree = 3, no primitive), Table 5.3 contains the profile of only those sequential circuits which have been used in the thesis in order to compare the performance of three different LFSMs (NCCA, LFSR, and LHCA) as pseudorandom generators. The columns in Table 5.3 correspond to circuit name, the number of primary inputs, the number of primary outputs, the number of D-type flip flops, the number of gates and the number of collapsed faults.

5.2 Experimental Results

Though Chapter 4 shows that NCCA have a slightly higher randomness than LHCA and much higher than LFSR, this does not guarantee that NCCA will lead to the higher fault coverage than LFSR and LHCA, so we need to perform experiments to measure their fault coverage.

During the following experiments, the patterns produced by these generators are applied to the ISCAS benchmark circuits, because the benchmark circuits have been profoundly studied, widely accepted and commonly used. All of LFSMs are the minimal-cost primitive LFSMs. NCCA used are listed in Table 3.2. LHCA used are listed in [8]. LFSR used are listed in Appendix A. We perform the fault simulation on generators with initial vectors: 000...01, 111...10 and a random vector listed in Appendix C.

5.2.1 The Experimental Results for ISCAS'85

The experimental results are listed in Table 5.4 and they show that:

- all of the three LFSMs can obtain 100% fault coverage, though the number of patterns

Circuit Name	Primary Input	Primary Output	D-type Flip Flop	Number of Gates	Collapsed Faults
s27	4	1	3	10	32
s386	7	7	6	159	384
s1488	8	19	6	653	1486
s1494	8	19	6	647	1506
s344	9	11	15	160	342
s349	9	11	15	161	350
s208	11	2	8	96	215
s38584	12	278	1452	19253	36303
s1196	14	14	18	529	1242
s1238	14	14	18	508	1355
s15850	14	87	597	9772	11725
s953	16	23	29	395	1079
s1423	17	5	74	657	1515
s820	18	19	5	289	850
s832	18	19	5	287	870
s420	19	2	16	196	430
s510	19	7	6	211	564
s9234	19	22	228	5597	6927
s38417	28	106	1636	22179	31180
s13207	31	121	669	7951	9815
s641	35	24	19	379	467
s713	35	23	19	393	581
s838	35	2	32	390	857
s5378	35	49	179	2779	4603
s35932	35	320	1728	16065	39094

Table 5.3. *ISCAS 89 Benchmark Circuits*

The Minimum Number of Patterns to achieve 100% Fault Coverage							
Initial Value	LFSM Name	Circuits Name					
		c6288nr	c499nr	c1355nr	c1908nr	c432nr	c3540nr
000...01	LFSR	544	928	2528	14016	1472	11456
	LHCA	128	864	2880	12096	704	13504
	NCCA	160	992	1856	10944	1600	29056
111...10	LFSR	128	960	2144	9792	1216	76288
	LHCA	96	736	5376	18080	928	23136
	NCCA	224	992	2848	12224	576	67456
random number	LFSR	288	1344	3008	6368	896	30944
	LHCA	128	864	2432	11232	1408	30336
	NCCA	128	544	3744	10016	768	64480

Table 5.4. *Experimental Results for ISCAS'85*

used is different.

- looking at the number of patterns used, none of the LFSMs is superior to the others.

For the stuck-at faults in combinational circuits, the fault coverage is dependent on the number of distinct patterns. Because all of the three LFSMs have maximal length sequences, the final fault coverage is 100%, which is exactly what was expected.

5.2.2 The Experimental Results for ISCAS'89

The experimental results are listed in Tables 5.5.1 - 5.5.3. Each entry in the table gives the fault coverage for all faults in a circuit applying 2^n ($n \leq 17$, n is the number of the length of LFSMs) or 20,000 ($n > 17$) test vectors. During simulation, we respectively set all of flip-flops to unrestricted, logic 0 or 1.

