Cellular Automata Evolution : Theory and Applications in Pattern Recognition and Classification

Niloy Ganguly

A report in partial fulfillment for the Degree of

Doctor of Philosophy

in

Computer Science & Technology

Under the supervision of

Prof. P Pal Chaudhuri

Dept. of Computer Science Technology
Bengal Engineering College(D U)

Department of Computer Science & Technology
Bengal Engineering College (Deemed University)
October, 2002

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Researchers in artificial life conceive the universe as a computer implementing transformations of information. If the universe can be viewed as a computation, it should be possible to build computing models of physical systems of the universe [53]. Many people in Artificial life have been enamored of a mathematical formalism of computing model known as the Cellular Automata. This modeling tool can be regarded as parallel processing computer. In an abstract sense, it can be also viewed as a logical universe with its own local physics and with emergent structures as artificial molecules. The researchers have explored the possibility of building Cellular Automata model having the capacity for self-reproduction and other essential functions of biomolecules leading to the possibility of life-like behavior. As a result, *Cellular Automata* (CA) came to be just as essential to the study of physical systems as the microscope was to study microbes and the telescope to the exploration of deep space.

The cellular automata (CA) is sufficiently complex to develop an entire universe as sophisticated as the one in which we live. Our own universe might be thought of as one mammoth Cellular Automaton. CA allows a programmer to specify rules for local interaction between 'cells' on a lattice-like grid and to study the emergent consequences of those rules. Von Neumann [40] [52] first envisioned that proper specification of rules empower CA to build models simulating bacterial growth, the growth of patterns on seashells, fluid dynamics, and the voting patterns of individuals who made decisions based on their local neighbors.

One of the most important milestones in the history of development of the simple homogeneous structure of cellular automata is due to Wolfram [54]. The suggested simplification leads to a one/two dimensional structure of simple cells, each having only two states with uniform three-neighborhood dependence. This simplified structure motivated a number of researchers [9] to undertake the study of CA behavior with Linear/Additive next state function amenable for matrix algebraic analysis. This new group of researchers explored innovative applications of this simple, regular, modular, and cascadable structure of CA machine.

However, some important theoretical constraints and design issues partially restrict the capability of this modeling tool. The construction/synthesis/search of Cellular Automata(CA) having the ability to simulate a given modeling task is extremely difficult. Scientists have emulated the evolution mechanism of the living organisms of the universe to solve this hard problem. Evolution is mostly directed by the popular genetic algorithm whose underlying philosophy is *survival of the fittest.* The fitter rules slowly overpower the less fit rules to arrive at the CA with the desired configuration. In order to implement genetic algorithm, the rules of CA are encoded in chromosome format. The scheme is termed by a variety of name like $EVCA$, $CAGA$ etc [36].

However, the scheme suffers from a few inherent problems, discussed next. First, in order to find the fit rules, GA has to traverse a huge search space. For example, in an n-cell CA, if each cell has a modest 256 different rule options, the search space becomes a staggering 256^n. Secondly, specific to a problem, we can analytically discard many of the rules/ rule sequences. Alternatively, we may be able to specify the particular rule sequences as candidate solutions of the problem. However, when we implement genetic algorithm, it is difficult for the designer to accommodate such analysis into consideration. The genetic algorithm has to consider all those discarded rule sequences as the candidate solution to the problem.

In order to overcome the above problems, one of the major challenges in the research of CA evolution is to develop schemes which can initiate evolution within a sub-class of CA. A subclass differ from a class of CA in the sense that not only its state transition behavior depends upon a subset of rules but it also depends upon the sequence in which the rules are arranged. The conventional chromosome operations of GA cannot be applied for such solution, since a different sequence of the bits of the same chromosome will take us out of the search domain. The thesis handles this problem in the domain of Linear/Additive Cellular Automata and shows that vector algebraic analysis embedded in genetic algorithm can drastically reduce the search space.

The last decades of twentieth century have witnessed a colossal stride in the power of computation and communication leading to Internet Technology. While pushing human civilization to new heights, such a large stride has opened up new challenges to the human society of cyber age. One of the major challenges we are facing is the exponential explosion of data in every field of our daily life. The need of the hour is to have good knowledge extraction methodology to extract meaningful and perceptible information from the large volume of data that are apparently uncorrelated and random in nature.

In the above context, Pattern Recognition/Classification has become an extremely important class of problem of the internetworked society of the twenty first century. In this class of problems, a machine identifies patterns of interest from its background. Several type of machines, most notable among them is neural network, has been proposed and widely used for solving Pattern Recognition/Classification problems. However, in order to tackle the large volume of data-set and as well as to have fast decisions, high speed, low cost hardwired implementation of Pattern Recognition & Classification algorithm/machine

is a necessity. The sparse network of cellular automata (CA) holds enormous importance and potential in this domain.

The thesis reports a comprehensive approach in tackling the problems of Pattern Recognition and Classification. The versatility of CA modeling tool is explored in this dissertation to identify special subclasses of Linear/Additive CA which can perform Pattern Recognition/Classification task. A framework for efficient evolution of desired with required configuration has been proposed.

1.1 Aim of the dissertation

The major focus of this dissertation is to explore the applications of the powerful modeling tool of Cellular Automata in two important areas - Pattern Classification and Pattern Recognition. This demands a framework of CA evolution to meet the challenges of such ubiquitous applications. In order to design efficient evolutionary algorithms, an analytical foundation for characterization of Linear/Additive CA is an essential prerequisite.

In the above background this thesis concentrates on building the theoretical framework for complete analysis of CA state transition behavior in terms of its cyclic/non-cyclic vector subspaces. The solution to the reverse problem of CA synthesis is also necessary to realize the stated objective of this dissertation.

Once the theoretical foundation of CA analysis and synthesis is laid down, the framework for CA evolution is developed. The framework enables evolution of different types of CA. The evolution scheme enables us to direct search within Group CA - that is, the sub-class of CA whose state transitions always form cycles. The scheme can be further extended to direct search within a special type of Group CA to drastically reduce the search space. A special sub-class of Non-Group CA which forms basins of attraction has been also evolved with the framework. The evolution scheme can be directed in such a manner so that the number of basins either dynamically changes over generations or it remains same throughout the period of evolution.

The evolution of Non-Group Linear/Additive CA has provided the platform to build the CA based model for design of a general Pattern classifier. The same sub-class of Non-Group CA is also used to model associative memory.

Non-Linear CA evolved with the framework adds versatility and strength to the CA model. This model has been also employed for design of associative memory and its consequent application for Pattern Recognition. The comparison between the two models of associative memory realized through Linear and Non-Linear CA has been also investigated.

1.2 Organization of the Dissertation

Prior to dealing with CA theory and its applications we have reported in this dissertation, a comprehensive survey of the relevant research publications is undertaken. The survey reported in *Chapter 2* covers the following aspects:

(i) a general study of Linear/Additive CA;

(ii) application of Genetic Algorithm for CA evolution;

(iii) the schemes employed to address Pattern Recognition/Classification problems, their strength and weakness; and

(iv) CA application reported so far to solve Recognition/Classification problems

Chapter 3 reports the complete characterization of an Additive CA in terms of its vector subspaces. The chapter establishes an analytical framework to study the state transition behavior of a Linear/Additive CA in terms of its cyclic and non-cyclic subspaces.

If analysis is viewed as one side of a coin, its reverse is 'Synthesis'. The *Chapter 4* presents the schemes for synthesizing a Linear/Additive CA from the given cyclic/non-cyclic structure of its state transition behavior.

Once the general analysis and synthesis methodologies are in place, we start characterizing special sub-classes of Linear CA and start building their applications. Two most important applications of the internetworked society of twenty-first century are addressed in subsequent chapters. These are Pattern Classification and Pattern Recognition. A special class of Non-Group CA termed as Multiple Attractor Cellular Automata($MACA$) is found capable of performing the above two tasks.

Chapter 5 establishes Multiple Attractor Cellular Automata($MACA$) as a special type of hash family named as Hamming Hash Family(HHF). The probability of collision of two patterns in HHF varies inversely with the hamming distance between them. Beside establishing the property of $MACA$, this chapter also reports a special type of Constrained Genetic Algorithm which ensures the directed search be restricted within the $MACA$ family and consequently evolve the desired $MACA$ required for a specified problem.

In *Chapter 6* we report two important applications of $MACA$. The first application is design of a $MACA$ based Pattern classifier. The basic scheme is of a two class classifier which have been hierarchically employed to design a set of $MACA$ for classifying patterns of multiple classes. Secondly, the basins of attraction of $MACA$ has been used for building associative memory model. The performance of $MACA$-based associative memory has been enhanced by developing a system comprising of multiple $MACA$ instead of single $MACA$. Majority voting by such multiple $MACA$ enhances noise immunity of the system. Various aspects of this associative memory model and its application for Pattern Recognition is the major focus of this chapter.

We have so far dealt with the evolution of Non-Group CA. While extending the study for Group CA evolution, we observed an interesting phenomena of practical relevance. Pseudo-random patterns are employed in many applications such as simulation, testing $VLSI$ circuits etc. Some of these applications demand generation of pseudo-random patterns without a given set of prohibited patterns. Group CA have been evolved in *Chapter 7* to address this specific problem, that is generation of patterns without a given prohibited pattern set while maintaining the desired pseudo-random quality. The solution proposed has its direct relevance for design of Pseudo-Random Pattern Generator for testing $VLSI$ circuits.

All the earlier chapters have dealt with Linear/Additive CA that employ $XOR/XNOR$ CA rules characterized in *Chapter 2,3,4* . On the other hand, Non-Linear CA employs Non-Linear rules that are not amenable to such analysis. So Non-Linear CA are evolved with Genetic Algorithm to design generalized $MACA$ referred to as $GMACA$. The *Chapter 8* establishes $GMACA$ as an efficient model of an Associative Memory. The experimental results of the two models of associative memory - $GMACA$ and Multiple $MACA$ - are next subjected to comparative study.

The final chapter concludes the thesis and points out the future direction of research.

Chapter 2

Survey

Terminologies

$f(x)$	Characteristic polynomial of the Characteristic Matrix T
CS	Cycle Structure generated by a CA
$\mathcal{N}$	No. of elementary divisors in a characteristic polynomial.
$\tilde{\mathcal{N}}$	No. of $PFCS$ comprising an LCA
k_i	The length of the i^{th} primary cycle
$2^j.k_i$	Cycle length of secondary cycle generated out of i^{th} primary cycle with length k_i, j is an integer
PCS	Primary Cycle Structure representing the cycle structure arising from a single elementary divisor $\phi_i(x)^{n_i}$
$PFCS$	Primary Family Cycle Structure - Cycle structure arising from a set of elementary divisor set $[\phi(x)^{n_1} \cdot \phi(x)^{n_2} \cdots \phi(x)^{n_{\mathcal{N}}}]$ each elementary divisor having the same primary cycle
j^{th} Factor Set	The factors of an elementary divisor $\phi_i(x)^{n_i}$ starting from 2^{j-1} to 2^j. Important point to be noted all these factors produce the same cycle length $2^j \cdot k_i$, where k_i is the primary cycle of the factor.
$N(F_j)$	Cardinality of the j^{th} Factor Set in a single elementary divisor.
$N(\tilde{F}_j)$	Cardinality of the j^{th} Factor Set in a set of elementary divisors - each having same primary cycle
$S_{2^j \cdot k_i}[2^j \cdot k_i]$	where $S_{2^j \cdot k_i}$ means number of states covered by cycle length ($\leq 2^j \cdot k_i$), that is $S_{2^j \cdot k_i} = \sum_{j=0}^{j} \mu_{2^j \cdot k_i} \times (2^j \cdot k_i)$

Chapter 3

GF(2) Cellular Automata - Analysis

A comprehensive survey of Cellular Automata (CA) as a computing model and its applications in diverse fields is presented in *Chapter 2*. A breakthrough in the study of CA was initiated with the introduction of local neighborhood CA with 2-state per cell whose operations are defined over the Galois Field 2 and is referred to as GF(2) CA. It has a simple elegant structure. In recent years it has been proposed as a powerful modeling and computing paradigm [6, 19, 13, 15, 38]. However, while developing CA based models it has been observed that much more deep insight into the GF(2) CA is needed to exploit the full potential of this class of CA. Although, there are various studies which confirm the absolute rich variety in the state transition behavior of GF(2) CA, scientists fail to artificially synthesize a desired state transition behavior required to map a particular problem. This prompts us to develop methodology to synthesize GF(2) CA for a given state transition behavior. This chapter and the chapter next is specifically aimed to solve the problem.

In order to achieve the aim of synthesis a more detailed characterization of GF(2) CA is required. The GF(2) CA has an important subclass of namely Linear Cellular Automata(LCA)- the CA which uses XOR logic to generate its next state function. The characterization of linear CA, the most important subclass of GF(2) CA has been widely done by [51] as part of his exploration of linear machine and later by [9, 17]. We take a more detailed look into the characterization, dig out some hindsight from the established results. However, the characterization of the more generalized class of CA - the Additive Cellular Automata(ACA) - the CA which uses both XOR and $XNOR$ logic as its next state function - hasn't been done so far, we in this chapter do a thorough characterization of additive CA.

In the above background this chapter is organized as follows. In order to develop the vector space theoretic analysis of GF(2) CA, we report the GF(2) CA preliminaries from [9, 46] in *Section 3.1*. The vector space theoretic analysis of LCA, the essence of which is reported in [9, 50], is noted in *Section 3.2*. However, in this section, we reorient the

algorithm in such a fashion such that it provides insight to the reverse problem of synthesis in the next chapter. The vector space theoretic analysis of ACA is reported in *Section 3.3* .

3.1 GF(2) Cellular Automata

A Cellular Automata consists of a number of interconnected cells arranged spatially in a regular manner. In most general case, a CA cell can exhibit m different states and the next state of each cell depends upon the present states of its k number of neighborhood including itself. Such a generalized CA is called a m-state k-neighborhood CA. However, Wolfram worked with several features of finite CA known as 3-neighborhood (left, right and self) CA having 2-state for each cell. The state $q \in \{0,1\}$ of the i^{th} cell at time $(t+1)$ is denoted as

$$q_i^{t+1} = f(q_{i-1}^t, q_i^t, q_{i+1}^t),$$

where q_i^t denotes the state of the i^{th} cell at time t and f is the next state function called the rule of the automata [54]. Since f is a function of 3 variables, there are 2^{2^3} or 256 possible next state functions. The structure of a 3-neighborhood CA cell is shown in *Figure 3.1.*

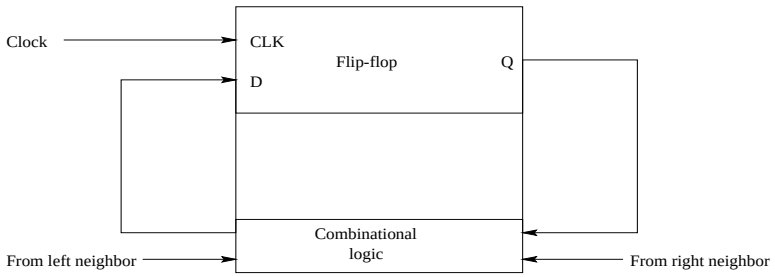


Figure 3.1: A 3-neighborhood CA cell.

If the next state function of a cell is expressed in the form of a truth table, then the decimal equivalent of the output column is conventionally called the *rule number* for the cell. The following is an illustration for two such rules, 90 and 150:

Neighborhood :	111	110	101	100	011	010	001	000	$RuleNo$
(i) NextState :	0	1	0	1	1	0	1	0	90
(ii) NextState :	1	0	0	1	0	1	1	0	150

The first row lists the possible combinations of present states (left, self and right) for a 3-neighborhood CA cell at time t. The next two rows list the next states for the i^{th} cell at time instant $(t+1)$ - the decimal equivalent of the 8 bit binary number is referred to as Rule Number. Each rule can be realized by a set of logic functions. The logic functions for different rules which uses $XOR/XNOR$ logic are noted in *Table 3.1.* The q_{i-1}, q_i, and q_{i+1} denote the state of the $(i-1)^{th}$, i^{th}, and $(i+1)^{th}$ cells respectively at t^{th} instant of time. The following subsection defines a few terminologies that are widely used in literature and has been referred to throughout this thesis. These terminologies are quoted from [9].

Table 3.1: Additive CA Rules

With XOR (Linear CA)			With $XNOR$ (Complemented rule)		
rule 60	:	$q_i(t+1) = q_{i-1} \oplus q_i$	rule 195	:	$q_i(t+1) = \overline{q_{i-1} \oplus q_i}$
rule 90	:	$q_i(t+1) = q_{i-1} \oplus q_{i+1}$	rule 165	:	$q_i(t+1) = \overline{q_{i-1} \oplus q_{i+1}}$
rule 102	:	$q_i(t+1) = q_i \oplus q_{i+1}$	rule 153	:	$q_i(t+1) = \overline{q_i \oplus q_{i+1}}$
rule 150	:	$q_i(t+1) = q_{i-1} \oplus q_i \oplus q_{i+1}$	rule 105	:	$q_i(t+1) = \overline{q_{i-1} \oplus q_i \oplus q_{i+1}}$
rule 170	:	$q_i(t+1) = q_{i+1}$	rule 85	:	$q_i(t+1) = \overline{q_{i+1}}$
rule 204	:	$q_i(t+1) = q_i$	rule 51	:	$q_i(t+1) = \overline{q_i}$
rule 240	:	$q_i(t+1) = q_{i-1}$	rule 15	:	$q_i(t+1) = \overline{q_{i-1}}$

3.1.1 Cellular Automata Terminologies

Definition 3.1 *If all the CA cells obey the same rule, then the CA is said to be a uniform CA; otherwise it is a hybrid CA.*

Definition 3.2 *A CA is said to be a Null Boundary CA ($NBCA$) if the left(right) neighbor of the leftmost(rightmost) terminal cell is connected to logic 0-state.*

Throughout the thesis, we have used Null Boundary Hybrid CA.

Definition 3.3 *If the next-state generating logic employs only XOR logic then it is called a linear rule.*

Definition 3.4 *Rules involving $XNOR$ logic are referred to as complemented rules.*

Definition 3.5 *A CA with all the cells having linear rules is called a linear CA (LCA).*

Definition 3.6 *CA having a combination of XOR and $XNOR$ rules is called additive CA (ACA).*

The CA using XOR or $XNOR$ rules follow the property of Galois Field 2. Hence it is also termed as GF(2) CA.

3.1.2 Characterization of GF(2) Cellular Automata

Polynomial and matrix algebraic tools are used to formulate the global properties of CA. The state of an n-cell CA at time t can be represented by a characteristic polynomial [34]

$$P^{(t)}(x) \;=\; \sum_{i=0}^{n-1} q_i^{(t)} \times x^i$$

where the value of the i^{th} cell is the coefficient of x_i and belongs to GF(2). Extensive analysis of linear CA in terms of topological characterization of the state transition graph based on irreversibility, cyclic components, depths, etc. has been carried out with polynomial algebra.

Following the work reported in [34], Pries et. al. [45] have also studied the state transition behavior of linear one dimensional uniform CA. A comparatively new analytical tool based on *matrix algebra* [9, 18, 21] has been developed to overcome the limitations of existing tools based on polynomial algebra. In our work we concentrate on the matrix algebraic model to characterize the CA. A brief overview of this model is next outlined.

An n-cell one dimensional additive GF(2) cellular automata is characterized by a linear operator $[T]_{n\times n}$ matrix and an n-dimensional inversion vector referred to as F vector[9]. T is referred to as the *characteristic* matrix of the cellular automata. The i^{th} row of T corresponds to the neighborhood relation of the i^{th} cell, where

$$T_{ij} = \begin{cases} 1, & \text{if the next state of the } i^{th} \text{ cell depends on the present state of the } j^{th} \text{ cell} \\ 0, & \text{otherwise.} \end{cases}$$

Since the CA is restricted to three neighborhood dependency, therefore, $T[i, j]$ can have non-zero values for $j = (i-1), i, (i+1)$.

In addition to the operator T, an ACA employs the inversion vector F defined as

$$F_i = \begin{cases} 1, & \text{if the next state of the } i^{th} \text{ cell results from inversion} \\ 0, & \text{otherwise.} \end{cases}$$

If s_t represents the state of the automata at the t^{th} instant of time, then the next state - that is, the state at the $(t+1)^{th}$ time, is given by [9]:

$$s_{(t+1)} = T \cdot s_t + F, \quad that\ is, \quad s_{(t+p)} = T^p \cdot s_t + (I + T + T^2 + \cdots + T^{p-1})F, \tag{3.1}$$

where for an n cell CA, F is the n bit *Inversion Vector* with its i^{th} $(0 \leq i \leq n-1)$ bit as 1 if $XNOR$ rule is applied on the i^{th} cell. The two operations ($.$, $+$) follows the operations of Galois Field 2.

The LCA is a special case of ACA where the inversion vector F is an all $0's$ *vector*. As a result, the state transition *equations 3.1* gets simplified to

$$s_{t+1} = T \cdot s_t \quad that\ is, \quad s_{t+p} = T^p \cdot s_t \tag{3.2}$$

respectively.

The state transition diagrams of cellular automata have been characterized from its T matrix and its Inversion vector F. The state transition behavior of a GF(2) CA can be classified into two different categories - Group and Non Group.

3.1.3 Group and Non-group CA

Group CA and Cycle Structure : For a Group CA (*Fig 3.2*), each state has a unique predecessor. That is, all states lie on a disjoint set of cycles. The entire state transition

diagram consists of a set of cycles (referred to as Cycle Structure) represented by

$$CS = [\mu_{k_1}(k_1), \mu_{k_2}(k_2), \cdots \mu_{k_m}(k_m)] \tag{3.3}$$

where k_i is the cycle length and μ_{k_i} is the number of times a component with cycle length k_i has occurred and is referred as cyclic component. The *Fig 3.2* illustrates the cycle structure of a 5 cell GF(2) Group CA. It has three cycles of length 3, 7 & 21 respectively, besides the all zero state forming a self-loop.

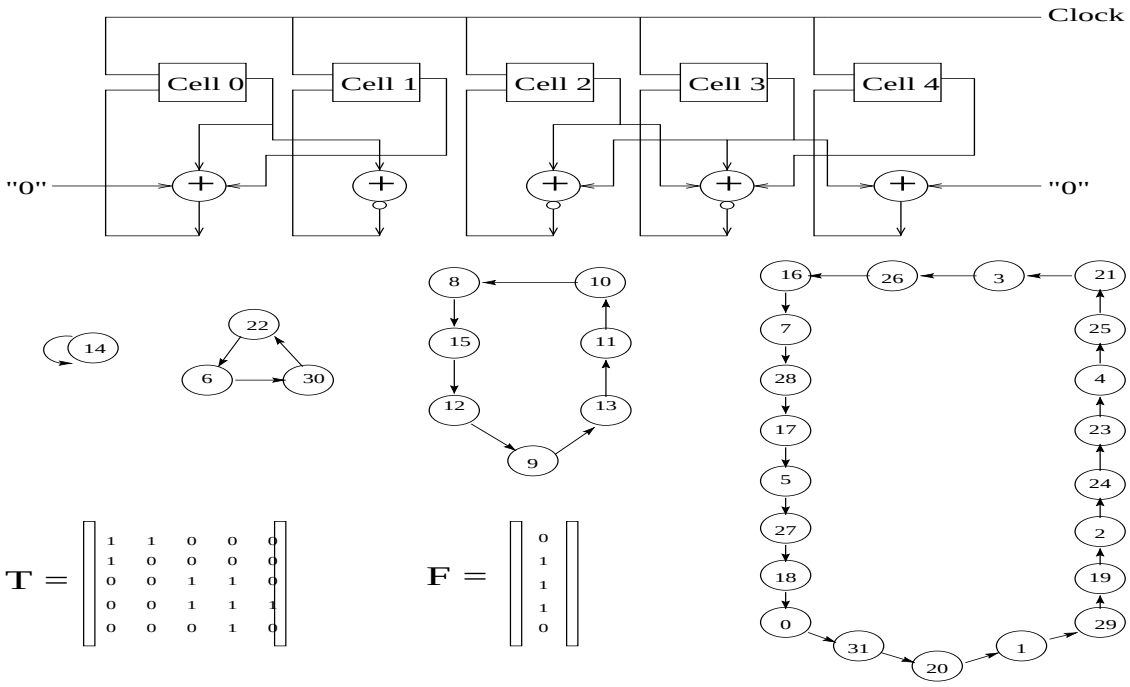


Figure 3.2: A 5-cell GF(2) Group CA along with its Characteristic Matrix(T) and inversion vector(F) with cycle structure [1(1),1(3),1(7),1(21)]

Non-group CA and Depth(d) : In the case of non-group CA, all its states do not lie on a cycle. There are cyclic states along with non-cyclic (also referred to as transient) states. The maximum run of the transient states is termed as depth (d) of the CA. Thus the depth d can be viewed as the number of clock cycles necessary for a CA to reach a cyclic state starting from a non-reachable state. The state transition diagram of a non-group CA has a Cycle Structure consisting of a set of cycles similar to Group CA. In addition, it has non-cyclic subspace represented by run of transient states forming an inverted tree rooted on a cyclic state For the 4 cell GF(2) CA (*Fig 3.3*), the depth $d = 2$ with cycle structure of [1(1),1(3)] - that is, one cycle of length 1 & another cycle of length 3.

[Note : From the results noted in [9], it can be shown that each inverted tree of a non-group CA has identical structure. So for characterizing GF(2) CA, we have concentrated on identifying the (i)cycle structure and (ii)depth of the inverted tree rooted on a cyclic state.]