In order to compare the three LFSMs, we can extract the corresponding best fault coverage for each circuit from Tables 5.5.1 - 5.5.3 to form the following Table 5.5, then we

LFSR, LHCA and NCCA generators with initial value 0000…0001											
Circuits	Number	Number	Fault Coverage (%)								
			Unrestricted			0			1		
Name	of Inputs	of Vector	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA
S27	4	16	68.75	37.5	90.625	71.875	75	93.75	71.875	37.5	93.75
S386	7	128	30.469	33.854	52.083	34.115	37.5	52.083	30.469	33.854	52.083
S1488	8	256	33.042	47.914	51.346	33.244	48.048	51.48	35.868	49.529	52.894
S1494	8	256	32.337	47.278	50.398	32.537	47.41	50.531	35.126	48.871	51.926
S344	9	512	91.813	58.772	93.86	95.322	73.392	95.906	95.029	61.696	96.784
S349	9	512	91.429	59.143	93.429	94.857	73.429	95.429	94.571	62	96.286
S208	11	2048	26.977	38.14	47.442	32.558	44.186	53.488	37.209	48.372	57.674
S38584	12	4096	18.673	18.72	18.649	48.591	49.966	48.952	58.367	58.243	57.855
S1196	14	16384	92.432	93.639	97.665	92.432	93.639	97.665	92.432	93.639	97.665
S1238	14	16384	87.38	88.561	92.103	87.38	88.561	92.103	87.38	88.561	92.103
S15850	14	16384	0.725	0.725	0.725	5.092	5.109	5.109	18.823	18.883	18.755
S953	16	65536	8.341	8.341	8.341	97.776	99.073	99.073	97.776	99.073	99.073
S1423	17	131072	37.294	59.934	64.554	38.68	61.452	66.271	39.01	62.046	67.063
S820	18	200000	41.529	43.059	48.353	41.647	43.176	48.471	41.882	43.294	48.588
S832	18	200000	40.46	41.954	47.241	40.575	42.069	47.356	40.805	42.184	47.471
S420	19	200000	22.558	20	32.558	28.14	25.581	38.372	30.93	28.372	40.93
S510	19	200000	0	0	0	100	100	100	100	100	100
S9234	19	200000	0.26	0.26	0.26	5.904	5.904	5.904	14.985	14.985	14.985
S38417	28	200000	3.518	3.534	3.538	15.141	13.736	13.204	14.323	14.984	14.57
S13207	31	200000	6.51	6.449	6.205	27.723	27.438	27.438	24.279	24.004	23.943
S35932	35	200000	60.503	84.985	85.936	60.528	85.01	85.962	60.904	85.133	86.077
S5378	35	200000	65.74	69.433	68.129	68.173	70.867	70.28	69.346	72.301	71.214
S641	35	200000	86.51	86.296	86.51	87.366	87.152	87.366	88.009	88.009	88.223
S713	35	200000	81.928	81.756	81.928	82.616	82.444	82.616	83.133	83.133	83.305
S838	35	200000	20.187	25.321	24.854	25.788	31.039	30.572	27.655	32.789	32.322

Table 5.5.1. Expermental Results for ISCAS'89

LFSR, LHCA and NCCA generators with initial value 1111…1110											
Circuits	Number	Number	Fault Coverage (%)								
			Unrestricted			0			1		
Name	of Inputs	of Vector	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA
S27	4	16	71.875	40.625	93.75	71.875	40.625	93.75	71.875	40.625	93.75
S386	7	128	30.469	33.854	51.823	30.469	33.854	52.08	32.292	34.635	52.08
S1488	8	256	33.042	47.914	51.346	33.244	48.048	51.48	37.685	50.808	53.836
S1494	8	256	32.337	47.278	50.398	32.537	47.41	50.531	36.919	50.133	52.855
S344	9	512	91.813	58.772	93.86	93.567	63.158	95.614	94.737	61.404	96.491
S349	9	512	91.429	59.143	93.429	93.143	63.429	95.143	94.286	61.714	96
S208	11	2048	26.977	38.14	39.535	32.558	44.186	45.581	37.209	48.372	49.767
S38584	12	4096	18.83	18.825	18.781	48.602	48.946	48.985	58.665	58.725	58.425
S1196	14	16384	92.432	93.639	97.665	92.432	94.203	97.665	92.512	93.639	97.665
S1238	14	16384	87.38	88.561	92.103	87.38	89.077	92.103	87.528	88.561	92.103
S15850	14	16384	0.725	0.725	0.725	5.501	5.458	5.458	19.48	19.267	19.412
S953	16	65536	8.341	8.341	8.341	97.776	99.073	99.073	97.776	99.073	99.073
S1423	17	131072	37.228	59.934	64.554	39.01	61.452	66.271	39.472	62.178	67.195
S820	18	200000	41.529	43.059	48.353	41.647	43.176	48.471	42.706	44.235	49.529
S832	18	200000	40.46	41.954	47.241	40.575	42.069	47.356	41.609	43.103	48.391
S420	19	200000	22.558	20	32.558	28.14	25.581	38.372	30.93	28.837	40.93
S510	19	200000	0	0	0	100	100	100	100	100	100
S9234	19	200000	0.26	0.26	0.26	5.904	5.904	5.904	14.985	14.985	14.985
S38417	28	200000	3.518	3.534	3.534	13.419	13.736	13.708	14.628	14.984	14.913
S13207	31	200000	6.49	6.184	6.439	27.641	27.499	31.024	23.994	24.727	24.493
S35932	35	200000	64.066	85.146	85.479	64.575	85.172	85.504	71.51	85.89	85.849
S5378	35	200000	65.74	67.738	69.781	67.174	68.933	70.975	69.52	70.91	72.757
S641	35	200000	86.51	86.296	86.51	87.366	87.152	87.366	90.578	88.865	88.865
S713	35	200000	81.928	81.756	81.928	82.616	82.444	82.616	85.026	83.649	83.649
S838	35	200000	20.187	27.421	25.788	25.788	33.139	31.505	27.655	34.889	33.256