The classification of group and non-group CA can be done by enumerating the determinant of the Linear operator T.

$$\begin{aligned} If \;\; det(T) &= 1, \text{ the } CA \text{ is a Group } CA \\ &= 0, \text{ the } CA \text{ is a Non Group } CA \end{aligned}$$

Besides the characteristic Matrix(T), a more detailed analysis of the state transition be-

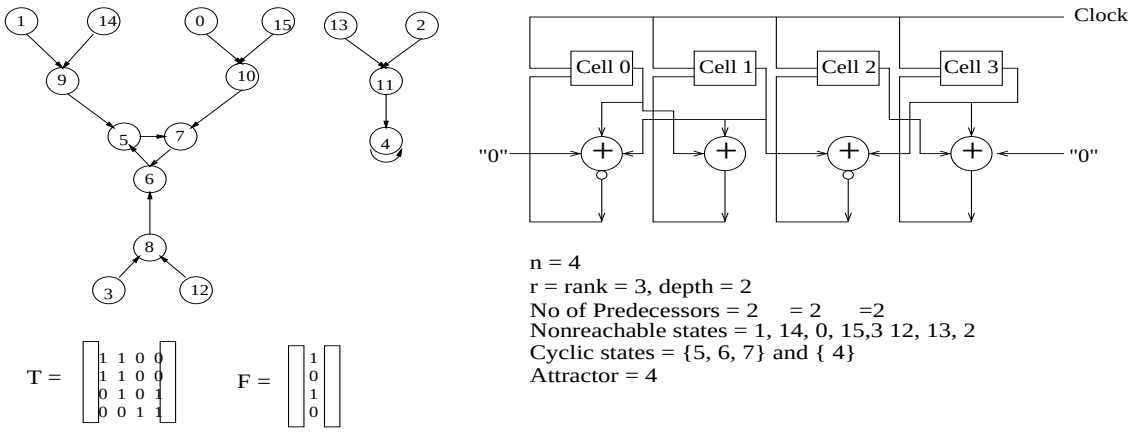


Figure 3.3: State Transition Diagram of a 4-cell non-group GF(2) CA along with its Characteristic Matrix(T) and inversion vector(F).

havior is done with the help of Characteristic Polynomial & Elementary Divisor.

3.1.4 Characteristic polynomial and Elementary Divisor :

Both characteristic polynomial and its Elementary Divisors play a key role in enumerating the cycle structure & depth of a CA.

Characteristic Polynomial: The characteristic polynomial $f(x)$ of a CA is derived from its characteristic matrix T by calculating $\det(T + Ix)$.

Minimal Polynomial: The minimal polynomial is the minimum degree polynomial which annihilates T.

Elementary Divisors: The characteristic polynomial $f(x)$ comprises of invariant polynomials $\phi_i(x)^{n_i}$ where $\phi_i(x)$ is irreducible. The polynomials $\phi_i(x)^{n_i}$, invariant to the linear operator T, are referred to as elementary divisors. Each elementary divisor, as outlined in Section 3.2.2, generates vector subspaces.

The characteristic polynomial $\phi(x)$ can be expressed as a product of its elementary divisors

$$\phi(x) = \phi_1(x)^{n_1}\phi_2(x)^{n_2}\cdots\phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}} \tag{3.4}$$

where $\phi_i(x)$ is irreducible (i = 1,2,$\cdots$,$\mathcal{N}$) [Note : In all subsequent discussions, the number of elementary divisors comprising a characteristic polynomial is denoted by $\mathcal{N}$.]

The characteristic polynomial of a non group CA takes the form

$$\phi(x) = x^{n_1} \cdot x^{n_2} \cdot \cdot x^{n_l} \phi_{l+1}(x)^{n_{l+1}} \cdot \cdot \phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}} \tag{3.5}$$

where a factor x^{n_i} $(i = 1,2,\cdot\cdot\cdot,l)$ $(l \leq \mathcal{N})$ represents a non-cyclic subspace. The depth of the CA corresponds to the largest power of x. Thus, from the *relation 3.5*,

$$depth(d) = (max(n_i))_{i=1}^{l} \tag{3.6}$$

The detailed characterization of CA state transition behavior are reported in [17, 12, 35, 37, 7, 9]. Some fundamental results along with some new framework for analysis of linear CA are presented below. The next section concentrates on building the theoretical framework for analysis of LCA. Based upon the theoretical framework of LCA, we build the complete characterization of Additive CA in *Section 3.3*.

3.2 Vector Space Theoretic Analysis of Linear CA (LCA)

The analysis reported in this section characterizes a Linear CA in terms of the cyclic and non-cyclic sub-spaces it generates. The methodology of analysis is as follow - we derive the characteristic polynomial in elementary divisor form and analyze each individual elementary divisor comprising the characteristic polynomial of the LCA. Consequently, we device a methodology to aggregate each individual result and accordingly derive the final result.

The major steps required to enumerate cycle structure/depth of an LCA are noted below.

Algorithm 3.1 *Enum_CS_Depth_from_LCA(T, CS, d)*
Input : T matrix.
Output : Cycle Structure (CS), depth (d).
Step 1: *Given a T matrix, generate the characteristic matrix in elementary divisor form and generate depth of the machine by evaluating max(n_i), where n_i is the power of $\phi_i(x)_{i=1}^{l}$ and $\forall_l \;\; \phi_i(x) = x$.*
Step 2: *Generate cycle structure of each elementary divisor $\phi_i(x)^{n_i}$, where $\phi_i(x) \neq x$.*
Step 3: *Output the final cycle structure by aggregating cycle structure of each individual elementary divisor.*

The next example illustrates the execution steps of *Algorithm 3.1*

Example 3.1 *Let the T matrix of a 7 cell GF(2) LCA be*

$$T = \begin{pmatrix} \lceil 0 & 0 \rceil & 0 & 0 & 0 & 0 & 0 \\ \lfloor 1 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & [1] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lceil 1 & 1 \rceil & 0 & 0 \\ 0 & 0 & 0 & \lfloor 0 & 1 \rfloor & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lceil 0 & 1 \rceil \\ 0 & 0 & 0 & 0 & 0 & \lfloor 1 & 1 \rfloor \end{pmatrix}$$

Step 1 : *The characteristic polynomial, as per Step 1 of* Algorithm 3.1, *in elementary divisor*

form is $x^2(x+1)(x+1)^2(x^2+x+1)$.
From the factor x^2 *we can directly calculate the depth of the* $LCA = 2$.
Step 2 : *Cycle Structure generated by each individual elementary divisor are*

$$CS_{LCA(x+1)} = [1(1), 1(1)], \quad CS_{LCA(x+1)^2} = [1(1), 1(1), 1(2)], \quad CS_{LCA(x^2+x+1)} = [1(1), 1(3)]$$

respectively.
Step 3 : *The complete cycle structure is enumerated by aggregating the effect obtained from each individual elementary divisor. Hence*

$$CS_{LCA} = [1(1), 1(1)] \times [1(1), 1(1), 1(2)] \times [1(1), 1(3)] = [4(1), 2(2), 4(3), 2(6)].$$

where $\times$ *is the aggregation operator.*

We next elaborate the detailed execution of each steps.

3.2.1 Deriving Elementary Divisors From T Matrix (STEP I)

The task of deriving elementary divisor from T matrix is achieved by converting the matrix $(T + Ix)$ in *Smith Normal Form*[23].

Definition 3.7 Smith Normal Form *is a diagonal form of matrix* $(T + Ix)$*, with nonzero elements* $a_1(x)$, $a_2(x)$, $\cdots$ $a_{\mathcal{L}}(x)$ $\cdots$ $a_n(x)$ *of* $GF(2)[x]$ *with degrees at least one in the diagonal. The elements* $a_{\mathcal{L}}(x)$ *are so arranged that* $a_{\mathcal{L}}(x)$ *is a factor of* $a_{\mathcal{L}+1}(x)$*. The elements* $a_1(x)$,...,$a_n(x)$ *are the invariant factors of* T *and* $a_n(x)$ *is the minimal polynomial of* T.

Since each $a_{\mathcal{L}}(x)$ in Smith Normal Form is an invariant polynomial, the irreducible factors of $a_{\mathcal{L}}(x)$ can be denoted as $\phi_l(x)^{n_l}, \phi_{l+1}(x)^{n_l+1}, ..., \phi_{l+m}(x)^{n_l+m}$, where each $\phi_{l+i}(x)$ is an irreducible polynomial and $\phi_{l+i}(x)^{n_{l+i}}$ is also invariant to T. Each $\phi_{l+i}(x)^{n_{l+i}}$ is an elementary divisor of T.

The characteristic polynomial can be expressed in *Elementary Divisor Form (EDF)* as

$$f(x) = \phi_1(x)^{n_1}\phi_2(x)^{n_2} \cdots \phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}} \tag{3.7}$$

where each $\phi_i(x)$ is an irreducible polynomial and factor of any $a_{\mathcal{L}}(x)$.

Example 3.2 *The following example illustrates conversion of a matrix to its Smith Normal Form by executing the following three elementary operations on the matrix* T *of* Example 3.1
(a) interchanging two rows or columns.
(b) adding a multiple (in GF(2)[x]) *of one row or column to another.*
(c) multiplying any row or column by a non-zero element in $GF(2)$.

The Smith Normal Form(SNF) of matrix T is

$$\text{SNF} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1+x & 0 \\ 0 & 0 & 0 & 0 & (1+x)^2(x^2+x+1) \end{pmatrix}$$

The characteristic polynomial $f(x)$ *in EDF* $= (1+x)(1+x)^2(x^2+x+1)$
The minimal polynomial $f_{min}(x) = (1+x)^2(x^2+x+1)$.

[Note : $a_1(x) = a_2(x) = a_3(x) = 1$, $a_4(x) = (1+x)$ and $a_5(x) = (1+x)^2(x^2+x+1)$.]

We next present the theorem reported in [23] which establishes the feasibility of conversion of any matrix T into *Smith Normal Form.*

Theorem 3.1 *Let T be an $n \times n$ matrix over the field GF(2). Using the three elementary row and column operations (a,b and c noted above), the $n \times n$ matrix $(T+Ix)$ with entries from* GF(2^p)[x] *can be expressed in* Smith Normal Form.

Proof : Proof available in [23]. □

The complete algorithm for *Step 1* is presented next.

Algorithm 3.2 *Enum_ED_from_T(T, $[\phi_i(x)^{n_i}]$, d)*
Input : Characteristic Matrix T
Output : $[\phi_i(x)^{n_i}]$: Elementary Divisor(ED) set : $\phi(x) \neq (x+1)$, d: depth;

(a) Convert $T + Ix$ into Smith Normal Form.

(b) Represent the characteristic polynomial in the Elementary Divisor Form.

$$f(x) = \phi_1(x)^{n_1}\phi_2(x)^{n_2}\cdots\phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}} \tag{3.8}$$

where each $\phi_i(x)$ is an irreducible polynomial and degree of $\phi_i(x) \geq \phi_{i-1}(x)$.

(c) For each $\phi_i(x)^{n_i}$, where $\phi_i(x) = (x+1)$ $\forall_{i=1}^{l}$
Enumerate $d = max(n_i)$ $\forall_{i=1}^{l}$.

(d) Form the elementary divisor set $[\phi_i(x)^{n_i}]$ with $\phi_{l+1}(x)^{n_{l+1}} \cdot \phi_{l+2}(x)^{n_{l+2}} \cdots \phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}}$.

3.2.2 Generation of Cyclic Sub-space by each individual Elementary Divisor(Step II)

An *Elementary Divisor* $\phi_i(x)^{n_i}$ produces two different cyclic sub-spaces referred to as Primary and Secondary periods [9]. A Primary period is the cycle structure generated by the factor $\phi_i(x)$ while the Secondary Period is the cycle structure formed by the factors $\phi_i(x)^t$ $\forall_{t=2}^{n_i}$. Thus an elementary divisor $\phi_i(x)^{n_i}$ generates the cycles corresponding to the factor $\phi_i(x)$ and corresponding each of the factors – $\{\phi_i(x)\}^2, \cdots \{\phi_i(x)\}^{n_i}$.

The complete cycle structure is the union of the cycle structures of these factors. *Enumeration of Primary Cycle Structure for $\phi_i(x)$* - It consists of μ_k cycles of length k (denoted

as $[\mu_k(k)]$), where

$$\mu_k = \frac{2^{r_i} - 1}{k} \tag{3.9}$$

r_i being the degree of $\phi_i(x)$, and k is the smallest integer such that $\phi_i(x)$ divides $x^k + 1$.

Enumeration of Secondary Cycle Structure :
The enumeration of secondary cycle structure consists of enumerating cycle length and cyclic components respectively.

Enumeration of Secondary Cycle Length : The secondary cycle length is of the form $2^j \cdot k$ where j is an integer. *Lemma 3.2* establishes the relationship between factor of a polynomial and the length of the cycle it generates.

Lemma 3.2 *All the factors of an elementary divisor $\phi_i(x)^{n_i}$, starting from $\phi_i(x)^{2^{j-1}+1}$ to $\phi_i(x)^{2^j}$, contribute the same cycle length $(2^j \cdot k)$, where k is the primary cycle length.*

Example 3.3 *Let an elementary divisor be $(x^2+x+1)^9$. The primary cycle length is $1(3)$. Then all the factors $(x^2+x+1)^5$, $(x^2+x+1)^6$, $(x^2+x+1)^7$ and $(x^2+x+1)^8$ form cycles of length $2^3 \cdot 3 = 24$.*

Enumeration of Secondary Cyclic Component : In the above framework to enumerate cyclic component corresponding to each secondary cycle length, we introduce the term j^{th} factor set.

Definition 3.8 *A factor of $\phi_i(x)^{n_i}$ whose power lies between $2^{j-1}+1$ to 2^j is referred to as a member of j^{th} factor set. The cardinality of the j^{th} factor set in an elementary divisor $\phi_i(x)^{n_i}$ is denoted as $N(F_j)$ and can be determined by the formula*

$$N(F_j) = \begin{cases} 2^{j-1}, & \text{if } n_i > 2^j \\ n_i - 2^{j-1} & \text{if } 2^{j-1} < n_i \leq 2^j \\ 0, & \text{if } n_i \leq 2^{j-1}. \\ 1, & \text{if j = 0 \& } n_i \geq 1 \end{cases} \tag{3.10}$$

[Note : Each member of the j^{th} factor set produces a cycle of length $(2^j \cdot k)$, where k is the primary cycle of the polynomial & the maximum value of $j = \rceil log_2(n_i) \lfloor$ and the maximum value of $j = \lceil log_2(n_i) \rceil$.]

The following theorem ensures the enumeration of cyclic component of cycle length $2^j \cdot k$.

Theorem 3.3 *The cyclic component corresponding to the cycle length $2^j \cdot k$, is given by the following relation*

$$\mu_{2^j \cdot k} = \frac{2^r \sum_{\mathcal{J}=0}^{j} N(F_j) - 2^r \sum_{\mathcal{J}=0}^{j-1} N(F_j)}{2^j \cdot k} \tag{3.11}$$

The union of the cycle structures of all the factors yield the complete cycle structure of LCA. It is represented as $CS_{LCA(\phi_i(x)^{n_i})}$

$$CS_{LCA(\phi_i(x)^{n_i})} = [1(1) + \sum_{j=0}^{m_k} \mu_{2^j \cdot k}(2^j \cdot k)] \tag{3.12}$$

where $m_k = \lceil log_2(n_i) \rceil$, 1(1) represents the all-zero vector forming a self-loop.
We introduce a new terminology and a new method of representing $CS_{LCA(\phi_i(x)^{n_i}}$.

Definition 3.9 *A Primary Cycle Structure (PCS) represents the cycle structure generated by an elementary divisor $\phi_i(x)^{n_i}$. In addition to the representation by* Relation 3.12, *it is also denoted by a triplet $(\mu_k, k)^{n_i}$, where [1(1), $\mu_k(k)$] is the cycle structure of the irreducible polynomial $\phi_i(x)$, that is,*

- *μ_k is the cyclic component corresponding to the irreducible polynomial $\phi_i(x)$*
- *k is the cycle length of the irreducible polynomial $\phi_i(x)$*
- *μ_k and k are also referred to as primary cyclic component and primary cycle length of the elementary divisor respectively*
- *n_i is the power to which the irreducible polynomial $\phi_i(x)$ is raised in the process of its formation of elementary divisor.*

This form of representation imparts an identical mapping with the elementary divisor. Henceforth, the terms factors of an Elementary Divisor and factors of PCS are used synonymously.

Definition 3.10 *Primary Cycle Family : All the cycles of the form $(2^j \cdot k)$, where k is a primary cycle, are the members of the family of cycles referred to as Primary Cycle Family. It is also referred as k-Cycle Family.*

We now state the generation of cycle structure in algorithmic form and subsequently illustrate it with an example.

Algorithm 3.3 *Enum_PCS_from_ED ($\phi_(x)^{n_i}$, PCS)*

Input : Elementary Divisor(ED) : $(\phi_i(x)^{n_i})$
Output : Primary Cycle Structure(PCS).
Primary Cycle Enumeration : *Find the least value of k such that $\phi(x)$ divides $x^k + 1$.*
Secondary Cycle Enumeration :
Enumerate the j^{th} factor set for $j = 0$ to $\lceil log_2(n_i) \rceil$
For $j = 0$ to $\lceil log_2(n_i) \rceil$
    Enumerate $\mu_{2^j \cdot k}$ through relation 3.11 and
    Accordingly obtain $\mu_{2^j \cdot k}(2^j \cdot k)$.
Output the Final Primary Cycle Structure (PCS) by union of all secondary cycles.

Example 3.4 *The characteristic polynomial $f(x)$ of an example LCA is given by $f(x) = (x^2 + x + 1)^9$.*
Therefore, degree $r_i = 2$
Primary Cycle:
Since $(x^2 + x + 1)$ divides $(x^3 + 1)$, $k = 3$. The $\mu_k = \frac{2 \cdot 1 - 1}{3} = 3$. Therefore, the cycle structure contributed by $(x^2 + x + 1)$ is 1(3)
Secondary Cycle :
The j^{th} Factor Sets are $\mathrm{N}(F_0) = 1$, $\mathrm{N}(F_1) = 1$, $\mathrm{N}(F_2) = 2$, $\mathrm{N}(F_3) = 4$, $N(F_4) = 1$.
Cyclic Component corresponding to $\mu_{2^1 \cdot k} = \frac{2^{2(1+1)} - 2^{2(1)}}{2^1 \cdot 3} = 2$.
Cyclic Component corresponding to $\mu_{2^2 \cdot k} = \frac{2^{2(2+2)} - 2^{2(2)}}{2^2 \cdot 3} = 20$.
Cyclic Component corresponding to $\mu_{2^3 \cdot k} = \frac{2^{2(4+4)} - 2^{2(4)}}{2^3 \cdot 3} = 2720$.
cyclic Component corresponding to $\mu_{2^4 \cdot k} = \frac{2^{2(1+8)} - 2^{2(8)}}{2^4 \cdot 3} = 4096$.
Hence the complete cycle structure $CS_{LCA(x^2+x+1)^9}$ = [1(1), 1(3), 2(6), 20(12), 2720(24), 4096(48)].
The Primary Cycle Structure can also be represented by the triplet $(1, 3)^9$ since as per Definition 3.9, *$\mu_k = 1$, $k = 3$, $n_i = 9$.*

The following lemmas provide some more insight into the cycle length of an LCA. The lemmas provide the foundation for executing *Step 3* of *Algorithm 3.1*, as well as the efficient synthesis of LCA reported in *Chapter 4*.

Lemma 3.4 *Cycle length of primary period is always odd.*

Proof : From *Primary Cycle Enumeration* primary cycle structure $[1, \mu_k(k)]$ follows the relation

$$\mu_k \times k = 2^{r_k} - 1 \tag{3.13}$$

Since 2^{r_k} - 1 is odd, both the factors are necessarily odd. □

Lemma 3.5 *A secondary period corresponds to an unique primary period.*

Proof : Cycle length of a secondary period is expressed in the form of $2^j \cdot k$ for $j \geq 1$ and where k is a primary cycle and hence odd. Thus for two primary period k_1 and k_2 to have same secondary period the relation $2^{j_1} \cdot k_1 = 2^{j_2} \cdot k_2$ must hold, which is possible only when $k_1 = k_2$. □

Characterization of vector spaces generated by each elementary divisor of an LCA enables the characterization of the LCA having multiple elementary divisors. The next subsection provides detail analysis of such an LCA

3.2.3 Vector Space of an LCA having Multiple Elementary Divisors

The entire vector space S produced by a LCA is the direct sum of the sub-spaces generated by its elementary divisors. The cycle structure of the LCA is obtained from cross product $(\times)$[50], as defined below, of all the Primary Cycle Structures (*Definition 3.9*).

Definition 3.11 *Cross Product ($\times$) of two cycle structures CS_1 and CS_2, where*

$$CS_1 = [1(1) + \sum_{i_1=1}^{m_{k_{i_1}}} \mu_{k_{1i_1}}(k_{1i_1})] \quad and \quad CS_2 = [1(1) + \sum_{i_2=1}^{m_{k_{i_2}}} \mu_{k_{1i_2}}(k_{1i_2})]$$

is the product of each ${i_1}^{th}$ term of CS_1 with ${i_2}^{th}$ term of CS_2.
The product of $\mu_{k_{1i_1}}(k_{1i_1})$ and $\mu_{k_{2i_2}}(k_{2i_2})$ results in cyclic component μ_k of length k following the equations [50]

$$\mu_k = \mu_{k_{1i_1}}.\mu_{k_{2i_2}}.gcd(k_{1i_1}, k_{2i_2}) \quad and \quad k = lcm(k_{1i_1}, k_{2i_2}) \tag{3.14}$$

The complete cycle structure of LCA, the cross-product of cycle structures of individual elementary divisors, is then expressed through the following equation.

$$CS_{LCA(f(x))} = CS_{LCA(\phi_1(x)^{n_1})} \times \cdots CS_{LCA(\phi_i(x)^{n_i})} \times \cdots \times CS_{LCA(\phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}})} \tag{3.15}$$

To efficiently utilize the concept of cross product, the algorithm for evaluation of the final cycle is divided in two steps.
Step A : We evaluate the cross product of PCS having same primary cycle and form a Primary Family Cycle Structure(PFCS).
Step B : We then successively evaluate the cross product each of the obtained $PFCS$, starting from the product of the $PFCS$ which has the largest primary cycle.

But before explaining each of the steps, we characterize the nature of the final cycle structure of the LCA.

Theorem 3.6 *The cycle structure $(CS(\mathcal{N}))$ of an LCA with $\mathcal{N}$ elementary divisors can be represented as*

$$CS = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)] \tag{3.16}$$

where k_i is odd.

Proof : Let $\alpha_i = \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)$. Then $CS(\mathcal{N}) = [1, \alpha_1] \times [1, \alpha_2] \cdots [1, \alpha_{\mathcal{N}}]$
On performing successive cross product, as per *relation 3.15*

$$CS(\mathcal{N}) = [1(1), (\alpha_1, \alpha_2 \cdots, \alpha_{\mathcal{N}}), (\alpha_1 \times \alpha_2, \cdots \alpha_{\mathcal{N}-1} \times \alpha_{\mathcal{N}}), \cdots, (\alpha_1 \times \alpha_2 \times \cdots \times \alpha_{\mathcal{N}})]$$

That is, each member in the i^{th} group (marked within '()') is formed by cross producing i number of α's that forms a component of CS($\mathcal{N}$).
Each component, as noted below for $i = 2$, represents a primary cycle family. Generalization for $i > 2$ follows directly.

Let $\tilde{CS} = (\alpha_1 \times \alpha_2)$ and let [CL] be the set of Cycle Lengths formed from the above cross product.
Then the general nature of [CL] set is expressed by the following relation.

$$[CL] = lcm((2^{j_1} \cdot k_1), (2^{j_2} \cdot k_2)) = 2^{max(j_1,j_2)} lcm(k_1, k_2). \tag{3.17}$$

Let $lcm(k_1, k_2) = k_3$. Consequently, all cycle lengths of the [CL] set is a member of k_3 cycle family and the cycle structure can be expressed in the form

$$\tilde{CS} = \sum_{j=0}^{max(m_{k_1}, m_{k_2})} \mu_{2^j \cdot k_3}(2^j \cdot k_3)$$

Since both k_1, k_2 are primary cycles and odd, therefore k_3 is also odd. Hence the proof. □

As it is obvious from the above theorem, that the cycle structure of a LCA follows a certain symmetry. In this connection, we define the term *Legal Cycle Structure.*

Definition 3.12 Legal Linear Cycle Structure *is a cycle structure which satisfies the criteria defined in* relation 3.16 *and which can be broken into Primary Cycle Structures.*

With the results of the above theorem in the background we proceed to explain each of the steps - *Step A* and *Step B.*

Step A : Final Cycle Structure of set of Elementary divisors with identical Primary Cycle Length

This step shows the final structure CS obtained from the following relation

$$CS = PCS_1 \times PCS_2 \times \cdots \times PCS_i \times \cdots PCS_{\mathcal{N}}$$

The final cycle structure obtained from cross-producting cycle structures of Elementary Divisor having identical primary cycle is termed as *Primary Family Cycle Structure (PFCS).* We defined $PFCS$ and develop the method of obtaining it.

Definition 3.13 Primary Family Cycle Structure (PFCS) *is a special class of cycle structure that results from cross product of Primary Cycle Structures, each having the same primary cycle length. That is, the primary cycle length is identical for all the underlying elementary divisors generating the $PFCS$. It is of the form -*

$$PFCS = [1(1) + \sum_{j=0}^{m_k} \mu_{2^j \cdot k}(2^j \cdot k)] \tag{3.18}$$

The following example illustrates the concept of *Primary Family Cycle Structure* ($PFCS$)

Example 3.5 *Let us consider the characteristic polynomial* $f(x) = (x^2+x+1)^1 \cdot (x^2+x+1)^4 \cdot (x^2+x+1)^7 \cdot (x^2+x+1)^8$ *represented in the elementary divisor form.*

The PCS are $(1,3)^1$, $(1,3)^4$, $(1,3)^7$, $(1,3)^8$ *respectively.*
The cycle structure $CS = (1,3)^3 \times (1,3)^2 \times (1,3)^1$ *that is, writing in expanded form*
CS = [1(1), 1(3)] × [1(1), 1(3), 2(6), 20(12)] × [1(1), 1(3), 2(6), 20(12), 672(24)] × [1(1), 1(3), 2(6), 20(12), 2720(24)]
Consequently, CS = [1(1),85(3), 3688(6), 5591040(12), 4581088288(24)]
which shows CS *has a primary cycle of length 3. The* CS *is a* Primary Family Cycle Structure (PFCS).

The cycle structure of the $PFCS$ is obtained directly from the cardinality of factor set of each individual constituent elementary divisors, that is, Primary Cycle Structure(PCS), forming it.

Let a $PFCS$ be formed from characteristic polynomial $f(x)$ where $f(x) = \phi(x)^{n_1} \cdot \phi(x)^{n_2} \cdots \phi(x)^{n_{\mathcal{N}}}$ in elementary divisor form, which implies the $PFCS$ is formed from cross-product of the set of PCS and

$$PFCS = (\mu,k)^{n_1} \times (\mu,k)^{n_2} \times (\mu,k)^{n_{\mathcal{N}}}$$

each PCS $(\mu,k)^{n_i}$ corresponds to Elementary Divisor $\phi(x)^{n_i}$. $N(F_j)$ is the number of j^{th} component present in a single elementary divisor or PCS(*Definition 3.10*). From $N(F_j)$, we derive the definition of $\tilde{N}(F_j)$.

Definition 3.14 $\tilde{N}(F_j)$ *is the sum of the number of* j^{th} *factor set present in all the constituent PCS comprising the PFCS.*
[Note : j^{th} components are the factors $2^{j-1}+1$ to 2^j present in any PCS - $(\mu,k)^{n_i}$.]

Example 3.6 *Let a PFCS be formed from cross product of PCS such that*

$$PFCS = (1,3)^1 \times (1,3)^4 \times (1,3)^7 \times (1,3)^8$$

The underlying elementary divisors are $(x^2+x+1)^3$, $(x^2+x+1)^2$ *and* (x^2+x+1) *respectively.*
While computing $\tilde{N}(F_j)$ *for each value of* j, *it result in*
$\tilde{N}(F_0) = 4$ *(*0^{th} *component) - that is, the irreducible polynomial* (x^2+x+1) *must be present in all the four PCS.*
$\tilde{N}(F_1) = 3$; *the* 1^{st} *component. That is, the factors starting from* $2^{1-1}+1$ *to* 2^1 *are present in the last three PCS.* $\tilde{N}(F_2) = 6$; *the* 2^{nd} *component - that is, the factors starting from* $2^{2-1}+1$ *to* 2^2 *are present only in the last three PCS.*
$\tilde{N}(F_3) = 7$; *the* 3^{rd} *component - that is, the factors starting from* $2^{3-1}+1$ *to* 2^3 *are present only in the last two PCS, four in the last one and three in the* 2^{nd} *last state.*
The highest value of j *is 2 as the highest value of* j *among all the individual components is 2.*

The algorithm to enumerate $\tilde{N}(F_j)$ from $\mathcal{N}$ PCS comprising the $PFCS$ is reported below.

Algorithm 3.4 *Enum_$\tilde{N}(F_j)$_from_PCS*([*PCS*], [$\tilde{N}(F_j)$])
Input : $\mathcal{N}$ *set of PCS*
Output : $[\tilde{N}(F_j)]$ *:* j *= 0, 1, 2,* $\cdots$
For j *= 0 to* $max(\lceil log_2(max(n_i)_{i=1}^{\mathcal{N}})\rceil)$
{
$\tilde{N}(F_j)$ *= 0*
For i *= 1 to* $\mathcal{N}$
{
For each PCS_i *calculate* $N(F_j)$ *by relation 3.10*
$\tilde{N}(F_j)$ *=* $\tilde{N}(F_j)$ *+* $N_i(F_j)$ */* $N_i(F_j)$ - the number of* j^{th} *component in the* i^{th} *PCS */*
}
}

Based upon the concept of $\tilde{N}(F_j)$, we formulate the theorem for calculating the cycle structure of $PFCS$

Theorem 3.7 : *The cycle structure of* $PFCS$ *is of the form* $[1(1), \mu_k(k), \cdots \mu_2^{j\cdot k}(2^j \cdot k)\cdots]$ *where the value of the cyclic component* $\mu_{2^j\cdot k}$ *follows the relation*

$$\mu_{2^j\cdot k} = \frac{2^r \sum_{\mathcal{J}=0}^{j} \tilde{N}(F_{\mathcal{J}}) - 2^r \sum_{\mathcal{J}=0}^{j-1} \tilde{N}(F_{\mathcal{J}})}{2^j \cdot k} \tag{3.19}$$

where $\tilde{N}(F_{\mathcal{J}})$ *represents the sum of the number of* $\mathcal{J}^{th}$ *factor set present in all the constituent* PCS.

Proof : A cycle of length $(2^j \cdot k)$ can be formed only by cross product of $(2^j \cdot k)$ & $(2^{\mathcal{J}} \cdot k)$ where $(\mathcal{J} \leq j)$ (*relation 3.17*). Therefore, the value of cyclic component $\mu_{2^j\cdot k}$ depends only upon the number of $(\mathcal{J} \leq j)^{th}$ factor set. Since this is a iterative relation, the extra states produced with the introduction of the j^{th} factor set is responsible for the value of cyclic component $\mu_{2^j\cdot k}$.

The extra set of states S generated in the process is given by

$$S = 2^r \sum_{\mathcal{J}=0}^{j} \tilde{N}(F_{\mathcal{J}}) - 2^r \sum_{\mathcal{J}=0}^{j-1} \tilde{N}(F_{\mathcal{J}})$$

Hence $\mu_{2^j\cdot k} = \frac{S}{2^j\cdot k}$, where $2^j \cdot k$ is the cycle length corresponding to the component $\mu_{2^j\cdot k}$.
□

Example 3.7 *Let a* $PFCS$ *be formed from cross product of* PCS *such that*

$$PFCS = (1,3)^1 \times (1,3)^4 \times (1,3)^7 \times (1,3)^8$$

8 is the highest power of any constituent PCS*, in this case, the last* PCS.

Therefore, for the values 0, 1 and 2, the $\mathcal{J}^{th}$ factor set is present in the characteristic polynomial.
The cyclic components

$$\mu_{2^0 \cdot 3} = \frac{2^{2(4)} - 2^{2 \cdot 0}}{2^0 \cdot 3} = 85, \quad \mu_{2^1 \cdot 3} = \frac{2^{2(4+3)} - 2^{2 \cdot 4}}{2^1 \cdot 3} = 3688$$

$$\mu_{2^2 \cdot 3} = \frac{2^{2(7+6)} - 2^{2 \cdot 7}}{2^2 \cdot 3} = 5591040 \quad \mu_{2^3 \cdot 3} = \frac{2^{2(13+7)} - 2^{2 \cdot 13}}{2^2 \cdot 3} = 4581088288$$

The complete cycle structure CS = [1(1),85(3), 3688(6), 5591040(12), 4581088288(24)]

The complete algorithm is now stated

Algorithm 3.5 *Enum_PFCS_from_PCS(* $[PCS]$*, PFCS)*
Input : $\mathcal{N}$ *constituent PCS of PFCS : [PCS]*
Output : Cycle Structure $(PFCS)$: [1(1), $\mu_k(k)$, $\cdots$, $\mu_{2^j \cdot k}(2^j \cdot k)$, $\cdots$]
*Enum_*$\tilde{N}(F_j)$*_from_PCS(*$[PCS]$, $[\tilde{N}(F_j)]$)
Calculate $\mu_{2^j \cdot k}$ *for each* j *by using relation 3.19*

The *Step B* where the methodology to obtain the final cycle structure from the set of $PFCS$ each having different primary cycle is discussed next.

Step B : Final Cycle Structure from set of $PFCS$ each having different primary cycle

This step shows the final structure CS obtained from successive cross product of the constituent $PFCS$, that is

$$CS = PFCS_1 \times PFCS_2 \times \cdots PFCS_i \times PFCS_{i+1} \cdots \times PFCS_{\mathcal{N}} \tag{3.20}$$

The methodology for successive cross product is discussed taking into consideration a dynamic situation $CS = CS_1 \times CS_2$. where

$$CS_1 = PFCS_i, \qquad CS_2 = PFCS_{i+1} \times \cdots \times PFCS_{\mathcal{N}}$$

with the assumption that the primary cycle (k_1) of CS_1 is smaller than all the primary cycle k_i in CS_2. Therefore, with the help of *relation 3.16* & *3.18*, we can write the above statement in notation form.

$$CS = CS_1 \times CS_2 \;\; where \;\; CS_1 = [1(1), \alpha] \;\; and \;\; CS_2 = [1(1), \sum_{k_i}, \beta_i]$$

and

$$\alpha = \sum_{j=0}^{m_{k_1}} \mu_{2^j \cdot k_1}(2^j \cdot k_1), \quad \beta_i = \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i), \quad i > 1$$

β_i represent each cycle family in CS_2. The next couple of theorems elaborate the effect CS_1 & CS_2 respectively produces on CS.

Theorem 3.8 *The cyclic components ($\mu_{2^j \cdot k_1}$) corresponding to k_1 cycle family will be same in CS and CS_1, where $CS = CS_1 \times CS_2$.*

Proof : By definition,

$$CS = [1(1), \alpha] \times [1(1), \sum_i \beta_i)] \tag{3.21}$$

That is,

$$CS = [1(1), \alpha, \sum \beta_i, \sum_i \alpha \times \beta_i] \tag{3.22}$$

In the above equation, the cycle structure α is generated directly from CS_1 while the cycle structure β is generated from CS_2, and it does not have any primary cycle of length k_1. The cross product of k_1 cycle family with each k_i cycle family results in a k_3 cycle family where $k_3 = \text{lcm}(k_1, k_i)$. Hence it is greater then k_1.

Therefore, the cyclic components of k_1 cycle family only comes from α and it is same in CS & CS_1.
Hence the proof. □

The following example illustrates generation of CS_1 from a given CS. [Example to be changed]

Example 3.8 *Given CS_1= [1(1), [3(1), 2(2)]] and CS_2 = [1(1), 1(3), 3(5), 3(15)]. CS formed from cross product of CS_1 and CS_2 is given by* [1(1), [3(1),2(2)], [4(3),2(6)], [12(5),6(10)], [12(15),6(30)]]. *We see that the cyclic component corresponding to primary cycle 1 is same in CS_1 & CS_2.*

Subsequent discussions formalize the behavior of CS_2 in relation to CS ($CS = CS_1 \times CS_2$). *Relation 3.22* shows that the cross product occurs between each k_i cycle family generated by CS_2 and k_1 cycle family generated by CS_1. Let us concentrate on a single member $k_2 \in k_i$, where $k_3 = \text{lcm}(k_1,k_2)$. The cycle family obtained from cross product of $\alpha \times \beta_2$ thus has primary cycle k_3.

The following theorem and corollary characterizes k_3.

Theorem 3.9 *The value of the cyclic component ($\mu_{2^j \cdot k_3}$) formed from the cross product of cycle component of k_1 & k_2-cycle family in CS_1 and CS_2 respectively is determined by the following relation.*

$$\mu_{2^j \cdot k_3} = [A + B] \tag{3.23}$$

where

$$A = \frac{\mu_{2^j \cdot k_2} \cdot S_{2^j \cdot k_1} \cdot gcd(k_1, k_2)}{k_1} \quad \& \quad B = \frac{\mu_{2^j \cdot k_1} \cdot S_{2^{j-1} \cdot k_2} \cdot gcd(k_1, k_2)}{k_2}$$

$S_{2^j \cdot k_i}$ means number of states covered by cycle length ($\leq 2^j \cdot k_i$), that is

$$S_{2^j \cdot k_i} = \sum_{\mathcal{J}=0}^{j} \mu_{2^{\mathcal{J}} \cdot k_i} \times (2^{\mathcal{J}} \cdot k_i) \tag{3.24}$$

Proof : As per the *relation 3.17*, the cross product of $2^{j_1} \cdot k_1$ and $2^{j_2} \cdot k_2$ results in cycle length $2^{max(j_1, j_2)}(k_3)$, where $k_3 = lcm(k_1, k_2)$. Therefore, to produce $2^j \cdot k_3$, either j_1 or j_2 has to be equal to j.

Enumeration of A : If $j_2 = j$, then all the cycles $2^{j_1} \cdot k_1$ where $j_1 \leq j$ when cross producted with $2^j \cdot k_2$ will produce $2^j \cdot k_3$. The cross product, as per *relation 3.14*, will produce Part A of the cyclic component of $\mu_{2^j \cdot k_3}$ given by the relation

$$\mu_{2^j \cdot k_3} = \mu_{2^j \cdot k_2} \cdot [\mu_{k_1} + \mu_{2 \cdot k_1} \times 2 \cdot k_1 \cdots \mu_{2^j \cdot k_1} \times 2^j \cdot k_1] \cdot gcd(k_1, k_2)$$

The equation can be rewritten as

$$\mu_{2^j \cdot k_3} = \mu_{2^j \cdot k_2} \cdot [\frac{\sum_{\mathcal{J}=0}^{j} \mu_{2^{\mathcal{J}} \cdot k_1} \times (2^{\mathcal{J}} \cdot k_1)}{k_1}] \cdot gcd(k_1, k_2) \tag{3.25}$$

With the help of *relation 3.24*, the equation can be simplified

$$\mu_{2^j \cdot k_3} = \frac{\mu_{2^j \cdot k_2} \cdot S_{2^j \cdot k_1}}{k_1} \cdot gcd(k_1, k_2) \tag{3.26}$$

Enumeration of B : Similarly, if $j_1 = j$, cross product with $2^{j_2} \cdot k_2$ $j_2 \leq j$ will produce $(2^j \cdot k_3)$. To avoid cross-product between $2^j \cdot k_1$ *and* $2^j \cdot k_2$ twice, j_2 ends at $(j-1)$ in the second case. Hence the proof. □

From the theorem the following corollary follows

Corollary 3.1 : *The cycle structure of the k_3-cycle family will be of the form*

$$CS(k_3) = \sum_{j=0}^{m_{k_3}} \mu_{2^j \cdot k_3}(2^j \cdot k_3), where \quad m_{k_3} = max(m_{k_1}, m_{k_2}) \tag{3.27}$$

where $max(m_{k_1}, m_{k_2})$ indicate the maximum between the highest power of either k_1 and k_2 cycle family.

The value of k_3 can be of two types 1. $lcm(k_1,k_2) = k_3 > k_2$. and 2. $lcm(k_1,k_2) = k_3 = k_2$.

- If $k_3 = k_2$, then both β_2 & $\alpha \times \beta_2$ belong to the same cycle family. $\tilde{\mu}_{2^j \cdot k_2}$ in CS will be represented by the relation

$$\tilde{\mu}_{2^j \cdot k_3}(CS) = \tilde{\mu}_{2^j \cdot k_2}(CS) = \mu_{2^j \cdot k_2}(\beta_2) + \mu_{2^j \cdot k_2}(\alpha \times \beta_2) \tag{3.28}$$

  Since lcm(k_1,k_2) = k_2, therefore gcd(k_1, k_2) = k_1. Combining *Relation 3.23 & 3.28*

$$\tilde{\mu}_{2^j \cdot k_3}(CS) = [\mu_{2^j \cdot k_2} \cdot (S_{2^j \cdot k_1} + 1) + \mu_{2^j \cdot k_1} \cdot S_{2^{j-1} \cdot k_2} \frac{k_1}{k_2}] \tag{3.29}$$

- If $(k_3 > lcm(k_1, k_2))$, then besides the cyclic component $\mu_{2^j \cdot k_2}$ of the k_2-cycle family from CS_2, the k_3-cycle family is also added to CS where the cyclic component

$$\tilde{\mu}_{2^j \cdot k_3}(CS) = \mu_{2^j \cdot k_3} \tag{3.30}$$

We now state the algorithm for generating CS from CS_1 & CS_2

Algorithm 3.6 *Enum_CS_from_CS$_1$&CS$_2$* (CS_1, CS_2, CS)
Input : $CS_1 = [1(1), \sum \mu_{2^j \cdot k_1}(2^j \cdot k_1)]$ $CS = [1(1), \sum_{k_i} \sum \mu_{2^j \cdot k_i}(2^j \cdot k_i)]$
$k_1 < k_i \ \forall i$.
Output : CS
for each $k_i \in CS_2$ *in ascending order*
{
$k_2 = k_i$
if $lcm(k_1, k_2) = k_2$
{
Obtain each $\tilde{\mu}_{2^j \cdot k_2}(CS)$ *from relation 3.29*
$CS \leftarrow CS + \tilde{\mu}_{2^j \cdot k_2}$
}
else
{
Obtain each $\tilde{\mu}_{2^j \cdot k_3}$ *from relation 3.30*
$CS \leftarrow CS + \tilde{\mu}_{2^j \cdot k_3}$
$CS \leftarrow CS + \mu_{2^j \cdot k_2}(CS_2)$
}
$CS = CS + CS_1$ *from Theorem 3.8*
}

Example 3.9 Case : lcm(k_1, k_2) > k_3.
Let CS_1 = [1(1), 1(3)] & CS_2 = [1(1), 3(5)]. *Therefore* k_1 = *3,* k_2 = *5. lcm(3, 5) = 15* $\neq$ *5.*

$\mu_{15} = \frac{\mu_3 \cdot S_3}{3} \cdot gcd(3,5) = \frac{3 \cdot 3}{3} = 3.$
Final CS = [1(1), 1(3), 3(5), 3(15)].
Case : lcm(k_1, k_2) = k_3.
Let CS_1 = 1(1), 3(1), 2(2)]; CS_2 = [1(1), 1(3), 3(5), 3(15)]
k_1 = *1. First* k_2 = *3.* $lcm(k_1, k_2) = K_2$.
Therefore, $\tilde{\mu}_3 = [\mu_3 \cdot (S_1 + 1)] = 1 \times 4 = 4.$
$\tilde{\mu}_6 = [\mu_6 \cdot (S_2 + 1) + \mu_2 \cdot S_3 \cdot \frac{1}{3}] = [0 + 2 \cdot 3 \cdot \frac{1}{3}] = 2.$
Similarly, $\tilde{\mu}_5 = 12$, $\tilde{\mu}_{10} = 6$
$\tilde{\mu}_{15} = 12$, $\tilde{\mu}_{30} = 6.$

In order to dynamically generate the final CS from each of the constituent $PFCS$ the *Algorithm 3.6* has to be called iteratively starting from the $PFCS$ which has the largest primary cycle. The algorithm is noted below.

Algorithm 3.7 *Enum_CS_from_PFCS*([PFCS], CS)
Input : Cycle Structure set $PFCS_1$, $PFCS_2 \cdots PFCS_{\tilde{N}}$: $[PFCS]$
Output : Final Cycle Structure CS.
Sort the $PFCS$ *in descending order of the size of the primary key.*
$CS_2 \leftarrow \phi$
while $i \leq \tilde{\mathcal{N}}$
{
$CS_1 \leftarrow PFCS$
$Enum\_CS\_From\_CS_1 \& CS_2(CS_1, CS_2, CS)$
$CS_2 = CS$
$i{+}{+}$
}

3.3 Vector Space Theoretic Analysis of Additive Cellular Automata(ACA)

Complete characterization of the state transition behavior of an ACA is undertaken in this section. An ACA C', as noted in *Section 3.1*, is represented by the characteristic matrix T and non-zero inversion vector F. The linear counterpart of the C' is the C which is represented by the same characteristic matrix T with all 0's F Vector. The state transition behavior of C' and C share some common properties that are reproduced below from [9].

- An C' is a group CA if and only if LCA is a group CA.
- The depth of a non-group C' is same as that of C

Since the depth of an ACA can be directly derived from that of the corresponding LCA, we simply concentrate on enumareating the cycle structure of ACA.

The cycle structure of C' and C maintains some relations that are investigated in this section. In the subsequent discussions, both C and $C^{'}$ are assumed to have the same T matrix with inversion vector F=0 and $F \neq 0$ respectively. During analysis, we use the important concept of null space. The term null space and its relation with cycle length is stated below [42].

Definition 3.15 *Null Space : The null space of a matrix(T) consists of all such vectors that are transformed to the all-zero vector when premultiplied by the matrix.*

Axiom 3.1 : Cardinality of Null Space and Cycle Length : *Suppose a LCA represented by T has a cycle length k, then the cardinality of the null space of ($T^k + I$) indicates number of states forming cycles of length k or sub-multiple of k.*

The analysis of ACA concentrates on the characterization of the following properties :
P_1 : The method of checking whether a cycle length (k) is present in ACA, that is, in the state transition behavior of ACA.
P_2 : The special class of ACA - C' whose cycle structure is always same as that of C irrespective of its inversion vector F.
P_3 : The class of C' whose cycle structure differ from that of C. The property of F vectors which impart this difference and the nature of difference.
P_4 : Finally, the method to enumerate cycle structure and depth of C'.

Each of P_i (i= 1 to 4) has been detailed in subsequent discussions.

3.3.1 P_1 : Method to determine presence of cycle of length(k) in ACA

We present a theorem which enables us to determine whether a cycle of length(k) exists in ACA or not.

Theorem 3.10 *In the ACA characterized by T and the inversion vector F, there will exist a cycle of length k if*

$$rank([T^k + I]) = rank([T^k + I, \mathcal{F}]) \tag{3.31}$$

where $\mathcal{F} = [I + T + T^2 + \cdots + T^{k-1}]F$

Proof : Let x be a state in a cycle of length k in C'. Hence, as per the *relation 3.1* in *Section 3.1*,

$$x = [I + T + T^2 + \cdots + T^{k-1}]F + T^k x \tag{3.32}$$

We can rewrite the equation as

$$[T^k + I]x = \mathcal{F} \tag{3.33}$$

where

$$\mathcal{F} = [I + T + T^2 + \cdots + T^{k-1}]F \tag{3.34}$$

If a cycle of length k is to exist in the C', then the *equation 3.33* should be consistent. The condition for consistency is that

$$rank([T^k + I]) = rank([T^k + I, \mathcal{F}]) \tag{3.35}$$

Hence the proof. □

3.3.2 P_2 : Special Class of C' for which cycle structure is identical to that of C irrespective of F

The *Theorem 3.10* is utilized to explore a special class of C' which has cycle structure identical to that of C irrespective of its inversion vector F. The following theorem formally identifies the class.

Theorem 3.11 *The cycle structure of C' and C are identical if the characteristic polynomial $f(x)$ of the T matrix doesn't have a factor $(x + 1)$.*

Proof : Let k be the length of a cycle in the ACA - C. For a cycle of length k to exist in corresponding ACA - C', the condition of *relation 3.35* has to be satisfied.
The number of vectors forming cycle of length k or sub-multiple of k in the ACA/LCA are derived from enumeration of the Null Space of $\alpha_1 = (T^k + I)$[42].
The solution vectors are also roots of the equation $\phi_c(x) = 0$, where $\phi_c(x)$ is the largest factor of the characteristic polynomial $f(x)$ which divides $(x^k+1) = (x+1)(x^{k-1}+\cdot\cdot+x+1)$. So the solution space, as derived for α_1, is also obtained from the Null Space of $\alpha_2 = \phi_c(T)$.
Since $f(x)$ doesn't have a factor $(x + 1)$, each $\phi_c(x)$ divides $[1 + x + x^2 + \cdots + x^{k-1}]$
Therefore, the Null Space of $\alpha_3 = [I + T + T^2 \cdots T^{k-1}]$ also produces the same solutions. Hence,

$$Rank(T^k + I) = Rank(T^k + I, I + T + T^2 \cdots T^{k-1}) \tag{3.36}$$

Therefore, the next relation

$$Rank(T^k + I) = Rank(T^k + I, \mathcal{F})$$

directly follows for any F.

Therefore, all the cycle lengths of LCA -C also exist in ACA - C'. Since the number of vectors forming each cycle length is same for both, (directly derived from cardinality of Null Space) the cycle structures for both is identical. Hence the proof. □

Example 3.10 *Let an example ACA of Figure 3.2 be represented by the Matrix T with inversion vector $F \neq 0$, where*

$$T = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The characteristic polynomial of the matrix(T) *is* $(x^3 + x + 1)(x^2 + x + 1)$ *and the cycle structure of the LCA is [1(1),1(3),1(7),1(21)]*

The C'*, as per the* Theorem 3.11*, has the same cycle structure as that of the corresponding LCA - C, irrespective of the value of F. We illustrate the result of the theorem for a particular cycle length (say) 7. As per the theorem, we enumerate* $\alpha_1 = T^7 + I$, $\alpha_2 = T^3 + T + 1$, $\alpha_3 = T^6 + T^5 + T^4 + T^3 + T^2 + T + I$, *where*

$$T^7 + I = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad T^3 + T + 1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T^6 + T^5 + T^4 + T^3 + T^2 + T + I = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

All the three matrices with 3^{rd}, 4^{th} *&* 5^{th} *row as zero have the same null space solution :*

{*00000, 00001, 00011, 00010, 00100, 00001, 00100, 00110, 00111*}.

Therefore, Rank(α_1*) = Rank(*α_2*,*α_3*),*

and Rank(α_1*) = Rank(*α_2*,*$\mathcal{F}$*),* $\mathcal{F}$*=*$(I+T+T^2)\cdot F$*, F being a 5-dimensional inversion vector. Therefore, both* C' *and C have cycle of length 7. The number of states having cycle length 7 or sub-multiple of 7 (here 1) is 8. Therefore, the cycle component of cycle length 7 =* $\frac{(8-1)}{7}$ *=1.*

Enumerating in this fashion, the complete cycle structure of C' *can be shown as* [1(1), 1(3), 1(7), 1(21)]*, same as that of C.*

3.3.3 P_3 : Nature of the Class of ACA for which the Cycle Structure differs from that of LCA

From the *Theorem 3.11* it is obvious that the cycle structure of C and C' can differ only if the characteristic polynomial has a factor of $(x+1)$. Hence, study of the role of the factor $(x+1)$ is necessary. The subsequent analysis first concentrates on the LCA/ACA having a single elementary divisor as $(x+1)^n$ and next proceed for its generalization. We introduce terminologies employed in subsequent discussions.

- $(\mathbf{C}((\mathbf{x}+\mathbf{1})^{\mathbf{n}}))$: The LCA having characteristic matrix T with characteristic and minimal polynomial $(x+1)^n$. The cycle structure of the LCA is given by

$$CS = [1(1) + \sum_{j=0}^{m} \mu_{2^j}(2^j)] \quad where\ m = \lceil log_2(n) \rceil. \tag{3.37}$$

and

$$Rank\ of\ (T+I)\ =\ n-1,\ and\ Rank\ of\ (T+I)^i\ =\ n-i. \tag{3.38}$$

- $[\mathbf{F^k}]$: The set of inversion vectors which annihilates only $(x+1)^k$, that is,

$$(T+I)^k \cdot x = 0,\ while\ (T+I)^{k'} x \neq 0\ where\ k' < k\ \&\ x \in [F^k]. \tag{3.39}$$

- $\mathbf{Car}[\mathbf{F^k}]$: Cardinality of the set F^k.