Table 5.5.2 Expermental Results for ISCAS'89

LFSR, LHCA and NCCA generators with initial value randam number											
Circuits	Number	Number	Fault Coverage (%)								
			Unrestricted			0			1		
Name	of Inputs	of Vector	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA	LFSR	LHCA	NCCA
S27	4	16	71.875	100	100	71.875	100	100	75	100	100
S386	7	128	30.469	33.854	52.083	33.854	33.854	52.083	30.469	37.76	52.083
S1488	8	256	33.042	47.914	51.346	33.244	48.048	51.48	35.868	49.798	53.096
S1494	8	256	32.337	47.278	50.398	32.537	47.41	50.531	35.126	49.137	52.125
S344	9	512	91.813	58.772	93.86	95.322	66.374	96.491	95.322	61.696	96.784
S349	9	512	91.429	59.143	93.429	94.857	66.571	96	94.857	62	96.286
S208	11	2048	26.977	38.14	39.535	32.558	44.186	45.581	40	49.302	50.698
S38584	12	4096	18.83	18.825	18.759	48.756	48.974	49.844	58.491	58.824	58.048
S1196	14	16384	92.432	93.639	97.665	92.432	93.639	97.665	92.432	93.639	97.665
S1238	14	16384	87.38	88.561	92.103	87.38	88.561	92.103	87.38	88.561	92.103
S15850	14	16384	0.725	0.725	0.725	5.424	5.518	5.527	19.318	19.574	19.497
S953	16	65536	8.341	8.341	8.341	97.776	99.073	99.073	97.776	99.073	99.073
S1423	17	131072	37.294	59.934	64.422	38.746	61.452	66.007	39.076	62.046	66.997
S820	18	200000	41.529	43.059	48.353	41.647	43.176	48.471	43.059	43.294	48.588
S832	18	200000	40.46	41.954	47.241	40.575	42.069	47.356	41.954	42.184	47.471
S420	19	200000	22.558	20	34.186	28.14	25.581	40	36.744	30.698	44.419
S510	19	200000	0	0	0	100	100	100	100	100	100
S9234	19	200000	0.26	0.26	0.26	5.904	5.904	5.904	14.985	14.985	14.985
S38417	28	200000	3.518	3.538	3.534	12.611	13.477	13.271	14.833	14.952	14.91
S13207	31	200000	6.449	6.205	6.439	30.973	29.536	29.74	24.738	24.687	24.055
S35932	35	200000	59.579	85.453	85.41	59.605	85.479	85.435	60.347	85.732	85.701
S5378	35	200000	65.74	66.609	69.976	67.369	68.064	71.062	71.171	69.998	72.822
S641	35	200000	86.51	86.296	86.51	87.366	87.152	87.366	89.936	88.651	89.507
S713	35	200000	81.928	81.756	81.928	82.616	82.444	82.616	84.682	83.649	84.337
S838	35	200000	20.187	25.554	24.504	25.788	31.272	30.222	36.523	36.406	35.473