Theorem 3.12 *The cycle structure of ACA-$C'(x+1)^n$ differs from that of its linear counterpart LCA-$C(x+1)^n$ if and only if the inversion vector of ACA - $F \in [F^n]$.*

Proof : In order to test whether the cycle structure of $C(x+1)^n$ and $C'(x+1)^n$ are identical, the consistency of the *relation 3.33* is checked for presence of a cycle of length k in the C',where k is the cycle formed by C. The LCA has cycles of length k where

$$k = 2^j, \quad j = (0, 1, 2, \cdots, m), \quad m\ =\ \lceil log_2(n) \rceil. \tag{3.40}$$

Since the cycle lengths (k) of the $C(x+1)^n$ is of the form 2^j, the relation for consistency can be rewritten as

$$(T^{2^j} + I)x = (I + T + T^2 + \cdots + T^{2^j - 1})F \tag{3.41}$$

where $j = (0,\ 1,\ 2,\ \cdots,\ m)$, $m = \lceil log_2(n) \rceil$.
The relation has to be satisfied for all values of j to establish equivalence of cycle structure of C & C'. Since

$$(T^{2^j} + I) = (T+I)^{2^j}\ \&\ (I + T + T^2 + \cdots + T^{2^j - 1}) = (T+I)^{2^j - 1}$$

we have

$$(T+I)^{2^j} x = (T+I)^{2^j - 1} F \tag{3.42}$$

In order to prove the theorem, the consistency of the above equation is checked for the following two cases -

- Case 1. $C'(x+1)^n$ with inversion vector $F \in [F^{n'}]$ where $n' < n$.
- Case 2. $C'(x+1)^n$ with inversion vector $F \in [F^n]$.

The proof establishes that in the first case both C and C' have the same cycle structure while in the second case the cycle structure of C' differs from that of C. The proof is established by checking consistency for every cycle length ($k = 2^j$, $j = \{0,1,2,\cdots,m\}$)
Case 1: Inversion vector $F \in [F^{n'}]$ *where* $n' < n$.

The *relation 3.42* can be rewritten as

$$(T+I)^{2^j-1}[(T+I)x = F] \quad where \quad F \in [F^{n'}] \tag{3.43}$$

Therefore, the consistency of the following relation

$$(T+I)x = F$$

will directly prove the consistency of *relation 3.42*. We show that the relation is consistent if and only if $x \in [F^{n'+1}]$.
If $x \in [F^{n'+1}]$, the state transition equation can be written as

$$(T+I)^{n'+1} \cdot x = 0 \Rightarrow (T+I)^{n'} \cdot [(T+I)x] = 0 \tag{3.44}$$

Since according to definition, $(T+I) \cdot x \neq 0$, therefore, the vectors y will be formed by enumerating the equation

$$(T+I)x = y \tag{3.45}$$

Moreover, the *relation 3.42* can be rewritten as

$$(T+I)^{n'} \cdot y = 0 \quad \Rightarrow \quad y \in [F^{n'}] \quad i.e. \quad (T+I)^{n''} \cdot y \neq 0 \quad n'' < n \tag{3.46}$$

Since rank of $(T+I)$ is n -1, therefore, 2 different x, (x_1, $x_2 \in [F^{n'+1}]$), when premultiplied with $(T+I)$ will produce the same y in *relation 3.45*. That is,

$$(T+I) \cdot x_1 = (T+I) \cdot x_2 = y$$

Hence, exploiting all the possible pairs of x, we obtain the set of y have cardinality $\frac{Car(F^{n'+1})}{2}$ where the cardinality of $[F^{n'+1}]$ is denoted as $\mathrm{Car}(F^{n'+1})$.
Since rank of $(T+I)^{n'+1}$ is one less than that of $(T+I)^{n'}$, therefore directly from rank relationship (*relation 3.38*)

$$Car(F^{n'+1}) = 2 \times Car(F^{n'})$$

Therefore, $Car(y) = Car(F^{n'})$ that is the full set of $(F^{n'})$ is represented by y. That implies the relation

$$(T + I)x = F$$

is consistent for all values of $F \in [F^{n'}]$ and consequently, the *relation 3.42* is consistent for all F $\in [F^{n'}]$, $n' < n$.

Case 2 : Checking Consistency of Cycle Length for ACA $C'(x+1)^n$ with inversion vector $F \in [F^n]$

In this case, it is seen that *relation 3.42* is inconsistent for all cycle length $2^j \leq n$. Multiplying either side of the *relation 3.42* with $(T + I)^{n-2^j}$. we obtain

$$(T + I)^n x = (T + I)^{n-1} F. \tag{3.47}$$

It is inconsistent since the $left\ hand\ side(LHS) = 0$ while $right\ hand\ side(RHS) \neq 0$. Since the characteristic and minimal polynomial of C is $(x + 1)^n$, the *relation 3.42* is consistent if $2^j - 1 \geq n$ when both $LHS\ \&\ RHS = 0$. □

Corollary 3.2 : *The cycle structure of $C'(x+1)^n$ with $F \in F^n$ is*

$$CS = \mu'(2^{m'}), \quad \mu' = 2^{n-m'} \quad \& \quad m' = \lfloor log_2(n) \rfloor + 1 \tag{3.48}$$

where the cycle structure of $C(x+1)$ is

$$CS = [1(1) + \sum_{j=0}^{m} \mu_{2^j}(2^j)] \quad where\ m = \lceil log_2(n) \rceil.$$

Proof : From *Theorem 3.12*, the *relation 3.42* becomes consistent if $2^j - 1 \geq n$, that is cycle of length 2^j exists in C' if $2^j - 1 \geq n$. The minimum value of j, at which the equation becomes consistent is represented by m' where $2^{m'-1} \leq n < 2^{m'}$.
If $n = 2^{\lceil log_2 n \rceil}$, then n is of the form 2^j where j is an integer. Therefore, $\lceil log_2 n \rceil = \lfloor log_2 n \rfloor = log_2 n$. In that case, $m' = \lfloor log_2 n \rfloor + 1$.
If $n < 2^{\lceil log_2 n \rceil}$, then $m' = \lceil log_2 n \rceil = \lfloor log_2 n \rfloor + 1$. Combining both cases, $m' = \lfloor log_2 n \rfloor + 1$.
Since all the states fall in the Null Space of $(T + I)^{2^{m'}}$, the ACA will have only cycles of length $2^{m'}$ and the number of cyclic component(μ) will be $2^n/2^{m'} = 2^{n-m'}$. □

The following example illustrates the result of the theorem & corollary.

Example 3.11 *The T matrix and two inversion vector F of a 4 cell GF(2) CA is given by*

$$T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad F_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad F_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Characteristic and Minimal Polynomial of the Matrix is $(x+1)^4$.
The cycle structure(CS_C) *of the LCA -* $C((x+1)^4)$ *with characteristic matrix* T *is given by* [2(1),1(2),3(4)].
The inversion vector F_1 *annihilates* $(T+I)^4$. *The ACA* $C'(x+1)^4$ *with characteristic matrix* T *and inversion vector* F_1 *changes cycle structure to [2(8)].*
The inversion vector F_2 *doesn't annihilate* $(T+I)^4$. *The ACA* $C'(x+1)^4$ *with characteristic matrix* T *and inversion vector* F_2 *has its cycle structure* $CS_{C'}$ *identical to* CS_C *generated by LCA* $C(x+1)^4$

Generalization of Theorem 3.12: The above theorem is now extended for a more generalized class of LCA/ACA. The cycle structure of C', as per *Theorem 3.11*, can differ from C only if the characteristic polynomial of C/C' contains a factor of $(x+1)$. The *Theorem 3.12 & corollary 3.2* has characterized the role of the invariant factor $(x+1)$ and its relation with the inversion vector F. We here define such class of CA as Eligible Cellular Automata (ECA).

Definition 3.16 *A* LCA/ACA *whose characteristic polynomial contains a factor of* $(x+1)$ *is termed as Eligible Linear(Additive) Cellular Automata (*$ELCA/EACA$*). The characteristic polynomial of* $ELCA/EACA$ *can be represented in Elementary Divisor Form(EDF) as*

$$f(x) = (x+1)^{n_1} \cdot\cdot (x+1)^l \phi_{l+1}(x)^{n_{l+1}} \cdot\cdot \phi_l(x)^{n_{\mathcal{N}}} \tag{3.49}$$

where (i). each $\phi_i(x)^{n_i}$ *is an elementary divisor and (ii). the irreducible factor of the first* l *elementary divisors is equal to* $(x+1)$. *These elementary divisors are also termed as* $(x+1)$ *- elementary divisors.*

The cycle structure of an $ELCA$ *from* relation 3.16 *is of the form*

$$CS_E LCA = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)] \quad k_1 = 1$$

The following theorem states the nature of cycle structure of corresponding Eligible ACA.

Theorem 3.13 : *Given the cycle structure of* $ELCA$ *with characteristic matrix* T *as*

$$CS_{ELCA} = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)] \quad k_1 = 1$$

the cycle structure of $EACA$, with characteristic matrix T and inversion vector F, is of the form

$$CS_{EACA} = \sum_{k_i} \sum_{j=\mathcal{M}}^{m_{k_i}} \mu'_{2^j \cdot k_i}(2^j \cdot k_i), \quad k_1 = 1 \tag{3.50}$$

$$\mu'_{2^{\mathcal{M}} \cdot k_i} = \begin{cases} \sum_{j=0}^{\mathcal{M}} \frac{(\mu_{2^j \cdot k_i} \times 2^j \cdot k_i) + 1}{2^{\mathcal{M}} \cdot k_i} & k = 1 \\ \sum_{i'=0}^{\mathcal{M}} \frac{\mu_{2^{i'} \cdot k_i} \times 2^{i'} \cdot k_i}{2^{\mathcal{M}} \cdot k_i} & otherwise \end{cases} \quad and \quad \mathcal{M} = 2^{\lfloor log_2 n_i \rfloor + 1} \tag{3.51}$$

and

$$\mu'_{2^j \cdot k_i} = \mu_{2^j \cdot k_i} \quad j > \mathcal{M} \tag{3.52}$$

where the $(x+1)$-elementary divisor with largest power annihilated by F is n_i.

Proof : We develop the proof with the assumption that $(x+1)^{n_i}$ is the *only* factor annihilated by F. It can be easily generalized to the fact that $(x+1)^{n_i}$ is the *largest* factor annihilated by F.

The characteristic polynomial $f(x)$ is denoted as

$$f(x) = (x+1)^{n_i} \times \tilde{\phi}(x) \tag{3.53}$$

where

$$\tilde{\phi}(x) = \phi_1(x)^{n_1} \cdot \cdot \phi_{i-1}(x)^{n_{i-1}} \cdot \phi_{i+1}(x)^{n_{i+1}} \cdot \cdot \phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}} \tag{3.54}$$

The cycle structure $CS_{\tilde{\phi}(x)}$ corresponding to $\tilde{\phi}(x)$ is same for both LCA and ACA. Let F be the inversion vector which annihilates the factor $(x+1)^{n_i}$.

As per the *Theorem 3.12*, the cycle structure generated due to the factor $(x+1)^{n_i}$ differ in C and C'. The cycle structures are respectively $CS_{C'(x+1)^{n_i}}$ and $CS_{C(x+1)^{n_i}}$ from*corollary 3.2*, where

$$CS'_{C'(x+1)^{n_i}} = \mu'(2^{m'}), \quad \mu' = 2^{n_i - m'} \quad \& \quad m' = \lfloor log_2(n) \rfloor + 1$$

and

$$CS_{C(x+1)^{n_i}} = [1(1) + \sum_{j=0}^{m} \mu_{2^j}(2^j)] \quad where\ m = \lceil log_2(n) \rceil.$$

from *relation 3.48 & 3.37.*

Therefore, the cycle structure of $CS_{C(\phi(x))}$ and $CS_{C'(\phi(x))}$ are respectively represented as

$$CS_{C(\phi(x))} = CS_{C(\tilde{\phi}(x))} \times CS_{C((x+1)^{n_i})} \tag{3.55}$$

$$CS'_{C'(\phi(x))} = CS'_{C(\tilde{\phi}(x))} \times CS'_{C'((x+1)^{n_i})} \tag{3.56}$$

The cycle structure of $C(\tilde{\phi}(x))$ follows from *relation 3.16*

$$CS_{C(\tilde{\phi}(x))} = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)]$$

The cross product in *relation 3.55* yields cycle structure of $CS_{C(\phi(x))}$ which is

$$CS_{ELCA} = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)] \quad k_1 = 1$$

whereas the cross product in *relation 3.56* yields $CS_{C'(\phi(x))}$ where

$$CS_{EACA} = \sum_{k_i} \sum_{j=\mathcal{M}}^{m_{k_i}} \mu'_{2^j \cdot k_i}(2^j \cdot k_i) \quad k_1 = 1 \quad \mu'_{2^{\mathcal{M}} \cdot k_i} \quad = \quad \sum_{j=0}^{\mathcal{M}} \frac{\mu_{2^j \cdot k_i} \times 2^j . k_i}{2^{\mathcal{M}} . k_i} \tag{3.57}$$

□

The theorem provides some important inference which is used to develop the algorithm for enumerating cycle structure of $EACA$. The inferences are

Inference 3.1 : *All cycles of length above $2^{\mathcal{M}} \cdot k_i$ have same component in $ELCA$ and $EACA$.*

Inference 3.2 : *There is a striking similarity between $CS_{C(\phi(x))}$ and $CS'_{C'(\phi(x))}$. The change between each individual clustering of k_i in $CS_{C(\phi(x))}$ and $CS'_{C'(\phi(x))}$ respectively is that all the cycle length $\leq 2^{\mathcal{M}} . k_i$ gets converted to $2^{\mathcal{M}} . k_i$ and the cyclic component corresponding to $2^{\mathcal{M}} \cdot k_i$ in ACA are adjusted to cover the complete state space generated by cycle lengths ($\leq 2^{\mathcal{M}} \cdot k$) accordingly.*

We also define two terms directly arising from the theorem.

Definition 3.17 *Minimum Additive Factor : The least cycle length of any primary cycle (k) in an additive cycle is denoted by $2^{\mathcal{M}} \cdot k$, where $\mathcal{M}$ is denoted Minimum Additive Factor.*

Definition 3.18 *Legal Additive Cycle Structure : A Legal Additive Cycle Structure is the cycle structure which can be expressed by relation 3.51 and where the corresponding Linear Cycle Structure* $CS = [1(1) + \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)]$ *is a Legal Linear Structure.*

The results of *Theorem 3.13* is illustrated below with the example ACA.

Example 3.12 *The T matrix of a 5 cell GF(2) CA is given by*

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{and } F_1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \text{and } F_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

- *The characteristic polynomial* $f(x)$ *of the example LCA/ACA in Elementary Divisor Form(EDF) is*
  $f(x) = (x+1)(x+1)^2(x^2+x+1)$.
- *The inversion vector F= [0 1 1 0 0]. The vector* F_1 *annihilates only* $(x+1)^2$. *Therefore,* $\mathcal{M} = \lfloor log_2(2) \rfloor + 1 = 2$.
- $f(x)$ *can be expressed as* $\tilde{\phi}(x) \times (x+1)^2$, *where* $\tilde{\phi}(x) = (1+x) \cdot (x^2+x+1)$.
- *The cycle structure* $CS_{C(\tilde{\phi}(x))} = [CS_{C(1+x)}] \times [CS_{C(x^2+x+1)}] = [1(1), 1(1)] \times [1(1), 1(3)] = [2(1), 2(3)]$.
- *The cycle structure* $CS_{C(1+x)^2} = [1(1), 1(1), 1(2)]$ *whereas* $CS_{C'(1+x)^2} = [1(4)]$.
- *Therefore, the resultant structure* $CS_{C(\phi(x))} = CS_{C(\tilde{\phi}(x))} \times CS_{C(1+x)^2} = [4(1), 2(2)], [4(3), 2(6)]$. *The set of odd cycles* $CL = \{1, 3\}$.
- *The resultant cycle structure* $CS_{C'(\phi(x))} = CS_{C(\tilde{\phi}(x))} \times [CS_{C'(1+x)^2}] = [2(4), 2(12)]$.
- *Similarly,* $F_2 = [111100]$ *annihilates both elementary divisor* $(x+1)$ *and* $(x+1)^2$. *Therefore,* $CS_{C(\tilde{\phi}(x))} = [2(2), 2(6)]$. *The resultant cycle structure* $CS = CS_{C(\tilde{\phi}(x))} \times CS_{C'(x+1)^2} = [2(2), 2(6)] \times [1(4)] = [2(4), 2(12)]$, *same as the previous.*
- *Note : The change between each individual clustering of* k_i *in* $CS_{C(\phi(x))}$ *and* $CS_{C'(\phi(x))}$ *respectively is according to* relation 3.51 *of* Theorem 3.13. *Taking for example 2(4) where* k_1 *=1 and* $2^{\mathcal{M}}$ *=4, the corresponding cycle structure in C which have got merged in C′ are [4(1),2(2)]. The cycle component of* $\mu_{2^2 \cdot 1}$ *has been formed according to* relation 3.51, $\mu_{2^2 \cdot 1} = \frac{4 \times 1 + 2 \times 2}{4} = 2$.

3.3.4 P_4 : Algorithm for Enumerating Cycle Structure of an ACA

The *Theorem 3.13* clearly establishes the relationship between the cycle structure of C and C'. It shows that once the cycle structure of C is evaluated and arranged as per the format noted in *relation 3.37* , the *whole task of the algorithm for evaluating the cycle structure of an ACA reduces to the deduction of the value of* $\mathcal{M}$.

The value of $\mathcal{M}$ is deduced with the help of the *Theorem 3.10*. The rank of $[T^k + I]$ and $[T^k + I, \mathcal{F}]$ is successively compared for all $k = 2^m \cdot k_1$, $m \geq 0$ until the rank becomes equal. The value of k at which both the ranks become equal gives us the necessary $\mathcal{M}$.

The algorithm for deduction of $\mathcal{M}$ is next elaborated.

Algorithm 3.8 *Enum_*$\mathcal{M}(T, F)$
Input : T matrix, F Vector.
Output : $\mathcal{M}$
$\mathcal{M}$ *= 0*
Evaluate $\mathcal{F} = [I + T + T^2 + ... + T^{2^{\mathcal{M}}-1}]F$
Evaluate R1 = rank$[T^{2^{\mathcal{M}}-1} + 1]$ *and R2 = rank*$[T^{2^{\mathcal{M}}-1} + I, \mathcal{F}]$
while ($R1 < R2$*)*
{
Evaluate $\mathcal{F} = [I + T + T^2 + ... + T^{2^{\mathcal{M}}-1}]F$
Evaluate R1 = rank$[T^{2^{\mathcal{M}}-1} + 1]$ *and R2 = rank*$[T^{2^{\mathcal{M}}-1} + I, \mathcal{F}]$
$\mathcal{M}$ *=* $\mathcal{M}$ *+ 1*
}
return $\mathcal{M}$.

Once the $\mathcal{M}$ is enumerated, the modified cycle structure for each k_i is deduced through *relation 3.51.*

Algorithm 3.9 *Enum_CS'_depth_from_ACA(T, F, CS', d)*
Input : (1) The Characteristic Matrix T and Inversion Vector F.
OUTPUT: The Cycle structure of C' & depth
Enum_CS_depth_from_LCA(T, CS, d)
where

$$CS_C = [1(1) \sum_{k_i} \sum_{j=0}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)]$$

where k_i *is odd and CL is the set of odd cycles* k_i.
Output the depth of C as that of C'.
If characteristic polynomial of T doesn't have a factor of $(x + 1)$*, then*
Output the cycle structure CS of C as CS' of C'.
else
$\mathcal{M}$ *= Enum_*$\mathcal{M}(T, F)$

for each $k_i \in CL$
{
Evaluate $\mu_{2^{\mathcal{M}}.k_i}$ *for each* k_i *by relation 3.51*
$\mu'_{2^j.k_i} = \mu_{2^j.k_i}$ *for* $j > \mathcal{M}$
}
Output $CS' = \sum_{k_i} \sum_{j=\mathcal{M}}^{m_{k_i}} \mu_{2^j.k_i}(2^j \cdot k_i), \quad k_1 = 1$

We now illustrate the algorithm of enumerating ACA with an example ACA. The ACA is represented by T matrix and F vector where the T matrix taken in the following example is one that has been used in *Example* **??**.

Example 3.13 *Let the T matrix of a 7 cell GF(2) ACA be*

$$T = \begin{pmatrix} \lceil 0 & 0 \rceil & 0 & 0 & 0 & 0 & 0 \\ \lfloor 1 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & [1] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lceil 1 & 1 \rceil & 0 & 0 \\ 0 & 0 & 0 & \lfloor 0 & 1 \rfloor & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lceil 0 & 1 \rceil \\ 0 & 0 & 0 & 0 & 0 & \lfloor 1 & 1 \rfloor \end{pmatrix} \quad \textit{and F} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The matrix $[T^k + I, \mathcal{F}]$ *is referred to as* Augmented Matrix.
Step 1 & 2 : Finding Characteristic Polynomial and Cycle Structure.
Characteristic Polynomial $x^2 \cdot (x+1)^3 \cdot (x^2+x+1)$
The cycle structure of the LCA is [4(1), 2(2), 4(3), 2(6)].
Depth of LCA = 2 $\Rightarrow$ *Depth of ACA =2*
Step 3 : Finding the values of $\mathcal{M}$.

- *Rank of* $(T^1 + I)$ *is 3, while the rank of the augmented matrix is 4. Hence cycle of length 1 does not exist*
- *Rank of* $(T^2 + I)$ *is 2, while the rank of the augmented matrix is 3. Hence cycle of length 2 also does not exist in the ACA*
- *Rank of* $T^4 + I$ *is 2, while the rank of the augmented matrix is 2. Hence cycle of length 4 exists.*
  So, the value of $\mathcal{M}$*, as per Step 3, is* 2.

Step 4 : Finding the Component
1. $k_i = 1$ $\mu_{2^2.1} = \frac{4\times1+2\times2+0\times4}{2^2.1} = 2.$
2. $k_i = 3$ $\mu_{2^2.3} = \frac{4\times3+2\times6+0\times12}{2^2.3} = 2.$
Hence the cycle structure becomes [**2(4), 2(12)**] .

3.4 Conclusion

This chapter presents the complete characterization of the cyclic and non-cyclic subspaces of Additive Cellular Automata(ACA). The proofs of related theorems and algorithms to

identify the cycle structure and depth of the state transition behavior of an ACA are presented. The analytical framework for characterization of the vector subspace of the ACA is based on the characteristic polynomial, minimal polynomial, elementary divisor, characteristic matrix and inversion vector of the ACA. Based upon analysis, we develop the synthesis scheme in the next chapter.

Chapter 4

GF(2) Cellular Automata - Synthesis

This chapter details the methodology of synthesizing GF(2) CA from a given the vector sub-space in terms of cyclic and non-cyclic components. The analysis of GF(2) CA in the previous chapter provides the foundation for the synthesis scheme.

Cellular Automata (CA), as noted in the survey of *Chapter 2*, appealed to researchers both for their theoretical interest as well as practical applications. In recent times, CA has been effectively employed in the areas of pseudo-random testing, cryptography [56, 28, 32, 38], signature analysis[31, 20], error correcting code[14, 16], image compression[41], pattern recognition, etc. However, all these applications depend on the availability of a CA with the specified characteristics. Principal among such characteristics is the desired state transition behavior of the CA to be used for an end application. This can be conveniently expressed as the cycle set and depth specification of a CA formulating a CA from a specification of the cycle structure and depth.

A number of researchers have concentrated their efforts for CA synthesis. Notable among them is the work of Kattel & Muzio[4, 24] where they proposed an algorithm for synthesis of CA from irreducible polynomial. Serra et al [47] presented an algorithm for synthesis of an 90/150 hybrid CA from the specification of the characteristic polynomial.

The first attempt of synthesis of Linear CA from a given cyclic/ non-cyclic vector sub-spaces through a table driven approach has been done by [44]. But the scheme fails to guarantee the generation of a solution even if a LCA exist for the given input. However, no attempt has been made to generalize the problem and effectively synthesize an ACA from a given set of cycle and depth specification. But the scheme fails to generate CA from all legal consideration. To the best of our knowledge none of other works so far reported in the literature has dealt with synthesis of a ACA to generate a specified cyclic/non-cyclic vector sub-spaces. The scheme reported in this chapter perfects the scheme for LCA, that is the scheme is always able to generate an LCA if the given input is valid, generalizes the scheme for ACA and reports an efficient synthesis methodology which has polynomial time

complexity for all practical consideration.

The cyclic/non-cyclic vector subspaces generated by an ACA follows some symmetry dictated by the laws of linear algebra. Naturally, any arbitrary asymmetric cyclic subspace cannot be generated by an ACA. The cycle structure which can be generated by an GF(2) CA is termed as legal cycle structure. In this chapter, we explore into the methodology of synthesizing GF(2) CA from a given legal cycle structure. The features of Legal Additive/Linear Cycle Structure are defined in *Definition 3.12* & *3.18*. If an illegal cycle structure is given the algorithm itself can understand that it is illegal. Accordingly, it returns that the cycle is illegal.

The chapter is organized similar to *Chapter 3*. We first present the methodology for synthesis of Linear CA and then based upon the results the algorithm is generalized for Additive CA.

4.1 Synthesis of an LCA

The characterization of LCA detailed in the last chapter has provided the foundation for its reverse operation - Synthesis. The synthesis scheme accepts a given legal cyclic (*Definition 3.12*) and non-cyclic subspace as input and outputs an LCA generating the vector space. The basic approach of synthesis is outlined below.

- At the first stage, elementary divisors are derived from the cycle structure(CS). *Primary Cycle Structure(PCS)* and *Primary Family Cycle Structure(PFCS)* enumeration acts as an intermediate step in this derivation process.
- The depth d of the CA is realized by the elementary divisor type x^d.
- Each elementary divisor $\phi_i(x)^{n_i}$ refers to a vector space represented by a equivalent T_i matrix. Total space is the direct sum of the vector spaces generated by individual elementary divisor. Consequently, the LCA represented by a T matrix is obtained by placing the individual matrices(T_i) in block diagonal form.
- If an illegal input is given, the algorithm reports that such a cycle structure is not feasible.

The major steps for the synthesis of LCA from legal cycle structure are noted in the algorithm reported below:

Algorithm 4.1 *Enum_LCA_from_CS_depth*(CS, d, T)
Input : *Cycle Structure* (*CS*), *depth*(*d*)
Output : *LCA* - (*T matrix*)
A : Generate each individual Primary Family Cycle Structure ($PFCS$) (Definition **??**) *from* CS.
B : Generate Primary Cycle Structures(PCS) *in triplet form* (Definition 3.9) *from each* $PFCS$.
C1. Find the elementary divisor $\phi_i(x)^{n_i}$ *corresponding to each primary cycle structure*

(PCS_i) and
D : Find $LCA_i(T_i)$ corresponding to each $\phi_i(x)^{n_i}$ and Find the $LCA_i(T_i)$ generating the non-cyclic space - depth d.
Synthesize the LCA combining the set of LCA_i generated.

Each of the above steps have been detailed in subsequent sub-sections. However, in order to express the underlying principle of synthesis we take the help of the following example illustrating each of the steps.

Example 4.1 *To design an LCA having its CS = [4(1), 2(2), 4(3), 2(6)] & depth(d) = 2.*
Step A: *The given CS can be expressed as a product of two PFCS as noted below. CS = [1(1),3(1),2(2)] × [1(1),1(3)]*
Step B: *From the two PFCS, three PCS can be derived in triplet form and so CS can be represented as $CS = (1,1)^1 \times (1,1)^2 \times (1,3)^1$*
Step C1: *Corresponding to the set of PCS, the elementary divisors $(x+1)$, $(x+1)^2$, (x^2+x+1) are obtained.*
Step C2: *From depth(d) we obtain the elementary divisor x^2.*
Hence we obtain the characteristic polynomial in elementary divisor form as $x^2(x+1)(x+1)^2(x^2+x+1)$.
Step D: *From the elementary divisors we obtain the LCA (T matrix) as*

$$T = \begin{pmatrix} \lceil 0 & 0 \rceil & 0 & 0 & 0 & 0 & 0 \\ \lfloor 1 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & [1] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lceil 1 & 1 \rceil & 0 & 0 \\ 0 & 0 & 0 & \lfloor 0 & 1 \rfloor & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lceil 0 & 1 \rceil \\ 0 & 0 & 0 & 0 & 0 & \lfloor 1 & 1 \rfloor \end{pmatrix}$$

Each of the steps is next discussed in detail.

4.1.1 Generation of $PFCS$ from Cycle Structure(CS)

The conversion tool accepts the given cycle structure(CS) and partitions it into corresponding $PFCS$s such that CS follows the relation

$$CS = [PFCS_1] \times [PFCS_2] \times \cdots \times [PFCS_{\tilde{\mathcal{N}}}] \tag{4.1}$$

where $\tilde{\mathcal{N}}$ specifies the number of *Primary Family Cycle Structure* in the synthesized LCA.

The overview of the conversion algorithm is stated below.
Step I : In the first step we divide $CS = CS_1 \times CS_2$ where CS_1 is Primary Family Cycle Structure(PFCS) while CS_2 follows the relation $CS_2 = CS$ - { CS_1, $CS_1 \times CS_2$}. While satisfying the following condition, the primary cycle k_1 produced by CS_1 is the smallest among all primary cycle lengths in CS.

Step II : Making CS_2 as CS, the steps are repeated until $CS = CS_1$.

As it is obvious from the above overview, the algorithm is recursive in nature. Hence, developing the algorithm for solving one step will lead to solution of the entire problem.

1. Generation of CS_1 The cycle structure CS_1 is generated with the help of *Theorem 3.8*, which states that the cycle structure corresponding to the smallest primary cycle family is same in CS & CS_1. The family whose primary cycle is least forms CS_1. The following example illustrates the generation of CS_1

Example 4.2 *Given* CS=[1(1), [3(1),2(2)], [4(3),2(6)], [12(5),6(10)], [12(15),6(30)]]
Then CS_1 *is the cycle structure obtained from the k- cycle family, where k has the least value.*
Since here 1 is lowest, therefore $k_1 = 1$ & CS_1 = [1(1), 3(1),2(2)].

2. Generation of CS_2 : During generation of CS_2 from CS, the cycle families k_is constituting CS are arranged in ascending order. Each cycle family is tested in ascending order and decision is taken whether to include it in CS_2 or not. When we decide to add a cycle family k_2 in CS to CS_2, we have to keep in mind that in addition to k_2 cycle family, we have to deduct the cyclic components of the cycle family from CS derived from the cross product of CS_1 & CS_2. We already know k_1 - the primary cycle of CS_1. Any cycle $k_3 \in CS$ which has arisen from cross product of k_1 & $k_2 \in CS_2$ where $k_3 = lcm(k_1, k_2)$ should be deducted from CS in the process of formation of CS_2.

In the process of adding a cycle family k_2 in CS to CS, following two cases may arise that are treated separately.

- **Case 1 :** $lcm(k_1, k_2) > k_2$
- **Case 2 :** $lcm(k_1, k_2) = k_2$

Case 1 : $lcm(k_1, k_2) > k_2$
It is illustrated through the following example.

Example 4.3 CS = [1(1), [1(3)], [3(5)], [3(15)]]; $k_1 = 3$.
CS_1 = [1(1), 1(3)]
The remaining cycles in CS *are* $k_i = \{5, 15\}$
Taking the first cycle $k_2 = 5$,
k_1 *is not a factor of* k_2.
Therefore, CS_2 = [1(1), 3(5)] *that is, the entire cycle structure 3(5) in* CS *comes from* CS_2.
The cycle family arising as a cross product of k_1 & k_2 *is* $k_3 = lcm(k_1,k_2) = 15$ &
Therefore, μ_{15} *which should be deducted from* CS *is calculated from relation 3.23*
$\mu_{15} = \mu_3 \cdot \mu_5 \cdot gcd(\mu_3 \cdot \mu_5) = 3$.
Cycles remaining in CS = CS - [3(5), 3(15)] = [1(1)].
Since all the cycles are exhausted, CS_2 = [1(1), 3(5)].

So to summarize, the following decisions and actions can be taken in this case to *Case I.*

- The cycle family (k_2) has not arisen as a result of cross product. As a result, the entire cycle family(k_2) is a member of CS_2.
- The k_3 cycle family derived from cross product of k_2 & k_1 cycle family has to be accordingly deducted from CS. The components of k_3 cycle family are obtained through *Relation 3.23.*
- Consequently, in this step, two cycle family (k_2, k_3) of CS are resolved; k_2 becomes a member of CS_2 while the effect of cross product of k_2 & k_1 cycle family is resolved with the nullification of k_3 cycle family.

Case 2 : $lcm(k_1, k_2) = k_2$
In this case, the cross product of k_1-cycle family and k_2-cycle family yields a cycle family whose primary length is also k_2. So unlike the previous case, instead of adding the entire cycle family to CS_2 from CS, we have to add a part of it, so that the effect of cross product of CS_1 and CS_2 gets balanced.

The case where cross product of k_1 & k_2 cycle family produces a k_2 cycle family is discussed through *Relation 3.29* where evaluation of $\tilde{\mu}_{2^j \cdot k_2}$ in CS from $\mu_{2^j \cdot k_2}$ in CS_2 is made. To evaluate the cyclic component $\mu_{2^j \cdot k_2}$ to be added to CS_2 from the given cyclic component $\tilde{\mu}_{2^j \cdot k_2}$ in CS the *relation 3.29* has been re-framed as follows

$$\mu_{2^j \cdot k_2} = \frac{\tilde{\mu}_{2^j \cdot k_2}(CS) - [\mu_{2^j \cdot k_1}(CS_1)] \cdot [S_{2^{j-1} \cdot k_2}(CS_2)] \cdot \frac{k_1}{k_2}}{[S_{2^j \cdot k_1}(CS_1)] + 1} \tag{4.2}$$

where

- $\tilde{\mu}_{2^j \cdot k_2}(CS)$ is the cyclic component corresponding to k_2-cycle family in CS.
- $\mu_{2^j \cdot k_1}(CS_1)$ is the cyclic component corresponding to k_1-cycle family in CS_1.
- $S_{2^j \cdot k_1}(CS_1)$ means number of states covered by cycle length ($\leq 2^j \cdot k_2$) in CS_1.
- $S_{2^{j-1} \cdot k_2}(CS_2)$ means number of states covered by cycle length ($\leq 2^{j-1} \cdot k_2$) in CS_2. When we are enumerating $\mu_{2^j \cdot k_2}$, all the cyclic component $\mu_{2^j \cdot k_2}$ ($j < j$) is already enumerated.

The following example illustrates the manner in which *equation 4.2* is used to extract individual cycle structure.

Example 4.4 CS=[1(1), [3(1),2(2)], [4(3),2(6)], [12(5),6(10)], [12(15),6(30)]]
k_1 = *1.(the smallest cycle length)*
CS_1 = [1(1), 3(1), 2(2)]
The cycle structure of CS *after* CS_1 *is subtracted from* CS *leading to* CS = [1(1), [4(3), 2(6)], [12(5), 6(10)], [12(15), 6(30)]]
The remaining primary cycles in CS $k_i = \{3,5,15\}$. *For each* k_i; $lcm(k_i, 1) = k_i$.

$k_2 = 3$.
From relation 4.2 - $\mu_3 = \frac{\tilde{\mu}_3(CS) - \mu_1(CS_1)\cdot S_0(CS_2)\cdot\frac{1}{3}}{S_1+1} = \frac{4-4\times0\times\frac{1}{3}}{3+1} = 1$
$\mu_6 = \frac{\tilde{\mu}_6 - \mu_2\cdot S_3\cdot\frac{1}{3}}{(S_2+1} = \frac{2-2\times3\times\frac{1}{3}}{7+1} = 0$
Similarly, for $k_2 = 5$, $\mu_5 = 3$, $\mu_{10} = 0$;
and $k_2 = 15$, $\mu_{15} = 3$, $\mu_{30} = 0$;
Then $CS_2 = [1(1), 1(3), 3(5), 3(15)]$

The algorithm for enumeration of both CS_1 and CS_2 is presented next. In the following algorithm, $\tilde{\mu}$ is used to denote cyclic components in CS, while μ is used to denote cyclic components in CS_1 & CS_2.

Algorithm 4.2 *Enum_*CS_1&CS_2*_from_*$CS(CS, CS_1, CS_2)$
Input : $CS = [1(1), \sum_{k_i}\sum \mu_{2^j\cdot k_i}(2^j \cdot k_i)]$
Output : CS_1, CS_2
$CS_1 = [1(1), \sum \mu_{2^j\cdot k_1}(2^j \cdot k_1)]$, *where* k_1 *is the smallest member of* k_i *- cycle family set.*
$CS = CS - \sum \mu_{2^j\cdot k_1}(2^j \cdot k_1)$.
for each $k_i \in CS_i$ *in ascending order*
$k_2 = k_i$
if k_1 *is a factor of* k_2
{
Obtain $\mu_{2^j\cdot k_2}(CS_2)$ *from* relation 4.2
$CS_2 \leftarrow CS_2 + \mu_{2^j\cdot k_2}$
$CS \leftarrow CS$ - $\tilde{\mu}_{2^j\cdot k_2}$
*/*Although* $\tilde{\mu}_{2^j\cdot k_2}$ *is deducted from* CS, $\mu_{2^j\cdot k_2}$ *cross-porducted with members of* k_1*-cycle family is added to* CS_2 *as* $\mu_{2^j\cdot k_2}$ *is responsible for formation of* $\tilde{\mu}_{2^j\cdot k_2}$ **/*
}
else
{
$CS_2 \leftarrow CS_2 + \tilde{\mu}_{2^j\cdot k_2}$
Obtain $\tilde{\mu}_{2^j\cdot k_3}$ *from* relation 3.23($k_3 = lcm(k_1, k_2)$)
$CS \leftarrow CS - \tilde{\mu}_{2^j\cdot k_3}$
$CS \leftarrow CS - \tilde{\mu}_{2^j\cdot k_2}$
/ Both the* k_2 *cycle family and the impact it creates when cross producted with* k_1 *is removed from CS */*
}

Complexity Analysis Evaluation of *Relation 4.2 & 3.23* in *Algorithm 4.2* needs a constant time. Let CS have m terms. At each iteration, at least one term gets resolved. Therefore, the complexity of the algorithm is O(m).
We now present the relationship between the LCA size n and number of terms in CS - m. We report the important assumptions made during analysis and the final result. The assumptions are

- The probability of two cycle length k_1 & k_2 to be mutually prime is 0.5 while
- If two cycles k_1 & k_2 are not mutually prime then there is 0.25 probability that k_1 is divisible by k_2 that is $lcm(k_1, k_2) = k_2$ or vice versa and there is is 0.25 probability that one is not divisible by other.

With these assumption the expected number of components $E(m)$ is given by the equation

$$E(m) = L \cdot [2^{\frac{n}{K}} + \frac{n}{K} - 4 \cdot (\frac{5}{4})^{\frac{n}{K}} + 3] \tag{4.3}$$

where L is the expected number of components present in a single $PFCS$ and K is the expected size produced through a single $PFCS$.

Hence in average case, the order of the algorithm is $O(2^{\frac{n}{K}})$, which implies the algorithm is super-polynomial in nature.

However, in practical applications we would be interested to synthesize an LCA at most of size 5000. We find that for all value of $n < 5000$, the computation of $E(m)$ through *relation 4.3* gives value which is less than n^3. Hence for all practical purpose the complexity is O(n^3). The complexity calculation comes from the following assumption of L & K.

In such large LCA, whose constituent cycles will satisfy the prime/non-prime probability relationship, the following assumptions regarding L and K can be easily made

- The expected number of components in each $PFCS$ is L is assumed to be 4.
- Let each $PFCS$ comprise of a single elementary divisor $\phi_i(x)^{n_i}$ and the expected degree of the polynomial is assumed to be 15. Since n_i is produced by 4 separate components, hence the value of $n_i = 8$ (follows directly from *Lemma 3.2*). Therefore, $K = 15 \times 2^{4-1} = 15 \times 8 = 120$.

The *Algorithm 4.2* can be recursively called, as noted in *Algorithm 4.3*, to generate $PFCS$ out of a given CS.

Algorithm 4.3 *Enum_PFCS_from_CS* (CS,[$PFCS$])
{
Input : CS - Cycle Structure
Output : Primary Family Cycle Structure Set $PFCS_1$, $PFCS_2 \cdots PFCS\tilde{\mathcal{N}}$: [$PFCS$]
Enum_*CS_1&*CS_2*_from_CS*(CS, CS_1, CS_2) (Algorithm 4.2)
$PFCS_i = CS_1$
$[PFCS] \leftarrow PFCS_i$
If ($CS_2 = PFCS$)
{
$[PFCS] \leftarrow PFCS_{\tilde{\mathcal{N}}}$
Exit

}
else
$CS = CS_2$
Enum_PFCS_from_CS(CS,[$PFCS$])
}

Complexity Analysis : The expected number of $PFCS$ is given by $\frac{n}{K}$ as the expected size of each individual $PFCS$ is K. Hence the complexity of the algorithm is given by $O(n \cdot 2^{\frac{n}{K}})$. However, for practical consideration for $n = 5000$, the complexity will be $= O(n \cdot n^3) = O(n^4)$.

At this point we have obtained a set of Primary Family Cycle Structure($PFCS$). Our next work lies in generating the Primary Cycle Structure(PCS) set from this $PFCS$.

[Note : Each PCS corresponds to an Elementary Divisor(ED). So in subsequent discussions the terms Primary Cycle Structure(PCS) and Elementary Divisor(ED) have been used interchangeably. Consequently the symbol $\mathcal{N}$ denotes the cardinality of the PCS set which is equal to the number of EDs in an LCA.]

4.1.2 Generation of Primary Cycle Structure (PCS) from PFCS

In this step the Primary Family Cycle Structure ($PFCS$) of (say) k cycle family is taken as input and the Primary Cycle Structures in triplet form is given as the output, such that it maintains the relation

$$PFCS = (\mu_k, k)^{n_1} \times (\mu_k, k)^{n_2} \times \cdots \times (\mu_k, k)^{n_{\mathcal{N}}} \tag{4.4}$$

where $\mathcal{N}$ denotes the number of elementary divisors forming the LCA.

In order to achieve this, we have the following two tasks.
Task 1 : Determination of μ_k.
We determine the value of μ_k with the help of the following axiom

Axiom 4.1 : *The irreducible polynomial $\phi(x)$ having cycle length k has cyclic component μ_k; where $\mu_k = (2^l - 1)/k$; l is the smallest integer for which k becomes a factor of $2^l - 1$; the coefficients of $\phi(x)$ lies in GF(2).*

The following three theorems form the foundation block based upon which we derive *Axiom 4.1* The first two theorems show we have to consider the smallest integer l for determination of μ_k and the third theorem proves the irreducibility of the polynomial $\phi(X)$.

Lemma 4.1 *If k is factor of $2^l - 1$, where l is the smallest integer for which k becomes a factor of $2^l - 1$, then k is only factors of $2^{ml} - 1$, where m is a positive integer.*

Proof : We proof the necessary condition first. If k is a factor of $2^l - 1$, then it is also a factor of $2^{ml} - 1$ as $2^l - 1$ is a factor of $2^{ml} - 1$.

Sufficient Condition : Let k be a factor of $2^{l_1} - 1$ where $ml < l_1 < (m+1)l$ for some m. Then $\frac{\mu_1 \times k+1}{\mu_2 \times k+1} = \frac{2^{l_1}}{2^{ml}} = 2^{l_2}$ where $l_2 = l_1 - ml$ and μ_1, μ_2 are results obtained from dividing $2^{l_1} - 1$, $2^{ml} - 1$ by k respectively.
Therefore, the following relation arises

$$k \cdot (\mu_1 + 2^{l_2} \times \mu_2) = 2^{l_2} - 1 \tag{4.5}$$

which implies k is a factor of l_2. But l_2 is less than l. Therefore, it is not possible.

Hence, k cannot be a factor of 2^{l_1}. □

The next theorem shows all other cyclic component can be realized by taking multiple instances of the base case.

Theorem 4.2 : *The cycle structure $[1(1) + \mu(k)]$ where $\mu \times k + 1 = 2^{ml}$ where l is the smallest integer for which k is a factor of $2^l - 1$ and for some integer m is realized by the characteristic polynomial in elementary divisor form $f(x) = \phi(x)\phi(x) \cdots m - times$, where the cycle structure of $\phi(x) = [1(1) + \mu_1(k)]$ and $\mu \times k + 1 = 2^l$*

Proof : If $\phi(x) = [1(1) + \mu_1(k)]$ then
$f(x) = [1(1) + \mu_k(k)] \times [1(1) + \mu_k(k)] \cdots m - times$
that is the cyclic component of the cycle length k
$\tilde{\mu} = {}^mC_1\mu$, ${}^mC_2\mu^2 \cdot k$, $\cdots$, ${}^mC_m\mu^m \cdot k^{m-1}$
Therefore, simplifying
$\tilde{\mu} = \frac{1+{}^mC_1(\mu \cdot k) \cdots {}^mC_m(\mu \cdot k)^m}{k} - \frac{1}{k}$. That is, $\tilde{\mu} = \frac{(1+\mu \cdot k)^m - 1}{k}$
That is, $\tilde{\mu} = \frac{2^{ml}-1}{k} = \mu$.
Hence the proof. □

The third theorem shows that the polynomial underlying the base case is necessarily irreducible.

Theorem 4.3 : *The polynomial $\phi(x)$ which has cycle structure $[1(1)+\mu_k(k)]$ and $\mu_k \times k+1 = 2^l$ and l is the smallest number dividing $2^l - 1$ is irreducible.*

Proof : Let $\phi(x)$ be reducible and $\phi(x) = \phi_1(x) \cdot \phi_2(x)$.
Therefore, cycle structure of each $\phi_{1(2)}(x)$ has to be $[1(1), \mu_{1(2)}(k)]$.
That is $\mu_{1(2)} \times k = 2^{l_{1(2)}} - 1$, where $l_{1(2)} < l$. But this is not possible. Hence $\phi(x)$ is irreducible. □

Task 2 : The next important task is to output the Primary Cycle Structure(PCS) in triplet form; that is, $PFCS = (\mu_k, k)^{n_1}$,$(\mu_k, k)^{n_2} \cdots (\mu_k, k)^{n_\mathcal{N}}$
Hence Task 2 lies in fulfiling the following objectives

- Determining the value of $\mathcal{N}$.
- Determining the value of the powers (n_1, $n_2 \cdots n_\mathcal{N}$) of the underlying irreducible polynomial $\phi(x)$

In order to attain the objectives

- We find out the total number of j^{th} factor set - $\tilde{N}(F_j)$
- Consequently we find out the number of PCS required to accomodate these $\tilde{N}(F_j)$ number of factors and the power of each resultant PCS.

The relationship between the primary cycle family and the underlying elementary divisors has been discussed in *section 3.2.3* where we have shown that the cyclic component of a particular cycle length $(2^j \cdot k)$ depends upon the total number of j^{th} factor set $\tilde{N}(F_j)$ present in the Primary Cycle Structure set. $\tilde{N}(F_j)$ represents the sum of number of j^{th} factor set present in all elementary divisors - that is, PCS.

The *relation 3.19* illustrates the relationship. We are reproducing it for convenience.

$$\mu_{2^j \cdot k} = \frac{2^r \sum_{j=0}^{j} \tilde{N}(F_j) - 2^r \sum_{\mathcal{J}=0}^{j-1} \tilde{N}(F_j)}{2^j \cdot k}$$

The equation can be reoriented and we can derive $\tilde{N}(F_j)$ as

$$\tilde{N}(F_j) = \frac{1}{r} log_2(\mu_{2^j \cdot k} \times k \times 2^{(j - r \cdot \sum_{\mathcal{J}=0}^{j-1} \tilde{N}(F_j))} + 1) \tag{4.6}$$

where $\mu_{2^j \cdot k}$ is the given cyclic component from which the number of j^{th} factor set has to be derived, r is the degree of the polynomial $\phi(x)$.

The following example illustrates the above formulation.

Example 4.5 *Given* CS = [1(1),85(3), 3688(6), 5591040(12), 4581088288(24)] *in GF(2),* $r = 2$. *Therefore, from the* relation 4.6
$\tilde{N}(F_0) = \frac{1}{2} log_2(85 \times 3 \times 2^{0-0} + 1) = \frac{log_2(256)}{2} = 4$
$\tilde{N}(F_1) = \frac{1}{2} log_2(2688 \times 3 \times 2^{1-4\times 2} + 1) = \frac{log_2(64)}{2} = 3$
$\tilde{N}(F_2) = \frac{1}{2} log_2(5591040 \times 3 \times 2^{2-(4+3)\times 2} + 1) = \frac{log_2(4096)}{2} = 6$
$\tilde{N}(F_3) = \frac{1}{2} log_2(4581088288 \times 3 \times 2^{3-(4+3+6)\times 2} + 1) = \frac{log_2(16384)}{2} = 7$

$\tilde{N}(F_j)$ denotes the total number of j^{th} factor set present in all the PCS. The next step lies in determining the manner in which these j^{th} factor set are distributed in individual PCS. In this connection we define the term - $N(PCS(j))$.

Definition 4.1 $N(PCS(j))$ *is defined as the number of elementary divisors/PCS having the* j^{th} *factor set and satisfying the constraints that the first* $N(PCS(j)) - 1$ *has all the possible* 2^{j-1} *factors. (from* Relation 3.10*)*

Example 4.6 Let $f(x) = \phi(x)^1 \cdot \phi(x)^4 \cdot \phi(x)^7 \cdot \phi(x)^8$. Then for $j = 2$, the 2^{nd} factor set(*Definition 3.8*), that is the factors of values $3(2^{3-1}+1)$ & $4(2^3)$ are present in 2 elementary divisors [$\phi(x)^7$, $\phi(x)^8$]. The last elementary divisor have all the possible 2^{3-1} factors namely 5, 6, 7 & 8, while the last have only the 3 factors. Hence $N(PCS(j)) = 2$. □

Following lemma formalizes the above discussion

Lemma 4.4 *The number of Primary Cycle Structure $N(PCS(j))$ required to accommodate $\tilde{N}(F_j)$ is given by the formula*

$$N(PCS(j)) = \begin{cases} \lceil \frac{\tilde{N}(F_j)}{2^{j-1}} \rceil & j \geq 1 \\ \tilde{N}(F_j) & j = 0 \end{cases} \tag{4.7}$$

Proof : Since in an elementary divisor, the maximum number of the j^{th} factor $= 2^{\mathcal{J}-1}$. Therefore, the number of elementary divisor to accommodate $N(F_{\mathcal{J}})$ is given by *Relation 4.7.* $N(F_0)$ represents the irreducible polynomial which each elementary divisor can have only one. □

The following example illustrates the concept, as well as lays the foundation for the next step.

Example 4.7 *Given $\tilde{N}(F_0) = 4$, $\tilde{N}(F_1) = 3$, $\tilde{N}(F_2) = 6$, $\tilde{N}(F_3) = 7$. Find $N(PCS(j))$ for each value of j. To evaluate we have to make use of* relation 4.7.

$N(PCS(0)) = 4$, that is there are 4 irreducible polynomial. Hence number of $PCS = 4$.
$N(PCS(1)) = 3$, that is 3 PCS out of 4 has factors ranging from $2^{1-1}+1$ to 2^1.
$N(PCS(2)) = 3$, that is 3 PCS has factors ranging from $2^{2-1}+1$ to 2^2.
$N(PCS(3)) = 2$, that is 2 PCS has factors ranging from $2^{3-1}+1$ to 2^3.

On calculation of N(PCS(j)), we know the constituent factors of each PCS. Therefore, the next task is enumerating the power $(n_1, n_2, \cdots n_{\mathcal{N}})$ of each elementary divisor/PCS by compiling from the obtained factor information.

The next theorem illustrated the method.

Lemma 4.5 *The number of PCS (say $\mathcal{N}_j$) which will have power P_j where $2^{j-1}+1 \leq P_j \leq 2^j$ is given by the equation*

$$\mathcal{N}_j = N(PCS(j)) - N(PCS(j-1)). \tag{4.8}$$

The exact value of P_j are given by the formula

$$P_j = \begin{cases} 2^j & \text{for } (\mathcal{N}_j \text{ -1) } PCS \\ 2^{j-1} + L_j & \text{for } \mathcal{N}_j^{th} \ PCS \end{cases} \tag{4.9}$$

where

$$L_j = \begin{cases} N(F_j) \ mod \ 2^{j-1} & j > 0 \\ 2^{j-1} & j = 0 \end{cases}$$

Proof : The value of $\mathcal{N}_j$ is determined by the fact that there is no higher factor set in those elementary divisors. The *Relation 4.8* determines that.

From *Lemma 4.4*, all the $(\mathcal{N}_j - 1)^{th}$ elementary divisors are fully filled with the $\mathcal{J}^{th}$ factor while in the last elementary divisor, the remaining factors are allotted. Hence the value of P_j is accordingly determined. □

The following example summarizes the above discussions on generation of PCS(Primary Cycle Structure) in triplet form. *Example 4.5 & 4.7* are reproduced for convenience.