Table 5.5.3. Expermental Results for ISCAS'89

conclude:

- obviously NCCA have better fault coverage than LHCA and LFSR, in Table 5.5, NCCA show the best fault coverage in the 19 circuits of the 25 circuits; LHCA show the best fault coverage in the 6 cirucits of the 25 circuits; However, LFSR show the best fault coverage in the 6 circuits of the 25 circuits.
- for a few circuits, LHCA or LFSR have a slightly higher fault coverage than NCCA such as in the circuit s15850, the fault coverages for LHCA and NCCA are 19.574 and 19.497; in the circuit s38417, the fault coverages for LFSR and NCCA are 15.141 and 14.913. .
- LHCA have better fault coverage than LFSR. In Table 5.5, in 15 of the 25 circuits, LHCA have higher fault coverage than LFSR.
- comparing to NCCA and LHCA, LFSR have the lowest fault coverage in the most circuits.

Note: In the circuit s510, 0% fault coverage is achieved when all of flip-flops are at the default (unrestricted) value, but 100% fault coverage is achieved when all of flip-flops are set to 0 or 1. Why? [4] explains: " when power is applied to a circuit, the sequential circuits may stabilize to either a logical 0 or 1. Before testing can begin they must all be brought to a known state. This require finding an input sequence that will take the circuit from an unpredictable power-on state to some known starting state. ..., either by initialization circuit that resets all sequential elements at power-on, ... " Hence, for the circuit s510 at the unrestricted state, no input pattern can take the circuit to a known starting state.

For the stuck-at faults in sequential circuits, the fault coverage is depended on the number of distinct transitions. In [21, 22], it was proven that LHCA have the bigger transition capacity than LFSR on the average. In chapter 3, we also prove that NCCA have more transitions than LHCA on the average. The experimental results are consistent with the above research.

Circuits Name	Number of Inputs	Number of Vectors	Best Fault Coverage (%) among Tables 5.5.1 - 5.5.3		
			LFSR	LHCA	NCCA
S27	4	16	75	100	100
S386	7	128	34.115	37.76	52.083
S1488	8	256	37.685	50.808	53.836
S1494	8	256	36.919	50.133	52.855
S344	9	512	95.322	73.392	96.784
S349	9	512	94.857	73.429	96.286
S208	11	2048	40	49.302	57.674
S38584	12	4096	58.665	58.824	58.425
S1196	14	16384	92.512	94.203	97.665
S1238	14	16384	87.528	89.077	92.103
S15850	14	16384	19.48	19.574	19.497
S953	16	65536	97.776	99.073	99.073
S1423	17	131072	39.472	62.178	67.195
S820	18	200000	43.059	44.235	49.529
S832	18	200000	41.954	43.103	48.391
S420	19	200000	36.744	30.698	44.419
S510	19	200000	100	100	100
S9234	19	200000	14.985	14.985	14.985
S38417	28	200000	15.141	14.984	14.913
S13207	31	200000	30.973	29.536	31.024
S35932	35	200000	71.51	85.89	86.077
S5378	35	200000	71.171	72.301	72.822
S641	35	200000	90.578	88.865	89.507
S713	35	200000	85.026	83.649	84.337
S838	35	200000	36.523	36.406	35.473

Table 5.5. *Experimental Results for ISCAS'89*

5.3 Summary

Fault simulation is used to evaluate the performance of different BIST generators: NCCA, LHCA and LFSR. For testing the combination circuits, the three LFSMs can attain the 100% fault coverage. Based on the results of ISCAS'89, we have confirmed the conclusion of our analysis of transition properties presented in chapter 3.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we have proposed a new class of cellular automata (NCCA) and have investigated the properties of the maximum-length NCCA.

No previous work in the maximum length NCCA has been done. Hence, we have defined the notation and computation rules, and presented the mathematical background for analysis. In order to further analyze the properties, the recursive relation of NCCA and maximum number of transitions of NCCA are derived. Based on the theoretical analysis and experimental results, the test sequences produced by NCCA have more transitions than those produced by LFSR and LHCA.