Example 4.8 *Given* CS = [1(1),85(3), 3688(6), 5591040(12), 4581088288(24)] *The* Table 4.1 *shows step by step calculation of each of the parameters to arrive at the final result.*

Table 4.1: Calculation of Primary Cycle Structure

j	$N(F_j)$	$N(PCS(j))$	$\mathcal{N}_j$	P_j
0	*4*	*4*	*1*	*1*
1	*3*	*3*	*0*	*0*
2	*6*	*3*	*1*	*4*
3	*7*	*2*	*2*	*7, 8*

We are elaborating the method of obtaining $\mathcal{N}_3$ *&* P_3 *to explain* Lemma 4.4.
Since N(PCS(3)) is 2 & N(PCS(4)) =0, (as there is no N(PCS(4))) therefore, there will be 2 [N(PCS(3)) - N(PCS(4))] PCS whose power will be between $2^{3-1}+1$ *to* 2^3.
Out of which the one PCS will have power P_3 *=8(*2^3*) from* Relation 4.9, *while the second one will have power* $2^{3-1} + N(F_3)$ *mod* $8 = 2^{3-1} + 7$ *mod* $4 = 7$.
Therefore, the final output of Primary Cycle Structure in Triplet Form is
$CS = (1,3)^1 \times (1,3)^4 \times (1,3)^7 \times (1,3)^8$.

We sum up the discussions providing the algorithm to convert a Primary Family Cycle Structure($PFCS$) into Primary Cycle Structure(PCS)

Algorithm 4.4 *Enum_PCS_from_PFCS* $(PFCS, [PCS])$
{
Input : *Primary Family Cycle Structure(PFCS)*
Output : *Primary Cycle Structure Set* PCS_1, $PCS_2 \cdots PCS_{\mathcal{N}}$: $[PCS]$ *in triplet form.*
Find μ_k *through* Axiom 4.1
For each j
{
Find $N(F_j)$. - Relation 4.6
Find $N(PCS(j))$ - Relation 4.7
$\mathcal{N}_{j-1} = N(PCS(j-1))$ - $N(PCS(j))$ - Relation 4.8
Find P_{j-1} - Relation 4.9
for each $N(PCS(j))^{th}$ PCS
$[PCS] \leftarrow PCS_i$
}

$\mathcal{N}_{j-1} = N(PCS(j-1))$ - Relation 4.8
Find P_{j-1} - Relation 4.9
$[PCS] \leftarrow PCS_i \cdots PCS_{\mathcal{N}}$
}

Complexity Analysis

- Finding μ_k - The value of $\mu_k \times k$ can be at most be 2^n -1 where n is the number of cells in LCA. So we have to do at most n comparisons.
- Each of the iterative steps are accomplished in constant time.
- *Number of iteration :* The highest value of j can be when the $PFCS$ constitutes of a single elementary divisor $\phi(x)^{n_i}$ - that is all the factor set is generated by a single elementary divisor and the value of r and p are both 1. In this case, the highest factor set is $\lceil log_2(n) \rceil$(follows directly from *Lemma 3.2*). Therefore, the number of iterative steps(j) can be at most $log_2(n)$.
- The overall complexity of the algorithm is thus $O(n)$

Once we have obtained the Primary Cycle Structure our next task lies in deriving the polynomials underlying them

4.1.3 Conversion of Primary Cycle Structure(PCS) into Elementary Divisors(EDs)

An elementary divisor $\phi_i(x)^{n_i}$, as noted in *Definition 3.9*, has the PCS in triplet form - $(\mu_k, k)^{n_i}$ where the irreducible polynomial $\phi_i(x)$ can produce the cycle structure $[1(1), \mu_k(k)]$.

This subsection reports the process of generation of the polynomials from the given cycle structure $\mu_k(k)$ where $\mu_k \times k = 2^r - 1$. Database of primitive polynomial for all values of r in GF(2) has to be maintained in this process.

A primitive polynomial is a special type of irreducible polynomial. The cycle structure of a primitive polynomial of degree r is $CS = [1(1), 1(2^r - 1)]$, whereas an irreducible polynomial of degree r has $CS = [1(1), \mu_k(k)]$ where $\mu_k \times k = 2^r - 1$ [22]. For example, for r= 6, CS of a primitive polynomial is CS = [1(1), 1(63)], whereas an irreducible polynomial can have CS = [1(1), 7(9)].

The following theorem illustrates the method of calculating the irreducible polynomial of degree r from an r degree primitive polynomial. The primitive polynomial can be represented by a linear operator T with the help of *Rational Canonical Form* [22].

Theorem 4.6 *If T is the characteristic matrix of a primitive polynomial having cycle structure $CS = [1(1), 1(\tilde{k})]$, then the characteristic polynomial of T^{μ} has the cycle structure $[1(1), \mu_k(k)]$ where $\mu_k \times k = \tilde{k}$.*

Proof : If the cycle structure of T is [1(1), $(1(\tilde{k})$], then $T^{\tilde{k}} = \mathrm{I}$, $T^{\mu_k \times k} = I$.

Therefore, the matrix T^{μ_k} will have cycle length of k or sub-multiple of k. Let k_1 be a factor of k and for some state x, $(T^{\mu_k})^{k_1} \cdot x = x$. Therefore, $T^{(\mu_k \times k_1)} \cdot x = x$. But this is not possible.

Hence T^{μ_k} has cycle structure [1(1), $\mu_k(k)$]. Consequently, characteristic polynomial $f(x) = det(T^{\mu_k} + Ix)$ has cycle structure [1(1), $\mu_k(k)$]. □

The complete algorithm is now presented.

Algorithm 4.5 *Enum_ED_from_PCS(PCS, $\phi_i(x)^{n_i}$).*
Input : *Primary Cycle Structure in triplet form* $[\mu_k, k, n_i]$
Output : *Elementary Divisor* $\phi_i(x)^{n_i}$
Step 1 : *Calculate* $r = log_2(\mu_k \times k + 1)$
Step 2 : *Choose primitive polynomial of degree* $r \times p$ *in GF*(2).
Step 3 : *Construct T matrix from polynomial* $f(x)$ *in GF*(2)*[29].*
Step 4 : *Find* T^{μ_k}.
Step 5 : *Find characteristic polynomial* $\phi_i(x) = det(T^{\mu_k} + Ix)$
Step 6 : *Output* $\phi_i(x)^{n_i}$.

Complexity Analysis

- To maintain the database of degree N, we have a space complexity of $\mathrm{O}(N^2)$, each polynomial of degree n requiring at most n^2 places.
- *Step 4 :* A matrix multiplication is of order n^3. The largest value of μ_k can be 2^n. Therefore, multiplication of T^{μ_k} will take n^4 unit time.
- *Step 5 :* The determinant calculation takes $\mathrm{O}(n^3)$ time.
- Thus the algorithm has a time complexity of $O(n^4)$ while a space complexity of $O(N^2)$.

The following example illustrates the algorithmic steps.

Example 4.9 *To generate the elementary divisors for* $PCS = (3,5)^2$*, that is the coefficients of the elementary divisor are in GF*(2^2).
Step 1: Find $r = (log_2(\mu_k \times k + 1))$. *(*$r = 4$*)*
Step 2: Find a primitive polynomial of degree 4 in GF(2). ($x^4 + x + 1$)
Step 3: Construct the T matrix from the polynomial using Rational Canonical Form.

$$T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 4: Calculate T^3.

$$T^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Step 5: Calculate the characteristic polynomial of T^{μ_k} *to obtain* $\phi(x)$.
$\phi(x) = det[T^3 + Ix] = (x^4 + x^3 + x^2 + x + 1)$. *Therefore, the elementary divisor* $= (x^4 + x^3 + x^2 + x + 1)$.

Once we have obtained the elementary divisors we can proceed to generate the linear operator T of the LCA. As we have already mentioned, this step is done specifically for a three neighborhood LCA.

4.1.4 LCA Synthesis from Elementary Divisor

The generation of CA from Elementary Divisor is accomplished by the following operations.
Operation 1 : Convert each elementary divisor - $\phi_i(x)^{n_i}$ into LCA_i.
Operation 2 : Merge each LCA_i to form the complete LCA.

Illustrative examples and formal algorithms follow the explanation of the operations.

Operation 1: Under the assumption of 3-neighborhood dependency of each cell of LCA, the elementary divisor, $\phi_i(x)$ is converted into a tridiagonal matrix. The elementary divisor may either represent a cyclic subspace or it can also represent an acyclic subspace of depth(d) of the machine where the corresponding irreducible polynomial is x^d. We discuss both the cases.

Case 1: *Elementary Divisor Representing a Cycle Structure.* To achieve this target, we first convert the underlying irreducible polynomial $\phi(x)$ into 3-neighborhood LCA-Matrix. The algorithm proposed by Kattel et.al. in [4] is executed to obtain the matrix. The formation of LCA with characteristic and minimal polynomial as $\phi_i(x)^{n_i}$ is executed using the principle of the following theorem [22].

Theorem 4.7 *Let* $d_{n-1}(x)$ *be the greatest common divisor of all minors of order* $(n-1)$ *in the matrix* $(T + Ix)$ *of order* n, *and let* $\phi(x)$ *be the characteristic polynomial of* T. *Then the minimal polynomial* $M(x)$ *of* T *is given by*

$$M(x) = \phi(x)/d_{n-1}(x) \tag{4.10}$$

We now present the theorem with the help of which the matrix $T(\phi_i(x)^{n_i})$ corresponding to $\phi_i(x)^{n_i}$ is formed.

Theorem 4.8 *Given a tridiagonal matrix* $T(\phi_i(x))$ *whose characteristic polynomial is* $\phi_i(x)$, *then the matrix* $T(\phi_i(x)^{n_i})$ *where*

$$T(\phi_i(x)^{n_i}) \;=\; \begin{bmatrix} T(\phi_i(x)) & 1 & 0 & \cdots \\ 0 & T(\phi_i(x)) & 1 & 0\cdots \\ \cdot & & & \\ \cdot & & & \\ 0 & \cdots & 0 & T(\phi_i(x)) \end{bmatrix}$$

has both characteristic and minimal polynomial as $\phi_i(x)^{n_i}$.

Proof : The proof is presented for n_i =2. The proof can be easily generalized for any value of n_i. We first find the characteristic polynomial of T where

$$T(\phi_i(x))^2 \;=\; \begin{bmatrix} T(\phi_i(x)) & 1 \\ 0 & T(\phi_i(x)) \end{bmatrix}$$

Therefore, the characteristic polynomial will be

$$\det \begin{bmatrix} T(\phi_i(x)) + Ix & 1 \\ 0 & T(\phi_i(x)) + Ix \end{bmatrix} = \phi(x) \cdot \phi(x) - 1 \cdot 0 = \phi(x)^2$$

The minimal polynomial is found with the help of *Theorem 4.7*. To have the irreducible polynomial represented by a 3-neighborhood LCA, the super-diagonal and sub-diagonal has to be non-zero[4]. Therefore, placing one in the junction gives us a non-zero minor. Through suitable row, column operation [22], the value of minor (d_i) will be one. Therefore, *gcd* of all minors

$d(x) = gcd(d_1(x), \cdots, d_n(x)) = 1$ since $d_i = 1$.

Hence minimal polynomial $= \frac{\phi(x)^2}{1} = \phi(x)^2$. □

Example 4.10 *Given an elementary divisor* $(x^2+x+1)^2$*, we have to synthesize a* T *matrix whose characteristic and minimal polynomial is* $(x^2 + x + 1)^2$.

Given $T(x^2 + x + 1) = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, *then* $T(x^2 + x + 1)^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Case 2 : *Building* T *matrix generating depth* d *:* The elementary divisor generating depth d is given by x^d (follows from *relation 3.5*). The algorithm to synthesize an LCA whose characteristic polynomial is x^d is reported next.

The algorithm is presented below.

Algorithm 4.6 *Build_D_Matrix*(d, D)
Input : d *: Depth*
Output : D *matrix with characteristic polynomial* x^d.
Initialize the $d \times d$ *matrix(D) with all* $0's$ *elements.*
Assign $D[i][i-1] \;=\; 1$ *where* $i = 2, \cdots, d$.
Output D.

Example 4.11 *Let depth = 3. Then depth matrix produced through the algorithm is*

$$D = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Once all the LCA Matrices for all the individual elementary divisor are obtained, the next task lies in combining them to form the final matrix through *Operation 2.*

Operation 2 : This operation involves the construction of the final LCA having characteristic polynomial $\phi(x)$, where $\phi(x) = \phi_1(x)^{n_1} \cdots \phi_i(x)^{n_i} \cdots \phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}}$. The LCA for the characteristic polynomial $\phi(x)$ can be easily formed by the following theorem.

Theorem 4.9 *If the characteristic matrices $T(\phi_i(x)^{n_i})$ with characteristic polynomial $\phi_i(x)^{n_i}$ is arranged in block diagonal form, then the resultant matrix T has characteristic polynomial $\phi(x)$ where*

$$T = \begin{bmatrix} T(\phi_i(x)^{n_1}) & 0 & \cdots \\ 0 & T(\phi_2(x)^{n_2}) & 0 \cdots \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ 0 & \cdots & T(\phi_{\mathcal{N}}(x)^{n_{\mathcal{N}}}) \end{bmatrix}$$

Example 4.12 *Let the characteristic polynomial $\phi(x) = x^3 \cdot (x+1)^2 \cdot (x^2+x+1)^2$. The $T-$matrix becomes*

$$T = \begin{bmatrix} \lceil 0 & 0 & 0 \rceil & 0 & 0 & 0 & 0 & 0 & 0 \\ \lfloor 1 & 0 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 & 0 \\ \lfloor 0 & 1 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lceil 1 & 1 \rceil & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lfloor 0 & 1 \rfloor & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lceil 1 & 1 & 0 & 0 \rceil \\ 0 & 0 & 0 & 0 & 0 & \lfloor 1 & 0 & 1 & 0 \rfloor \\ 0 & 0 & 0 & 0 & 0 & \lfloor 0 & 0 & 1 & 1 \rfloor \\ 0 & 0 & 0 & 0 & 0 & \lfloor 0 & 0 & 1 & 0 \rfloor \end{bmatrix}$$

We present the final algorithm for generation of T matrix.

Algorithm 4.7 *Enum_LCA_from_ED($[\phi_i(x)^{n_i}]$, d, T)*
Input : $[\phi_i(x)^{n_i}]$ - set of elementary divisor, d - Depth of LCA
Output : T matrix.
Build T_i matrix for each individual elementary divisor $\phi_i(x)^{n_i}$.
Build_D_matrix(d, D)
Form the final Matrix combining all T_i matrix and D Matrix in Block Diagonal Form.

Complexity Analysis : The algorithm to build T matrix from an irreducible polynomial $\phi(x)$, is non-deterministic. Each iteration of the algorithm takes $O(n^3)$ time [24]. Extensive

experimentation shows that in almost all the cases T matrix is obtained after a very few iterations. Consider, $\mathcal{M}$ number of iteration required to form the T. Therefore required time is $O(\mathcal{M} \cdot n^3)$. The typical value of $\mathcal{M} = 2$ or 3. Hence for all practical purpose the complexity is $O(n^3)$.

Overall complexity of the algorithm - *Enum_LCA_from_CS_depth*

- Theoretically, the algorithm has a super-polynomial complexity. The complexity has been imparted by *Step A* & *Step C* respectively.
- However, for all practical purpose, for synthesizing LCA of size upto 5000, the *Step A* has a complexity of $O(n^4)$.
- With certain space complexity of $O(N^2)$, the *Step C* also reduces to $O(n^4)$.
- Hence, the complexity of the entire algorithm is $O(n^4)$ with a space complexity of $O(N^2)$ where N is the degree upto which we want to operate.

4.2 Synthesis of ACA

The analysis of ACA has been detailed in *Section 3.3* of *Chapter 3*. This analysis provides the synthesis scheme discussed next. The synthesis scheme accepts a given legal additive cyclic(CS') and non-cyclic subspace (d) as input and outputs an ACA generating the vector space. In the analysis we have shown that the cycle structure of an ACA follows a definite relationship with its corresponding LCA. The basic approach of synthesis outlined below, utilizes the explored relationship between LCA & ACA.

- To synthesize the linear operator T, at the first stage, the *Primary Cycle Structure(PCS)* is derived from the additive cycle structure (CS'). The *Additive Primary Family Cycle Structure*($PFCS'$) acts as an intermediate step in this derivation process.
- Once PCS is formed, the synthesis of T follows the same methodology of LCA.
- The inversion vector F is synthesized after the synthesis of T.

The major steps for the synthesis of ACA are noted in the algorithm reported below. *Steps A - D* refers to synthesis of T while the F vector is synthesized in *Step* E.

Algorithm 4.8 *Enum_ACA_from_CS&depth*(CS', d, T, F)
Input : *Cycle Structure* (CS'), *depth*(d)
Output : ACA - (T *matrix*) & (F *vector*)
A : Generate each individual Additive Primary Family Cycle Structure ($PFCS'$) *from* CS'.
B : Generate Primary Cycle Structures in triplet form from each $PFCS'$.
C & D : The Steps C & D are same as that of LCA
E : The inversion vector F *is synthesized.*

The following example illustrates the sequential steps of the synthesis algorithm. It accepts the analysis results obtained in *Example 3.1* and performs the reverse operation of Synthesis on this input to get back the original LCA.

Example 4.13 *Let* CS' = [16(4), 24(8), 1397760(12), 3072(20), 44039680(24), 4608(40), 71302144(60), 2246048256(120)]
The value of minimum additive factor - $\mathcal{M} = 2$.
Step A: *Separating each primary cycle family we obtained* CS' = [64[4], 24(8)] × [1(1), 4095[12], 2560(24)] × [1(1), 15[20]] ×[1(1), 15[60]]
Step B: *From each* $PFCS'$ *we obtain PCS in triplet form* $CS = (1,1)^2 \times (1,1)^5 \times (1,3)^2 \times (1,3)^6 \times (3,5)^1 \times (1,15)^1$
Step C & D: *After performing Steps C & D, the linear operator* T *is obtained.*
Step E : *Corresponding to* T, *the inversion vector* F *where*
$F = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^t_{17\times 1}$ *is synthesized.*

The following important aspects arising from the above example are noted below.

- The cycle structure follows the criteria of Legal Additive Cycle Structure where the *minimum additive factor*($\mathcal{M}$) is 2.
- Secondly, the prerequisite of formation of additive Cycle Structure (CS') - the existence of the factor $(x+1)$, that is the presence of cycle family $k = 1$ is satisfied.

From the example it is clear, that to synthesize the ACA from the given CS' & depth, we proceed in a step by step fashion of segregrating each participating primary cycle. In this process we define the terms $PFCS'$ and redifine the term CS' in broader perspective.

While forming Additive Cycle Structure, the 1-cycle family encompasses the all-zero state (1(1) separately written in *relation 3.16* during definition of Cycle Structure of LCA) Thats why when we form $PFCS'$, it is of the form

$$PFCS'(k=1) = [S_{2^{\mathcal{M}}\cdot k_1}[2^{\mathcal{M}} \cdot k_1], \sum_{j=\mathcal{M}+1}^{m_{k_1}} \mu_{2^j \cdot k_1}(2^j \cdot k_1)] \tag{4.11}$$

$$PFCS'(k>1) = [1(1),\ S_{2^{\mathcal{M}}\cdot k_i}[2^{\mathcal{M}} \cdot k_i], \sum_{j=\mathcal{M}+1}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)] \tag{4.12}$$

Similarly CS' - the remaining additive cycle structure i to be resolved at any dynamic step is termed as

$$CS' = [\sum_{k_i}(S_{2^{\mathcal{M}}\cdot k_1}[2^{\mathcal{M}} \cdot k_1], \sum_{j=\mathcal{M}+1}^{m_{k_1}} \mu_{2^j \cdot k_1}(2^j \cdot k_1))], \quad if\ k_1 \in k_i \tag{4.13}$$

$$CS' = [1(1), \sum_{k_i}(S_{2^{\mathcal{M}}\cdot k_i}[2^{\mathcal{M}} \cdot k_i], \sum_{j=\mathcal{M}+1}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i))], k_1 \notin k \tag{4.14}$$

The new notation $S_{2^{\mathcal{M}}.k_i}[2^{\mathcal{M}} \cdot k_i]$ is termed as Cycle Information(CI). The definition of CI is noted below.

Definition 4.2 *Cycle Information(CI) represents the information of a cycle length ($\leq 2^{\mathcal{M}} \cdot k$) of a primary cycle family. It has the notation $S_{2^{\mathcal{M}}.k}[2^{\mathcal{M}} \cdot k]$ where $S_{2^{\mathcal{M}}.k}$ indicates the states covered by cycle length starting from k and $\leq 2^{\mathcal{M}} \cdot k$.*

Steps A, B, E are next discussed in detail.

4.2.1 A. Generation of $PFCS'$ from Cycle Structure(CS')

The step is exactly similar to the step following in case of LCA for generating $PFCS$ from CS in *Section 4.1.1*. Only in this case the Cycle Information, that is the information regarding the number of states covered by cycles $\leq 2^{\mathcal{M}} \cdot k_i$ is generated differently. From *Theorem 3.13*, it is quite clear that the cyclic components of cycle length ($> (2^{\mathcal{M}} \cdot k_i) - \forall k_i$) are same in CS and CS' and hence are derived in same process as LCA. Therefore in our following discussion we concentrate of generating Cycle Information of $\leq (2^{\mathcal{M}} \cdot k_i) - \forall k_i$.

The conversion tool accepts the given additive cycle structure(CS') and partitions it into corresponding $PFCS$s such that CS' follows the relation

$$CS' = [PFCS'_1] \times [PFCS'_2] \times \cdots \times [PFCS'_{\tilde{\mathcal{N}}}] \tag{4.15}$$

In order to separate each primary family cycle structure, the same recursive technique as LCA of dividing $CS' = CS'_1 \times CS'_2$ is followed. Here CS_1 is the $PFCS$ with the smallest primary cycle and $CS'_2 = \{CS' - \{CS'_1, CS'_2 \times CS'_1\}\}$.

We present the algorithm for converting CS' to $PFCS'$ and the algorithm to generate CS'_1 and CS'_2 from CS'. The two algorithms are exactly similar to *Algorithm 4.3 & 4.2* respectively.

Algorithm 4.9 *Enum_PFCS'_from_CS'* (CS,[$PFCS'$])
{
Input : CS' - Cycle Structure
Output : Primary Family Cycle Structure Set $PFCS'_1$, $PFCS'_2 \cdots PFCS'\tilde{\mathcal{N}}$: [$PFCS'$]
Enum_CS'_1&CS'_2_from_CS'(CS', CS'_1, CS'_2)(Algorithm 4.10)
$PFCS'_i = CS'_1$
$[PFCS'] \leftarrow PFCS'_i$
If $(CS'_2 = PFCS')$
{
$[PFCS'] \leftarrow PFCS'_{\mathcal{N}}$
Exit
}
else
$CS' = CS'_2$
Enum_PFCS'_from_CS'(CS',[$PFCS'$])
}

The Algorithm *Enum_CS'$_1$&CS'$_2$_from_CS'*(CS', CS'_1, CS'_2) differs from its linear counterpart *Enum_CS$_1$&CS$_2$_from_CS'*(CS, CS_1, CS_2) (*Algorithm 4.2*) only in the enumeration of Cycle Information. The lines where it is differing is maked in a box.

Algorithm 4.10 *Enum_CS'$_1$&CS'$_2$_from_CS'*(CS', CS'_1, CS'_2)
Input : $CS' = [1(1), \sum_{k_i} \sum \mu_{2^j \cdot k_i}(2^j \cdot k_i)]$
Output : CS'_1, CS'_2
$CS'_1 = [1(1), \sum \mu_{2^j \cdot k_1}(2^j \cdot k_1)]$, *where* k_1 *is the smallest member of* k_i *- cycle family set.*
$CS' = CS' - \sum \mu_{2^j \cdot k_1}(2^j \cdot k_1)$.
for each $k_i \in CS'_i$ *in ascending order*
$k_2 = k_i$
if k_1 *is a factor of* k_2
{
Obtain $S_{2^{\mathcal{M}} \cdot k_2}(CS'_2)$*from relation 4.19*
$CS'_2 \leftarrow CS'_2 + S_{2^{\mathcal{M}} \cdot k_2}(CS'_2)$
$CS' \leftarrow CS'$ - $\tilde{S}_{2^{\mathcal{M}} \cdot k_2}(CS'_2)$
For $j > \mathcal{M}$
Obtain $\mu_{2^j \cdot k_2}(CS'_2)$ *from relation 4.2*
$CS'_2 \leftarrow CS'_2 + \mu_{2^j \cdot k_2}$
$CS' \leftarrow CS'$ - $\tilde{\mu}_{2^j \cdot k_2}$
}
else
{
$CS'_2 \leftarrow CS'_2 + \tilde{S}_{2^j \cdot k_2}$
Obtain $\tilde{S}_{2^j \cdot k_3}$ *from relation 4.18*$(k_3 = lcm(k_1, k_2))$
$CS' \leftarrow CS' - \tilde{S}_{2^j \cdot k_3}$
$CS' \leftarrow CS' - \tilde{S}_{2^j \cdot k_2}$
For $j > \mathcal{M}$
$CS'_2 \leftarrow CS'_2 + \tilde{\mu}_{2^j \cdot k_2}$
Obtain $\tilde{\mu}_{2^j \cdot k_3}$ *from relation 3.23*$(k_3 = lcm(k_1, k_2))$
$CS' \leftarrow CS' - \tilde{\mu}_{2^j \cdot k_3}$
$CS' \leftarrow CS' - \tilde{\mu}_{2^j \cdot k_2}$
/ Both the* k_2 *cycle family and the impact it creates when cross producted with* k_1 *is removed from CS' */*
}

The method of deriving *relation 4.18* & *4.19* is elaborated next. To derive the relations, we analyse the nature of Cycle Information(CI) produces when two cycle family k_1 & k_2 are cross producted.

Analysis : The *Theorem* for deriving the Cycle Information(CI) of cycle family k_3 from cross product of $k_1 \in CS'_1$ and $k_2 \in CS'_2$ is discussed next. The Theorem is in line with

Theorem 3.9 . The CS'_1 and CS'_2 are defined below

$$CS'_1 = \begin{cases} S_{2^{\mathcal{M}}\cdot k_1}[2^{\mathcal{M}} \cdot k_1], \sum_{j=\mathcal{M}+1}^{m_{k_1}} \mu_{2^j \cdot k_1}(2^j \cdot k_1) & k_1 = 1 \\ 1(1), S_{2^{\mathcal{M}}\cdot k_1}[2^{\mathcal{M}} \cdot k_1], \sum_{j=\mathcal{M}+1}^{m_{k_1}} \mu_{2^j \cdot k_1}(2^j \cdot k_1) & k > 1 \end{cases}$$

$$CS'_2 = [1(1), \sum_{k_i} S_{2^{\mathcal{M}}\cdot k_i}[2^{\mathcal{M}} \cdot k_i], \sum_{j=\mathcal{M}+1}^{m_{k_i}} \mu_{2^j \cdot k_i}(2^j \cdot k_i)].$$

We study the impact created on Cycle Information of k_3 cycle family. The impact created on cyclic components of cycles $2^j \cdot k_3$ $j > \mathcal{M}$ is elaborated in *Theorem 3.9*. The impact is characterized in the following theorem.

Theorem 4.10 *The Cyclic Information(CI) of k_3-cycle family is formed from the cross product of cyclic component of k_1 & k_2-cycle family in CS_1 and CS_2 respectively. The following relation elaborates the nature of formation*

$$S_{2^{\mathcal{M}}\cdot k_3} = S_{2^{\mathcal{M}}\cdot k_1} \times S_{2^{\mathcal{M}}\cdot k_2} \tag{4.16}$$

Proof : Proof analogous to proof of *Theorem 3.9* . □

As in LCA, the value of k_3 can be 1). lcm(k_1,k_2) > k_2. and 2). lcm(k_1,k_2) = k_2.

In the special case, $k_3 = k_2$, the number of states covered by cycles $\leq (2^{\mathcal{M}} \cdot k_2)$ - $S_{2^{\mathcal{M}}\cdot k_2}$ in CS'_2 has changed to $\tilde{S}_{2^{\mathcal{M}}\cdot k_2}$ in CS'. The change is characterized by the following relations.

$$\tilde{S}_{2^{\mathcal{M}}\cdot k_2}(CS') = [S_{2^{\mathcal{M}}\cdot k_1} + 1](CS'_1) \times S_{2^{\mathcal{M}}\cdot k_2}(CS'_2) \tag{4.17}$$

If lcm(k_1,k_2) > k_2 ,

$$\tilde{S}_{2^{\mathcal{M}}\cdot k_3}(CS') = S_{2^{\mathcal{M}}\cdot k_3}(\textit{as derived in relation } 4.16) \tag{4.18}$$

Synthesis : During the generation of CS'_2 from CS, when a cycle family k_2 is encountered, we check the nature of k_2

- *If $k_3 = lcm(k_1, k_2) > k_2$*, then the entire cycle family k_2 is added from CS' to CS'_2 and the resultant Cycle Information $\tilde{S}_{2^{\mathcal{M}}\cdot k_3}$ (obtained from *relation 4.18*) of k_3 cycle family is subtracted from CS'.
- *If $k_3 = lcm(k_1, k_2) = k_2$*, then the Cyclic Information which will be added from CS' to CS'_2 is determined from *relation 4.17*

Synthesizing back, the Cycle Information $S_{2\mathcal{M}.k_2}$ of CS'_2 the *relation 4.17* is reoriented and we get the following relation.

$$S_{2\mathcal{M}.k_2}(CS'_2) = \frac{\tilde{S}_{2\mathcal{M}.k_2}(CS')}{[S_{2\mathcal{M}.k_1} + 1](CS'_1)} \tag{4.19}$$

An example is given to illustrate the whole methodology.

Example 4.14 *Given* CS' = [1(1), 4095[12], 2560(24), 15[20], 1044465[60], 130560(120)] & *minimum additive factor -* $\mathcal{M} = 2$.
CS_1 =[1(1), 4095[12], 2560(24)].
The cycle structure of CS' *after* CS'_1 *is subtracted from* CS' *is* CS' =[1(1), 15[20], 1044465[60], 130560(120)]
The remaining primary cycles in CS' k_i = {5,15}
Taking the first cycle $k_2 = 5$,
k_1 **is not divisible by** k_2.
Therefore CS'_2 = [1(1), 15[20]], *that is the entire cycle structure arising from primary cycle 5 becomes member of* CS'_2.
The cycle family arising as a cross product of k_1 & k_2 *is* $k_3 = lcm(k_1, k_2) = 15$.
Therefore, from relation 4.16 - $S_{60} = S_{12} \times S_{20} = 61425[60]$
While $\mu_{8\times15} = \frac{\mu_{8\times3}\times S_5}{5}$ = 7680 - relation 3.23
Both the cycles are subtracted from CS'.
Therefore, the remaining cycle structure CS' = [1(1),983040[60], 122880(120)]
Here $k_2 = 15$ **which is divisible by** k_1.
Therefore, following relation 4.19, $S_{60} = \frac{983040}{4096} = 240[60]$
While $\mu_{8.15} = 0$. -relation 4.2
Hence CS'_2 = [1(1), 15[20], 240[60]]

Complexity Analysis of Algorithm 4.9 and 4.10 : *Algorithm 4.10* differs from *Algorithm 4.2* only in the derivation of Cycle Information (CI) which is accomplished in constant time (enumeration of *relation 4.18* & *4.19*). Therefore, the complexity of the *algorithm 4.10* is same as *algorithm 4.2*.

The *algorithm 4.9* takes the same number of steps as *algorithm 4.3*. Hence, its complexity is same as *algorithm 4.3*.

We now present the method of converting CS' into regular Primary Cycle Structure(PCS).

4.2.2 Generation of Primary Cycle Structure (PCS) from $PFCS'$

The methodology adopted to generate PCS from $PFCS'$ is largely similar to the one elaborated in *Section 4.1.2*. The only difference in step is the enumeration of the number of j^{th} factor set - $\tilde{N}(F_j)$ from the Cycle Information part of the $PFCS'$. To elaborate the similarity we present the algorithm to generate PCS set from $PFCS'$. The difference between the algorithm and that of *Algorithm 4.4* is pointed out.

Algorithm 4.11 *Enum_PCS_from_PFCS'*$(PFCS, [PCS])$
{
Input : *Additive Primary Family Cycle Structure(PFCS')*
Output : *Primary Cycle Structure Set* PCS_1, $PCS_2 \cdots PCS_{\mathcal{N}}$ *: [PCS] in triplet form.*
Find μ_k *through* Axiom 4.1
For each j
{
If $(j \leq \mathcal{M})$
Find $N(F_j)$ *from Cycle Information(CI)*
else
Find $N(F_j)$ $j > \mathcal{M}$- *Relation 4.6*
Find $N(PCS(j))$ - *Relation 4.7*
$\mathcal{N}_{j-1} = N(PCS(j-1))$ - $N(PCS(j))$ - *Relation 4.8*
Find P_{j-1} - *Relation 4.9*
for each $N(PCS(j))^{th}$ *PCS*
$[PCS] \leftarrow PCS_i$
}
$\mathcal{N}_{j-1} = N(PCS(j-1))$ - *Relation 4.8*
Find P_{j-1} - Relation 4.9
$[PCS] \leftarrow PCS_{\mathcal{N}}$
}

Therefore, for conversion of $PFCS'$ to PCS, the extra explanation is needed for the methodology for conversion of Cycle Information to the j^{th} factor set which is reported next. To explain the process, we bring forward the concept of elementary divisor - the base polynomials responsible for the formation of cycle structure. While designing the methodology for conversion, we base it upon the following salient features explained in *Chapter 3*.

- Additive Cycle Structure(CS') has arisen as a result of presence of an elementary divisor $(x+1)^{n_i}$, where $\mathcal{M} = \lfloor log_2(n_i) \rfloor + 1$, that is the maximum and minimum value of n_i is given by the following relation.

$$max(n_i) = 2^{\mathcal{M}} - 1 \quad \& \quad min(n_i) = 2^{\mathcal{M}-1} \tag{4.20}$$

- Additive Cycle Structure arising from elementary divisors other than $(x+1)s$ has their resultant cycles ($\leq 2^{\mathcal{M}} \cdot k_i$) merged in ($2^{\mathcal{M}} \cdot k_i$) and the components have been accordingly adjusted.

- The factors of elementary divisors which are responsible for the change in cycle structure from CS to CS' are $\leq 2^{\mathcal{M}}$. As it is noted in *Theorem 3.7*, the higher factors of elementary divisors doesn't affect the cycle structures of lower factors.

From the above salient features the enumeration of the j^{th} ($j \leq \mathcal{M}$) factor set can be done ignoring the existence of higher factors. We model the elementary divisor set accordingly.

- In case of $\phi(x)$ other than $(x+1)$, a cycle of length ($\leq 2^{\mathcal{M}} \cdot k$) arises from polynomial $\leq \phi(x)^{2^{\mathcal{M}}}$. Therefore, the underlying elementary divisor set $f(x)$ which generates the states $S_{2^{\mathcal{M}}.k}$ is given by the relation

$$f(x) = \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{L} \quad L \leq 2^{\mathcal{M}} \tag{4.21}$$

- In case of $(x+1)$, the $S_{2^{\mathcal{M}}.k}$ has arisen from the characteristic polynomial $f(x)$, where

$$f(x) = \phi(x)^{2^{\mathcal{M}-1}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{L} \tag{4.22}$$

  where the total states covered by $f(x)$ is $S_{2^{\mathcal{M}}.k}$; This is because one $(x+1)$-elementary divisor must have power between $2^{\mathcal{M}-1}$ and $2^{\mathcal{M}} - 1$. We are considering the least value.

Based upon the above knowledge we develop the theorem for enumeration of j^{th} factor set - $\tilde{N}(F_j)$ reported next.

Lemma 4.11 *Given $S_{2^{\mathcal{M}}.k}$, the number of j^{th} factor set $\tilde{N}(F_j)$ $(j \leq \mathcal{M})$ is given by the equation*

$$N(F_j) = \begin{cases} \tilde{N} + B & j = 0 \\ (\tilde{N} + B)2^{j-1} & 0 < j < \lceil log_2 L \rceil \\ (\tilde{N})2^{j-1} + (L - 2^{j-1}) & j = \lceil log_2 L \rceil \\ (\tilde{N})2^{j-1} & \lceil log_2 L \rceil < j < \mathcal{M} \\ (\tilde{N} - 1)2^{j-1} & j = \mathcal{M} \quad k = 1 \\ (\tilde{N})2^{j-1} & j = \mathcal{M} \quad k > 1 \end{cases} \tag{4.23}$$

where

$$\tilde{N} = \begin{cases} \lfloor \frac{log_{2^r} S + 2^{\mathcal{M}-1}}{2^{\mathcal{M}}} \rfloor & k = 1 \\ \lfloor \frac{log_{2^r} (S+1)}{2^{\mathcal{M}}} \rfloor & k > 1 \end{cases} \tag{4.24}$$

and

$$L = \begin{cases} (log_{2^r}(S+1) - 2^{\mathcal{M}-1}) mod\ (2^{\mathcal{M}}) & k = 1 \\ log_{2^r} S mod\ 2^{\mathcal{M}} & k > 1 \end{cases} \tag{4.25}$$

and

$$B = \begin{cases} 0 & L = 0 \\ 1 & L > 0 \end{cases} \tag{4.26}$$

Proof : The proof is only shown for $\phi(x) \neq (x+1)$. The proof for $(x+1)$ accordingly follows. A cycle of length $2^{\mathcal{M}} \cdot k$ arises from polynomial $\phi(x)^{2^{\mathcal{M}}}$. Therefore, the underlying elementary divisor set $f(x)$ is given by the relation

$$f(x) = \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{2^{\mathcal{M}}} \cdot \phi(x)^{L} \tag{4.27}$$

where the total states covered by $f(x)$ is $S_{2^{\mathcal{M}}.k}$; The number of elementary divisors having the power $2^{\mathcal{M}}$ is $\tilde{N}$ and the residual states which cannot be covered by $\phi(x)^{2^{\mathcal{M}}}$ is covered by $\phi(x)^{L}$.

The number of elementary divisor is

$$\mathcal{N} = \begin{cases} \tilde{N} & L = 0 \\ \tilde{N}+1 & L > 0 \end{cases} \tag{4.28}$$

Generalizing
A. $\mathcal{N} = \tilde{N} + B$, where B follows *relation 4.26.*

The highest factor set which L covers is given by $\tilde{L} = \lceil log_2 L \rceil$. Hence each of the $\tilde{N} + B$ divisor have the full factor set 0 to $\tilde{L} - 1$. Therefore,
B. $N(F_j) = (\tilde{N} + B) \cdot 2^{j-1}; 0 < j < \tilde{L}$

For j $= \tilde{L}$, the $\tilde{N}$ elementary divisor has full $\tilde{L}^{th}$ factor set, while the $(\tilde{N}+1)^{th}$ elementary divisor has $L - 2^{j-1}$ factors of $\tilde{L}^{th}$ factor set. Therefore,
C. $N(F_j) = (\tilde{N}) \cdot 2^{j-1} + (L - 2^{j-1}); j = \tilde{L}$

For the j^{th} factor set, where $\tilde{L} < j < \mathcal{M}$ present in the first $\tilde{N}$ elementary divisor. Therefore,
D. $N(F_j) = (\tilde{N}) \cdot 2^{j-1}; \tilde{L} < j \leq \mathcal{M}$.

The value of $\tilde{N}$ & L is obtained from equating the number of states covered by $f(x)$ with $S_{2^{\mathcal{M}}.k}$.

$$S_{2^{\mathcal{M}}.k} = 2^{r \cdot 2^{\mathcal{M}} \cdot \tilde{N}} \times 2^{\tilde{L}} \tag{4.29}$$

□

An example is given to explain the method.

Example 4.15 *Given* $CS' = [1(1), 4095[12], 2560(24)]$.
Then $CI = [4095[12]]$, $\mathcal{M} = 2$ $r = 2$, $S = 4095$, $k = 3$;
$\tilde{N} = \lfloor \frac{log_{2^2}(4095)}{2^2} \rfloor = 1$
$L = log_{2^2}(4095)\ mod\ 2^2 = 2$;
$\tilde{L} = log_2(2) = 1$;
Therefore, B =1. Calculating each of the factor set.
$N(F_0) = 1 + 1 = 2$;
$N(F_1) = 1 \times 2^{1-1} + (2 - 2^{1-1}) = 2$;
$N(F_2) = 1 \times 2^{2-1} = 2$.
The total number of factors covered by $\tilde{N}(F_0) + \tilde{N}(F_2) + \tilde{N}(F_2) = 6$.

Next we find $\tilde{N}(F_3)$ *which has to be found by Relation 4.6* $\tilde{N}(F_3) = \frac{1}{2}log_2(2560 \times 3 \times 2^{3-1\times2\times6} + 1) = log_2(16)/2 = 2$

Finding out the PCS is now straightforward. The table shows step by step calculation of each of the parameters to arrive at the final result.

Table 4.2: Calculation of Primary Cycle Structure

j	$N(F_j)$	$N(PCS(j))$	$\mathcal{N}_j$	P_j
0	2	2	0	0
1	2	2	1	2
2	2	1	0	0
3	1	1	1	6

Therefore, the final output of Primary Cycle Structure in Triplet Form is
$CS = (1,3)^2 \times (1,3)^6$

For case where $k = 1$ that is the underlying irreducible polynomial is (x+1), an example is given to explain the method.

Example 4.16 *Given the cycle structure* CS = [16(4), 24(8)]
$\mathcal{M}$ *= 2, r=1;* CI = [64[4]]
The value of $\tilde{N} = \lceil \frac{log_{2^1}(16)+2^1}{2^2} \rceil$ *= 2.*
Similarly, the value of $L = (log_{2^1}(16) - 2^1) mod\ 2^2$ *=0.*
Hence, B = 0.
$N(F_0)$ *= 1 + 1 = 2.*
$N(F_1)$ *= (1 + 1)*2^{1-1} *= 2.*
$N(F_2)$ *= 1 ×* 2^{2-1} *= 2.*
Total number of factors covered is $N(F_0) + N(F_1) + N(F_2)$ *=6.*
$N(F_3)$ *= 1; calculated using* relation 4.6
Therefore, $CS = (1,1)^2 \times (1,1)^5$

Complexity Analysis of Algorithm 4.11 The complexity of the algorithm is same as that of *Algorithm 4.4*. This is because the complexity of enumerating $\tilde{N}(F_j)$ from CI through *relation 4.23* is same as the complexity of $\tilde{N}(F_j)$ from $\mu_{2^i.k}$ through *relation 4.6*.

We now present the method of synthesizing the inversion vector F

4.2.3 Synthesis of Inversion Vector - F

The synthesis of F vector is guided by the principle laid down by the results of the *Theorem 3.13 & 3.12* which states that the inversion vector F should lie . only in the null space of $(x+1)^{n_i}$ - where $\lfloor log_2(n_i) \rfloor + 1 = \mathcal{M}$ - the factor of the characteristic polynomial $f(x)$ responsible for imparting the additive structure. To synthesize F holding such property, we first synthesize F for the characteristic polynomial $f(x)$ having $(x+1)^{n_i}$ as the single Elementary Divisor, that is $f(x) = (x+1)^{n_i}$.

The T matrix representing $(x+1)^{n_i}$ is given by

$$T(x+1)^{n_i} = \begin{bmatrix} 1 & 1 & 0 & \cdots \\ 0 & 1 & 1 & 0 \cdots \\ \cdots & & & \\ \cdots & & & \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{n_i \times n_i}$$

Given the above T matrix representing $(x+1)^{n_i}$, the following lemma characterizes the desired F vector.

Lemma 4.12 *The inversion vector F with all $1's$ lies only in the null space of $(x+1)^{n_i}$.*

Proof: None of the equations $(T+I)^j \cdot F = 0$ unless $j = n_i$, that is, $F = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \end{bmatrix}^t_{n_i \times 1}$ lies only in the null space of $(x+1)^{n_i}$. □

Using this result, we generalize the synthesis of F vector for a characteristic polynomial $f(x) = \phi(x)^{n_1} \cdots (x+1)^{n_i} \cdots \phi(x)^{n_\mathcal{N}}$.

The characteristic polynomial is the direct sum of individual elementary divisors and consequently, the characteristic matrix corresponds to each elementary divisor are arranged in block diagonal form to obtain the final matrix T (*Theorem 4.9*). Hence each elementary divisor (ED) occupies a *Block* in the total matrix T where *Block* is defined below.

Definition 4.3 *Block (n_1,r) : A Block(n_1, r) is a submatrix of dimension $r \times r$ formed from matrix T by selecting elements from r columns and r rows of T staring from the (n_1,n_1) position.*

Example 4.17 *Given the characteristic polynomial $f(x)$ in elementary divisor form as $f(x) = x^2(x+1)(x+1)^2(x^2+x+2)$.*
The T matrix representing it is

$$T = \begin{pmatrix} \lceil 0 & 0 \rceil & 0 & 0 & 0 & 0 & 0 \\ \lfloor 1 & 0 \rfloor & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & [1] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lceil 1 & 1 \rceil & 0 & 0 \\ 0 & 0 & 0 & \lfloor 0 & 1 \rfloor & 0 & 0 \\ 0 & 0 & : 0 & 0 & 0 & \lceil 0 & 1 \rceil \\ 0 & 0 & 0 & 0 & 0 & \lfloor 1 & 1 \rfloor \end{pmatrix}$$

The $T(x+1)^2$ occupies Block(4,2).

Let $T(x+1)^{n_i}$ occupy the Block(L,n_i) Then the following lemma defines the method for synthesizing the desired F vector.

Lemma 4.13 *The inversion vector F has all $1's$ in the (L) to (L+n_i -1)th row belongs to the null space of $(T+I)^{n_i}$.*

Proof: Since the matrices are arranged in block diagonal form, the value of the equation $(T+1)^j \cdot F$ only depends upon the multiplication of the Block(L,n_i) with the (L)-(L+n_i - 1)th

rows of the F-vector. Therefore, whatever be the values in the other rows $(T+1)^j \cdot F = 0$ only when $n_i = j$. Hence the proof. □

Example 4.18 *The F vector which lies only in the null space of* $Tx^2(x+1)(x+1)^2(x^2+x+1)$ *of the previous example is given by* $F = (\ 0/1 \quad 0/1 \quad 0/1 \quad [1 \quad 1] \quad 0/1 \quad 0/1\ )^T$

The complete algorithm is presented

Algorithm 4.12 *Enum_F_from_T(T,* $\mathcal{M}$*, F)*
Input : T Matrix, minimum additive factor : $\mathcal{M}$
Output : Inversion Vector F, Cycle Length $\mathcal{M}$
Step 1 : Identify the Block in T matrix representing $(x+1)^{n_i}$ *where* $\mathcal{M} = \lceil log_2(n_i) \rceil + 1$
Step 2 : Put 1 in the corresponding row of F Vector
Step 3 : Put either 1 or 0 in the other rows of F Vector.

Complexity Analysis : Identification of Block takes O(n) time and the formation of F vector also takes $0(n)$ time. Hence the complexity of the algorithm is O(n).

Overall Complexity of the Algorithm - *Enum_ACA_from_CS'_depth*

The complexity of each of the steps A - D is same as the corresponding steps A - D in *Enum_LCA_from_CS_depth*. The additional work of synthesizing F Vector takes O(n) time which is much less than the overall complexity of *Enum_LCA_from_CS_depth*. Hence, complexity of *Enum_ACA_from_CS'_depth* is same as *Enum_LCA_from_CS_depth*.

4.2.4 Conclusion

The two chapters provide the detailed synthesis scheme for the Linear and Additive CA. The synthesis scheme can synthesize any legal linear or additive cycle structure(CS) to the corresponding CA. However, there may be cycle structure which may be required for different different application and may not be legal. So an interesting variation of the problem can be to synthesize a legal cycle structure 'close' to an illegal cycle structure We present the overview of the methodology of synthesizing an LCA from an illegal cycle structure. The details are however, beyond the scope of the thesis. Hence not discussed here.

Synthesis of LCA Close to the Specified Structure

In order to synthesize cycle structure CS_{il} that is not legal, we first convert the illegal cycle structure CS_{il} to CS_{legal} which 'close' to CS_{il} and then apply the synthesis algorithm. An example to illustrate the case.

Example 4.19 *Let* CS_{il} = [1(1), 1(3), 3(4)] *in GF(2). We convert* CS_{il} *to a legal cycle structure* CS_{legal} *which is 'close' to the illegal cycle structure (CS). A 'close' legal cycle structure is* CS_{legal} *= [1(1), 1(3), 2(6)].*
After this conversion, we apply the algorithm Syn_LCA *and obtain the LCA whose T matrix is*

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

The points which are to be involved in this conversion are

- The time complexity to convert an illegal cycle structure into legal cycle structure must be polynomial.
- The definition of 'close' will vary from application to application, the algorithm should be capable enough to handle each of the situation.
- Moreover, the direction of the algorithm should be based upon a quantitative assessment of 'close' rather than any qualitative, perceptual observation.

The in-depth discussions of the above issues require detailed analysis which is beyond the scope of this thesis and hence omitted.

The analysis and synthesis schemes provide a solid foundation based upon which we explore a special class of linear CA - termed as $MACA$ - which is used to develop the applications of pattern recognition and pattern classification.

Terminology

weight(w)	Number of ones in a pattern
$MACA(n, m)$	n-bit $MACA$ with 2^m attractors
$P_x(w, n, m, y)$	Set of patterns in 0-basin of $MACA$ w signifies weight, n signifies size of the pattern, 2^m signifies the number of attractors y signifies an optional parameter characterizing the vector subspace, (y is absent if the parameter we are looking for refers to the average value of a number of subspaces) x = Number of $MACA$ involved (x is ignored if $x = 1$)
$\|P\|$	Number of patterns in the set P
$\|\hat{P}(w, n, m)\|$	Number of n-bit patterns with weight w chosen unbiasedly forming a pattern pool of 2^{n-m}
$N(n, m)$	All possible n-bit $MACA$ with 2^m attractors

Chapter 5

Multiple Attractor Cellular Automata ($MACA$)

The last few decades of the twentieth century have encountered massive advancement in computing technology. The computing speed has leaped from kilo to mega to giga and is now reaching the unthinkable terra flops. However, in spite of such leaps, machines have failed to match the ease with which human brains recognize/classify patterns. The conventional machine in spite of gaining immense power leads to match the human brain due to its inherent weak memory organization concept.

The entire human brains is divided into zones each zone either associating or classifying a class of element. *Fig.* shows the broad methodology followed by human brain. In *Fig.* , it is seen that the brain associates all the different variations of same letter around a pivot point. While in *FIg.*, the brain classifies similar object in one zone. To match the dexterity of human brain, scientists, researchers, practitioners are trying to build machine which can simulate the associativity and classification capacity of human brain.

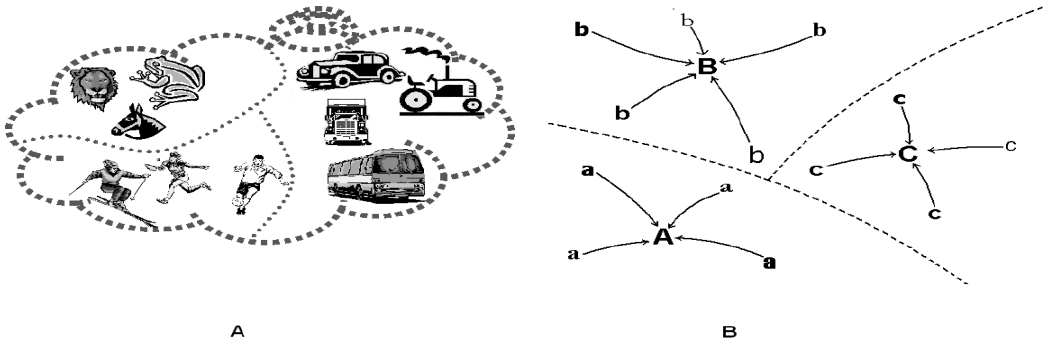


Figure 5.1: Association and Classification

This chapter introduces a special class of Cellular Automata termed as Multiple Attractor Cellular Automata. This class of cellular automata is capable of emulating the pattern

classification and association model of the human brain. The unique state transition behavior of the $MACA$ imparts it the power. The state transition graph of the $MACA$(*Fig.* divides the entire vector space into discreet zones, each zone converging towards a sink state. This nature of state transition diagram exactly maps with a classifier/associative memory model. In this chapter, we analyse the property of the state transition behavior and show that the division of state space into discrete zone is based on distance metric. We also describe the methodology of synthesizing a particular $MACA$ through evolutionary algorithm. The synthesized $MACAs$ state transition behavior is oriented according to a given problem. Based upon the analysis and synthesis technique we describe the $MACA$ Based Associative Memory and $MACA$ Based Pattern Classifier in the next chapter.

A brief preliminary overview of $MACA$ follows.