In order to evaluate the pseudorandom properties of NCCA, Knuth's randomness tests suggested in [14] are employed and the experimental results are compared with those of corresponding LFSR and LHCA. The results show that NCCA have a slightly better randomness properties than LHCA and much better randomness properties than LFSR which are most widely applied as pseudorandom test generators. Therefore, based on Knuth's randomness tests, NCCA are the most suitable pseudorandom pattern generators among the three.

We have also performed a feasibility study on the behavior of the maximum-length NCCA as pseudorandom test pattern generators by simulation on the standard benchmark circuit sets ISCAS'85 and ISCAS'89, and it is shown that NCCA as pseudorandom pattern generators have the better fault coverage than LFSR and LHCA as pseudorandom pattern generators for testing sequential circuits.

6.2 Future Work

There are many interesting aspects of our work that need be further studied. The most important of them are as follows:

- In order to obtain more general conclusions, we should employ the Knuth's random-

ness tests and simulate the benchmarks circuits with both large-sized LFSMs and the same-sized LFSMs which have different primitives and the large size LFSMs.

- The study of the behavior of the maximum-length NCCA as pseudorandom generators for stuck-open faults and delay faults should be investigated.
- More complex CA, such as other computation rules, should be studied in order to find out the relationship between the structure of CA and transition properties.
- The Knuth's randomness tests can not supply a strong proof to evaluate the randomness of binary sequences. Hence, other effective methods should be explored.

Bibliography

[1] Cellular automata. Available at: http://www.brunel.ac.uk/depts/AI/alife/al-ca.htm.

[2] Critical values of the chi-square distribution. Available at: http://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm.

[3] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital System Testing and Testable Design*. IEEE Press, 1990.

[4] P. H. Bardell, W. H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques*. A Wiley-Interscience Publication, 1987.

[5] D. Bryan. The ISCAS 1985 Benchmark Circuits and Netlist Format, [online document]. Available at: http://www.cbl.ncsu.edu/CBL_Docs/iscas85.html, Sept 1988.

[6] K. Cattell and J. C. Muzio. Synthesis of One-Dimensional Linear Hybrid Cellular Automata. *IEEE Transactions on Computer-aided Design*, 15(3):pages 325–335, 1996.

[7] K. Cattell, S.Zhang, M. Serra, and J. C. Muzio. 2-by-n Hybrid Cellular Automata with Regular Configuration: Theory and Application. *IEEE Transactions on Computers*, 48(3):pages 285–295, 1999.

[8] K. M. Cattell and J.C. Muzio. *Table of Linear Cellular Automato for minimal weight primitive polynomials of degrees up to 300*. Computer Science Department, University of Victoria, 1991.

[9] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *IEEE International Symposium on Circuits and Systems, 1989*, pages 1929–1934, New York, NY, USA, May 1989.

[10] F. Brglez, D. Bryan, and K. Kozminski. Notes on the ISCAS'89 benchmark circuits, [online document]. Available at: http://www.cbl.ncsu.edu/ CBL_Docs/iscas89.html, Oct 1989.

[11] F. Brglez and H. Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran. In *IEEE International Symposium on Circuits and Systems. Proceedings, 1985*, pages 1–1, New York, NY, USA, Jun 1985.

[12] P. D. Hortensius, R. D. Mcleod, and H. C. Card. Parallel Random Number Generation for VLSI Systems Using Cellular Automata. *IEEE Transaction on Computers*, 38(10):pages 1466–1473, 1989.

[13] Hassan Janoowalla. Analysis of maximum length linear two-dimensional cellular automata. Master's thesis, Department of Computer Science. The University of Victoria, 1992.

[14] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms, Third Edition.* Addison Wesley, 1997.

[15] D. M. Miller, S. Zhang, R.D. Mcleod, and W. Pries. Estimating Aliasing in CA and LFSR Based Signature Register. *In Proceedings of IEEE International Conference on Computer Design*, pages 157–160, 1990.

[16] D. K. Pradhan. *Fault Tolerant Computing: Theory and Techniques.* Prentice Hall, 1986.

[17] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller. The Analysis of One-Demensional Linear Cellular Automata and Their Aliasing Properties. *IEEE Transactions on Computer-aided Design*, 9(7):pages 767–778, July 1990.

[18] H. S. Stone. *Discrete Mathematical Structures and Their Application.* New York: Science Research Association, 1973.

[19] S. Wolfram. Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics*, pages 601–644, 1983.

[20] Stephen Wolfram. Universality and Complexity in Cellular Automata. *Physica 10D*, pages 1–35, 1984.

[21] S. Zhang, R. Byrne, and D. M. Miller. BIST Generators for Sequential Faults. *In Proceedings of IEEE International Conference on Computer Design*, pages 260–263, 1992.

[22] S. J. Zhang. BIST generators for faults with sequential behavior. Master's thesis, Department of Computer Science. The University of Victoria, 1993.

[23] Jing Zhong, Lin Sun, and Yongzhi Yu. Pseudo-Random and Transition Tests for Binary Sequences, 2002.

Appendix A

The Primitive Polynomials of the Minimum-Cost LFSR for Degrees 1 through 60

Degree n	the Primitive Polynomials of Minimal-Cost LFSR
1	$X + 1$
2	$X^2 + X + 1$
3	$X^3 + X + 1$
4	$X^4 + X + 1$
5	$X^5 + X^2 + 1$
6	$X^6 + X + 1$
7	$X^7 + X + 1$
8	$X^8 + X^7 + X^2 + X + 1$
9	$X^9 + X^4 + 1$
10	$X^{10} + X^3 + 1$
11	$X^{11} + X^2 + 1$
12	$X^{12} + X^8 + X^2 + X + 1$
13	$X^{13} + X^5 + X^2 + X + 1$
14	$X^{14} + X^{12} + X^2 + X + 1$
15	$X^{15} + X + 1$
16	$X^{16} + X^{12} + X^3 + X + 1$
17	$X^{17} + X^3 + 1$
18	$X^{18} + X^7 + 1$
19	$X^{19} + X^5 + X^2 + X + 1$
20	$X^{20} + X^3 + 1$
21	$X^{21} + X^2 + 1$
22	$X^{22} + X + 1$
23	$X^{23} + X^5 + 1$
24	$X^{24} + X^7 + X^2 + X + 1$
25	$X^{25} + X^3 + 1$
26	$X^{26} + X^6 + X^2 + X + 1$
27	$X^{27} + X^5 + X^2 + X + 1$
28	$X^{28} + X^3 + 1$
29	$X^{29} + X^2 + 1$
30	$X^{30} + X^{23} + X^2 + X + 1$

Degree n	the Primitive Polynomials of Minimal-Cost LFSR
31	$X^{31} + X^3 + 1$
32	$X3^2 + X^{22} + X^2 + X + 1$
33	$X^{33} + X^{13} + 1$
34	$X^{34} + X^{27} + X^2 + X + 1$
35	$X^{35} + X^2 + 1$
36	$X^{36} + X^{11} + 1$
37	$X^{37} + X^9 + X^2 + X + 1$
38	$X^{38} + X^{13} + X^3 + X + 1$
39	$X^{39} + X^4 + 1$
40	$X^{40} + X^{35} + X^2 + X + 1$
41	$X^{41} + X^3 + 1$
42	$X^{42} + X^{29} + X^2 + X + 1$
43	$X^{43} + X^{12} + X^2 + X + 1$
44	$X^{44} + X^{38} + X^3 + X + 1$
45	$X^{45} + X^4 + X^3 + X + 1$
46	$X^{46} + X^9 + X^3 + X + 1$
47	$X^{47} + X^5 + 1$
48	$X^{48} + X^{28} + X^3 + X + 1$
49	$X^{49} + X^9 + 1$
50	$X^{50} + X^{16} + X^2 + X + 1$
51	$X^{51} + X^{28} + X^2 + X + 1$
52	$X^{52} + X^3 + 1$
53	$X^{53} + X^6 + X^2 + X + 1$
54	$X^{54} + X^{17} + X^2 + X + 1$
55	$X^{55} + X^{24} + 1$
56	$X^{56} + X^{42} + X^2 + X + 1$
57	$X^{57} + X^7 + 1$
58	$X^{58} + X^{19} + 1$
59	$X^{59} + X^{24} + X^2 + X + 1$
60	$X^{60} + X + 1$

Appendix B

Knuth's Randomness Tests

Knuth's randomness tests in this testbench are used to evaluate the randomness of sequences. Six tests are included here: Equidistribution test, Serial-2 test, Poker-3 test, Gap test, Runup test, and Permutation test. For detailed information please refer to "The Art of Computer Programming", written by Knuth.

The tests here are for binary sequences and conversion is necessary. We convert N (N $\leq$ 100,000) successive n-bit (n $\leq$ 150) binary sequences into the corresponding decimal-value sequences, and the decimal-value sequences are further converted into the decimal-value sequences whose elements range from 0 to d-1 by taking their modulo d. Then we employ the Knuth tests on the latter decimal-value sequneces. If the sequences pass a certain number of empirical tests, they are considered to be random enough.

Example

Suppose a binary sequence of N items, with N = 31, with each items of n bits, with n=5, is as saved as follows in a file named "Mytest": 00001

00010

00100

01001

10010

00101

01011

10110

01100

11001

10011

00111

01111

11111

11110

11100

11000

10001

00011

00110

01101

11011

10111

01110

11101

11010

10101

01010

10100

01000

10000

Step 1: Select the name of the file.

Step 2: input an integer d for the modulo, e.g. "3".

Step : Choose the desired Knuth test, e.g. "Gap test".

Step 4 : a sample output file may look like:

*** SUMMARY OF GAP TEST RESULTS ***

Input File: Mytest Number of Vectors:31 Modulo: 3

GapV=4.571424 Chi-Table show:

For the degree v= 4

Probability 0.95 is: 0.711

Probability 0.05 is: 9.488

So the sequences PASS the Gap Test.

Appendix C

Random Numbers Used in Chapter 5

In Chapter 5, in order to evaluate the behaviors of different LFSMs as pseudorandom generators, we employ the benchmark circuits ISCAS'85 and ISCAS'89 by using the different initial vectors: 000...01, 111...10. and a random number. The random number is generated by a random generator. For all three LFSMs with the same length, the same random number is applied as their initial vectors. Here, we list the random numbers used in Chapter 5:

4-bit: 0110

7-bit: 0110010

8-bit: 01100101

9-bit: 011001011

11-bit: 01100101100

12-bit: 011001011000

14-bit: 01100101100010

16-bit: 0110010110001001

17-bit: 01100101100010010

18-bit: 011001011000100101

19-bit: 0110010110001001010

28-bit: 0110010110001001010011111100

31-bit: 0011001011000100101001111110011

32-bit: 01100101100010010100111111001110

33-bit: 00110010110001001010011111001110

35-bit: 00110010110001001010011111100111001

36-bit: 001100101100010010100111111001110011

50-bit: 00110010110001001010011111100111001101000000010010

Appendix D

Cost Analysis

Chapter 3-5 proved that NCCA are superior to LFSR and LHCA on the different metrics (transitions, randomness and fault coverage). However, it is obvious that NCCA have more complex structure than LHCA, and much more complex structure than LFSR, which means that NCCA should have the higher price than the other LFSMs. It is very difficult to estimate the cost of circuits in practice, but commonly we adopt the simple method that count the number of XOR gates in the three LFSMs. Here the number of XOR gates of all minimum-cost LFSMs applied in benchmark circuits ISCAS'89 is listed in Table D-1.

- From Table D-1, the number of NCCA's XOR gates is almost twice that of LHCA and more than 4 times that of LFSR.
- Tradeoff of cost and high performance is dependent on the need of customers. If high performance is preferred, cost may have to rise.

Circuits Name	Number of Inputs	The Number of XOR Gates of the Minimum-Cost LFSMs		
		LFSR	LHCA	NCCA
S27	4	1	4	5
S386	7	1	6	11
S1488	8	3	7	14
S1494	8	3	7	14
S344	9	1	8	16
S349	9	1	8	16
S208	11	1	10	20
S38584	12	3	12	22
S1196	14	3	13	25
S1238	14	3	13	25
S15850	14	3	13	25
S953	16	3	16	30
S1423	17	1	16	32
S820	18	1	18	34
S832	18	1	18	34
S420	19	3	18	36
S510	19	3	18	36
S9234	19	3	18	36
S38417	28	1	27	54
S13207	31	1	30	60
S35932	35	1	34	68
S5378	35	1	34	68
S641	35	1	34	68
S713	35	1	34	68
S838	35	1	34	68

Table D.1. *The Number of XOR gates of the Minimum-Cost LFSMs*