5.1 Multiple Attractor Cellular Automata

The set of non-cyclic states of an $MACA$ forms inverted trees rooted at the cyclic states. The *cycles* are referred to as *attractors*. *Fig.5.2* depicts the state transition diagram of a 5-cell $MACA$ with four attractors {00000(0),00011(3),00101(5),00110(6)} having self loop. In the following two chapters, by an attractor we refer to a cycle of length 1. The states of a tree rooted at the cyclic state α form the α-*basin*.

Definition 5.1 *The* depth d *of an $MACA$ is the number of edges between a non-reachable state and its attractor. For the 5-cell $MACA$ of* Fig.5.2 $d = 3$.

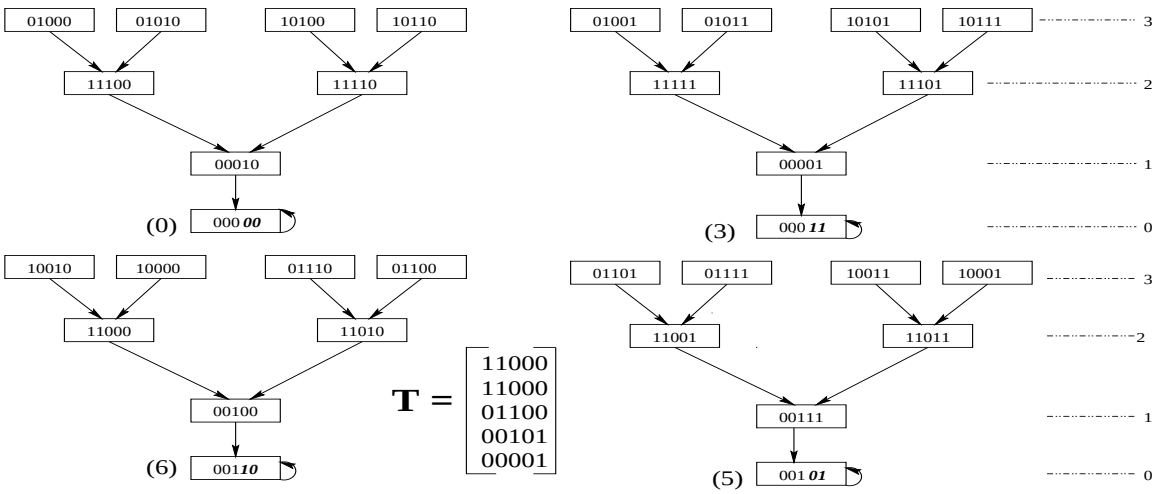


Figure 5.2: State transition diagram of a 5-cell $MACA$ with Characteristic Matrix T and Rule Vector $< 102, 60, 60, 90, 204 >$
Note : (i) The i^{th} cell (i = 1 to 5) employs the rule specified by the i^{th} element of the rule vector; the corresponding dependency, as specified in *Table I*, gets reflected by the i^{th} row of the T matrix.
(ii) Four cyclic states 0,3,5,6 are referred to as attractors and the corresponding inverted tree rooted on α (α = 0, 3, 5, 6) as α-basin.

The details on the characterization of $MACA$ is available in [9]. A few fundamental results for an n-cell $MACA$ having k attractor basins is next outlined.

- *Result I:* The characteristic polynomial of the $MACA$ is $x^{n-m}(1+x)^m$, where $m = log_2(k)$.
- *Result II:* The characteristic polynomial of an $MACA$ can be also written in elementary divisor form as $x^{d_1} \cdot x^{d_2} \cdot \cdot x^{d_p} \cdot (1+x) \cdot (1+x) \cdot \cdot m \ times$ where $d_1 \geq d_2 \cdot \cdot \geq d_p$ and $d_1 + d_2 \cdot \cdot d_p = n - m$.
- *Result III:* The minimal polynomial of an $MACA$ is $x^{d_1} \cdot (1+x)$, where d_1 is the depth.

Definition 5.2 *An m-bit field of an n-bit pattern set is said to be pseudo-exhaustive (PEF) if all possible* 2^m *patterns appear in the set.*

Theorem 5.1 *In an n cell MACA with* $k = 2^m$ *attractors, there exists* m-bit *positions at which the attractors generate pseudo-exhaustive* 2^m *patterns [9].*

Theorem 5.2 *The* modulo-2 *sum of two states is a predecessor of* 0-state *(pattern with all 0's) if and only if the two states lie in the same attractor basin [9].*

Example 5.1 *The example MACA of* Fig.5.2 *illustrates the above results.*

- *It is a* 5*-cell MACA having* 4 *attractors and the depth of the MACA is 3.*
- *The characteristic polynomial of the MACA is* $x^3 \cdot (1+x)^2$. *Therefore,* $m = 2$*; the number of attractors* $k = 2^m = 4$*; depth* $d_1 = 3$. *Its elementary divisor form is* $x^3 \cdot (1+x) \cdot (1+x)$ *and the minimal polynomial* $= x^3 \cdot (1+x)$.
- *In the MACA of* Fig.5.2, *two least significant bit positions constitute the pseudo-exhaustive field (PEF). Let take an attractor* 00011 *and any two states* 11111, 11101 *of* 00011*-basin. The* modulo-2 *sum of these two is* 00010 *which is a state in* $0-basin$. *By contrast, for two states* 00001 *and* 11000 *belonging to two different attractor basins* 00001 *and* 11000 *respectively, the* modulo-2 *sum is* 11011 *which is a state of the non-zero* 00101 *basin.*

Based upon the above properties of $MACA$, we design a special technique of synthesizing an n-cell, 2^m attractor $MACA$ termed as $MACA(n, m)$

Construction of MACA(n, m) : The basic guideline of the technique is dictated through the following theorem and its proof.

Theorem 5.3 *An* $MACA(n, m)$ *is conceived as a combination of m number of* n_i*-bit (*$i = 1, 2, \cdots m$*) 2-attractor MACA such that* $n_1 + n_2 + \cdots + n_m = n$

Proof : The characteristic polynomial of an $MACA$ (noted under *Result II* of *Section II-B*) can be written in elementary divisor form as $< x^{d_1} \cdot x^{d_2} \cdots x^{d_p} \cdot (1+x) \cdot (1+x) \cdots m \ times \ >$ The entire vector space produced by $MACA$ is the direct sum of individual vector spaces produced by each elementary divisor [27].

The elementary divisors can be clubbed into m groups each containing one $(1+x)$. Each cluster has the characteristic and minimal polynomial as $x^d(1+x)$, and therefore constructs an $MACA$ with 2-attractor basins. □

The construction of T matrix representing $MACA(n, m)$ following the above theorem is as follows. The T matrix is formed from block diagonal arrangement of $T_i s$ where each T_i represents an $MACA(n_i, 1)$ in *Figure 5.3.*

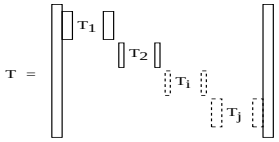


Figure 5.3: T_1, T_2, $\cdots$, etc. in Block Diagonal Form.

Example 5.2 The characteristic polynomial of the $MACA$ in *Fig. 5.2* can be represented as two clusters - $[(1+x) \cdot x^3] \cdot [(1+x)]$. The T matrix representing the $MACA$ can be conceived as block diagonal arrangement of T_1 & T_2 where

$$T = \begin{bmatrix} \lceil 1 & \overline{0} & \overline{0} & 0 \rceil & 0 \\ \lfloor 0 & 0 & 0 & 0 \rfloor & 0 \\ \lfloor 0 & 1 & 0 & 0 \rfloor & 0 \\ \lfloor 0 & \underline{0} & \underline{1} & 0 \rfloor & 0 \\ 0 & 0 & 0 & 0 & [1] \end{bmatrix} \text{ where } T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } T_2 = \begin{bmatrix} 1 \end{bmatrix}$$

Upon studying the pattern distribution of each cluster, the pattern distribution of zero basin of the corresponding $MACA$ can be easily accomplished through Cartesian product of each cluster. □

We term all the $MACA$ formed through this process as members of $MACA$ family and show that the family has a special property. The property of PEF allows $MACA$ to behave as a hash function. **MACA as a Hash Function:** For the purpose of hashing, a node in an $MACA$ tree (*Fig.5.2*) is viewed as the key to be hashed. The address for a key is given by the PEF of the attractor containing the key. That is,

$$H_{MACA}(key) = PEF(attractor) \tag{5.1}$$

where $H_{MACA}(key)$ is the hashed value of given key. The hash table size in this scheme is k, the number of attractors of the $MACA$.

Example 5.3 *The $MACA$ in* Fig. 5.2 *provides an Hash Table of size 4 with table entry (0 to 3), each attractor uniquely identifies its position in the table by the pseudo-exhaustive field(PEF). And for example all patterns in the basin* 11111 *hash to table entry 3(11).*

The $MACA$ family has an unique property in terms of hashing. It forms an *Hamming Hash Family (HHF)* - a hash family where the probability of collision of two patterns varies inversely with their hamming distance. The analytical foundation of HHF is next discussed

5.2 Hamming Hash Family

The concept of HHF is established through a detailed study of the vector space generated by the 0-basin.

Theorem 5.4 *The pattern distribution of the* 0*-basin reflects the nature of collision with respect to Hamming Distance between two random patterns hashed with* $MACA$.

Proof : Let $\mathcal{P}_{incom}$ be a noisy pattern derived out of a pattern $\mathcal{P}_i$. In an $MACA$ let a pair of patterns ($\mathcal{P}_{incom}$,$\mathcal{P}_i$) fall in the same basin where, $\mathcal{P}_{xor} = \mathcal{P}_i \oplus \mathcal{P}_{incom}$. Let $w(\mathcal{P}_{xor})$ = weight of pattern $\mathcal{P}_{xor}$ = number of $1's$ in $\mathcal{P}_{xor}$. So, $HD(\mathcal{P}_i, \mathcal{P}_{incom}) = w(\mathcal{P}_{xor})$. As per *Theorem 5.2*, $\mathcal{P}_{xor}$ falls in the 0-basin. Therefore, studying the distribution of patterns in 0-basin in respect of their weight is equivalent to studying the nature of collision of $\mathcal{P}_{incom}$ and $\mathcal{P}_i$ in terms of $\mathcal{P}_{incom}$s distance from $\mathcal{P}_i$. □

The behavior of the patterns which collide that is, fall under the same attractor basin is explained in *Example 1.* The pattern representing *modulo-2* sum of two states in the same attractor basin of a $MACA$ lies in the 0-basin. For the example $MACA$ of *Fig.5.2*, the patterns s_i=11001 and s_j=11100 are the members of 31-basin. The pattern s_h= 00101(11001 $\oplus$ 11100) is a state of the 0-basin and it corresponds to the *hamming distance* between 11001 & 11100.

Therefore, to establish the concept of HHF in $MACA$, we have to show that the 0-basin has a definite bias for lower weight patterns. The bias gets reflected in the ratio of (i) the probability of a pattern of particular weight(w) to fall in the 0-basin and (ii) the probability that a pattern when chosen from an unbiased pattern pool of cardinality 2^{n-m} (the cardinality of the 0-basin) would be of that particular weight(w). In other words, the bias can also be expressed in terms of ratio of the average number of patterns of weight w falling in the 0-basin of an $MACA(n, m)$ referred to as $|P(w, n, m)|$ and $|\hat{P}(w, n, m)|$ - it is defined as the number of patterns of weight w in pattern pool of cardinality 2^{n-m} provided each pattern in the pool is chosen unbiasedly from the entire n-bit pattern pool of 2^n.

Expected Occurrence: The Expected Occurrence (EO) of patterns $P(w, n, m)$ with particular weight (w) in the 0-basin of an $MACA(n, m)$ can be denoted as

$$EO(P(w, n, m)) = \frac{|P(w, n, m)|}{|\hat{P}(w, n, m)|} \tag{5.2}$$

Computation of $|\hat{P}(w, n, m)|$

The total number of w-weighted n-bit pattern = ${}^nC_w$. If we want to chose a pattern pool of cardinality 2^{n-m}, so that there is no bias towards any particular w, then the number of

w-weighted patterns $|\hat{P}(w,n,m)|$ in that pool can be derived by simple unitary method

$$|\hat{P}(w,n,m)| = \frac{^nC_w}{2^n} \times 2^{n-m} = \frac{^nC_w}{2^m} \tag{5.3}$$

The computation of $|P(w,n,m)|$ demands a detailed analysis of 0-basin described in the following subsection.

Computation of $|P(w,n,m)|$

In a vector space of dimension n, the total number of patterns $= 2^n$ and $|P(w,n)| = {}^nC_w$; that is, out of n bits w number of bits are 1's in such patterns. In a subspace of dimension $(n-m)$, the total number of patterns $= 2^{n-m}$. If there is no bias towards any particular w, then

$$|P(w,n,m)| = \frac{^nC_w}{2^m} \tag{5.4}$$

The computation of $|P(w,n,m)|$ requires a detailed analysis of 0-basin described in the following subsection.

Computation of $|P(w,n,m)|$

An $MACA(n,m)$, as already mentioned, is so constructed so that it can be conceived as concatenation of m - $MACA(n_i,1)$. Therefore, the computation of $P(w,n,m)$ is based upon the computation of $|P(w,n_i,1)|$. The details of $|P(w,n,m)|$ and $|P(w,n_i,1)|$ are explained in next section of the chapter. The following theorem theorizes the relation between $|P(w,n,m)|$ and $|P(w,n,1)|$.

Theorem 5.5 *Each w weighted pattern $P(w,n,m)$ in 2^m attractor $MACA$ is formed by selecting an unique combination of w_j bit pattern from each of the n_i bit 2-attractor $MACA$ and concatenating them. such that $w_1 + w_2 + \cdots + w_n = w$ and $n_1 + n_2 + \cdots n_m = n$.*

Proof : An n-bit pattern x in the 0-basin of the $MACA$ follows the relation $T \cdot x = 0$. Since the T matrix representing the $MACA(n,m)$ is formed from placing m $T_i s$ in block diagonal form, the relation can be represented as

$$T \times X = \begin{bmatrix} [T_1] & 0 & \cdots \\ 0 & [T_2] & 0 \cdots \\ . & & \\ . & & \\ . & & \\ 0 & \cdots & [T_m] \end{bmatrix} \times \begin{bmatrix} [X_1] \\ [X_2] \\ . \\ . \\ . \\ [X_m] \end{bmatrix} = 0$$

where X_i represents n_i-bits of X and occupying the same rows as T_i. Therefore, each $T_i \cdot X_i$ individually is zero. Hence each X is zero. Hence each X is formed from concatenation of unique combination of $X_i s$. □

As already mentioned, the details on computing $P(w,n,m)$ are noted in next section. We in this section show the nature of the graphs of $EO(P(w,n,m))$ obtained from enumerating *relation 5.2*. The graphs of *Fig.* **??** - *5.7* plots the Expected Occurrences $EO(P(w,n,m))$ -

denoted by *relation 5.2* in the y-axis, while the weight of patterns is plotted on x-axis. In *Fig.* **??**, *5.5* & *5.7*, the weights are represented as a fraction of n (the number of bits in a pattern) while in *Fig. 5.6* the x- axis plots in terms of the absolute value of w.

Fig. **??** depicts the expected occurrence $EO(P(w, n, 1))$ for different weights of patterns. From the graph, it can be observed that the bias for low weight patterns in 0 basin gets reflected.

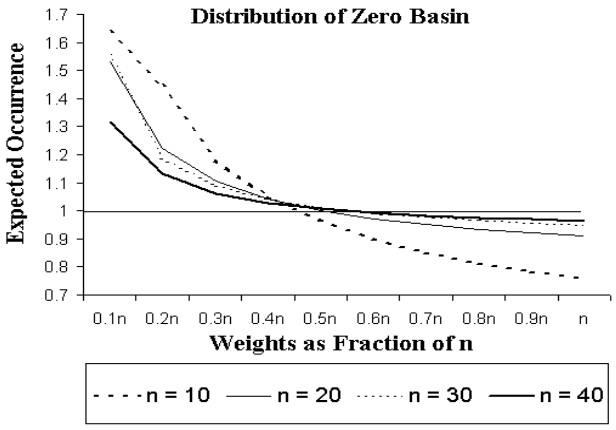
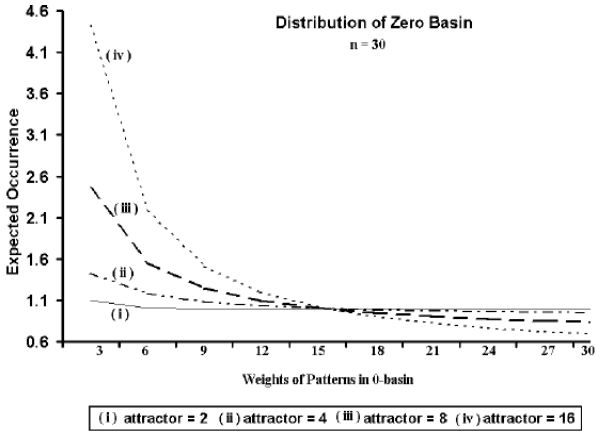


Figure 5.5: Expected Distribution of MACA with 4 attractors ($m = 2$)

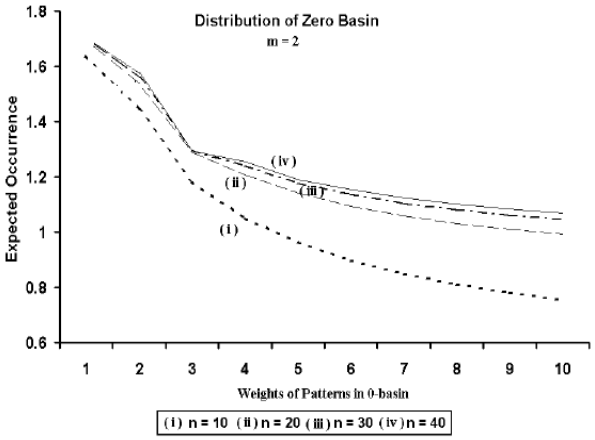
The biasness becomes more prominent as we increase the value of m. The graph in *Fig.5.5* plots the expected occurrence EO for different values of n for $m = 2$. Unlike for $m = 1$, the function becomes monotonically decreasing. That is, the probability of appearance of lower weight patterns in the 0 basin is significantly higher than its higher weight counterparts. The same graph is plotted in *Fig 5.6* with the x-axis representing the first 10 weights in absolute terms rather than as fraction of n. It is seen that with the increase of n the value of Expected Occurrence of lower values of w rises significantly.

Fig.5.7 shows that the expectation of lower weight patterns in 0 basin increases significantly as the value of m is further increased.

The formal proof of the bias of lower weight patterns in the 0 basin - the fundamental foundation stone of Hamming Hash Family is presented next. As already mentioned in *Theorem 5.3* & *Theorem 5.5*, the $MACA(n, m)$ and its patterns in 0-basin are formed from concatenating m- $MACA(n_i, 1)$ and its corresponding patterns. Due to this concatenation, the slight bias of lower weight patterns in two attractor $MACA$ (*Fig.* **??**) gets magnified as we increase the number of attractors(*Fig. 5.5*) The formal proof of the bias of lower weight patterns in the 0-basin - the fundamental foundation stone of Hamming Hash Family is presented next. To prove it for $m > 1$, we make an important approximation, that the bias of smaller weights is not present in 2-attractor $MACA$. We show that even then the effect of concatenation brings in biasness for lower weight patterns in $MACA(n, m)$ for $m > 1$.



A. For multiple attractors (m = 1, 2, 3, 4), n = 30



B. For attractor (m = 2), n = 10, 20, 30, 40

Figure 5.6: Expected Occurrence(EO) of patterns with weight w in 0 basin (as per *relation 5.2*)

A bias-less 0-basin of a two attractor basin $MACA$ will always have the all zero-vector as its attractor state. Hence,

$$|P(0, n, 1)| = Number\ of\ patterns\ with\ 0\ weight = 1 \tag{5.5}$$

while the number of patterns with weight w in the 0-basin (*equation 5.4*)

$$|P(w, n, 1)| = \frac{2^{n-1} - 1}{2^n - 1} \cdot {}^nC_w \tag{5.6}$$

This is obtained by excluding zero vector from both the vector space which has a cardinality of 2^n and the 0-basin which has a cardinality of 2^{n-1}.

For subsequent discussions, we shall use the symbol κ where

$$\kappa = \frac{2^{n-1} - 1}{2^n - 1}$$

The following theorem formally characterizes the occurrence of patterns with weight w - that is, the patterns using w number of $1's$.

Theorem 5.6 : *The probability of occurrence of a pattern with weight w in the 0-basin of an $MACA(n, m)$ varies inversely with w.*

Proof : We present the proof for $m = 2$ and show that it can be easily generalized for any value of m. Let an $MACA(n, 2)$ be formed from combination of $MACA(n_i, 1)$ & $MACA(\overline{n}_i, 1)$ where $n_i + \overline{n}_i = n$. We show that the pattern distribution of this $MACA(n, 2)$ varies inversely with w.

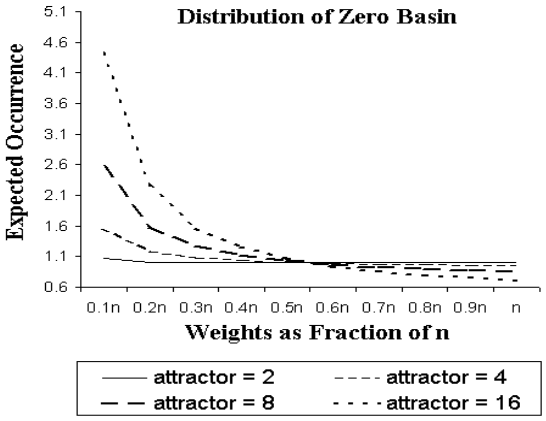


Figure 5.7: Expected Distribution of MACA (n =30) with multiple attractors (m = 1,2,3,4)

The number of w-weighted pattern $P(w,n,2)$ for a particular n_i will be formed by picking w_j weighted patterns from n_i and $\overline{w}_j$ weighted patterns from $\overline{n}_i$ (such that $w_j + \overline{w}_j = w$) and concatenating them. The number of w_j weighted patterns in $MACA(n_i, 1)$ as well as the number of $\overline{w}_j$ weighted patterns present in $MACA(\overline{n}_i, 1)$ is defined by *equations 5.5 & 5.6*. Therefore, $|P(w,n,2)|$ is given by

$$|P(w,n,2)| = \quad (\kappa \cdot^{n_i} C_w \cdot^{\overline{n}_i} C_0 + \cdots \kappa \cdot^{n_i} C_{w_j} \cdot \kappa \cdot^{\overline{n}_i} C_{\overline{w}_j} \cdots\cdots\cdots + ^{n_i} C_0 \cdot \kappa \cdot^{\overline{n}_i} C_w)$$

$n_i + \overline{n}_i = n$ and $w_j + \overline{w_j} = w$.
The equation can be rewritten as

$$|P(w,n,2)| = \kappa \cdot (1-\kappa) \cdot (^{n_i}C_w + ^{\overline{n}_i} C_w) + \kappa^2 \cdot (\sum_j {}^{n_i}C_{w_j} \cdot^{\overline{n}_i} C_{\overline{w}_j}) \tag{5.7}$$

Hence, the Expected Occurrence $EO(P(w,n,2))$ of a pattern of weight w from *equation 5.2*

$$EO(P(w,n,2)) = \frac{|P(w,n,2)|}{|\hat{P}(w,n,2)|} \tag{5.8}$$

From *equation 5.4*

$$|\hat{P}(w,n,2)| = \frac{^nC_w}{2^2} \tag{5.9}$$

Therefore,

$$EO(P(w,n,2)) = \quad \frac{\kappa \cdot (1-\kappa) \cdot (^{n_i}C_w + ^{\overline{n}_i} C_w)\kappa^2 \cdot (\sum_j {}^{n_i}C_{w_j} \cdot^{\overline{n}_i} C_{\overline{w}_j})}{\frac{^nC_w}{4}}$$

The value of

$$\frac{\sum_j {}^{n_i}C_{w_j} \cdot^{\overline{n}_i} C_{\overline{w}_j}}{{}^{n}C_w} = 1$$

as it is a hyper-geometric function[49]. Therefore,

$$EO(P(w,n,2)) = \frac{({}^{n_i}C_w +^{\overline{n}_i} C_w)}{{}^{n}C_w} \cdot 4 \cdot \kappa \cdot (1-\kappa) + 4 \cdot \kappa^2 \tag{5.10}$$

Both the functions

$$f_1(w) = \frac{{}^{n_i}C_w}{{}^{n}C_w}; \qquad f_2(w) = \frac{{}^{\overline{n}_i}C_w}{{}^{n}C_w} \tag{5.11}$$

are monotonically decreasing functions with respect to w for any particular value of n_i. Hence $EO(P(w,n,2))$ is a monotonically decreasing function. Hence the proof.

For any value of $m > 2$, the generalization can be done by assuming that the $MACA(n,m)$ is formed from concatenation of a 2^{m-1} attractor $MACA$ with a 2-attractor $MACA$. In line with the proof for $m = 2$, in this case also we assume that there is no bias in patterns either in $MACA(n_i, m-1)$ or in $MACA(\overline{n}_i, 1)$. □

The above discussions establish the fact that the 0-basin has strong bias for patterns with low weight value. Therefore, the patterns in a non 0-basin are close to each other in terms of Hamming Distance (HD), since HD of a pair of patterns in a non 0-basin reflects the weight of a pattern in 0-basin.

5.3 Computation of $|P(w,n,m)|$

This section elaborates the method of calculating $|P(w,n,m)|$ - the expected number of patterns with weight w in the 0 basin of an $MACA$ with 2^m attractors. The computation is carried in two steps. In *Step 1*, we compute $|P(w,n,1)|$ and subsequently calculate $|P(w,n,m)|$

To derive an analytical expression for computing $|P(w,n,m)|$, we introduce the concept of *Dependency Vector* and *Dependency String*. The *Dependency Vector* is used to explain $|P(w,n,1)|$ while *Dependency String* is used to explain $|P(w,n,m)|$ for $m > 1$.

Dependency Vector and Dependency String: In a subspace of dimension $(n-m)$, if the set of n-bit vectors of the subspace is conceived as a system of n variable equations, then there will be m dependent variables. The following example illustrates the point.

Example 5.4 *Take the vector subspace* V = {00000, 01001, 10001,11000, 00110, 01111, 10111,11110}. *All the vectors* $v_i \in V$ *are 5 (= n) bits long. The dimension of the vector subspace is* 3. *Therefore* m = 5 - 3 = 2. *Let the vector set (V) be a system of linear equations with five variables* (a,b,c,d,e). *Then the elements of the vector sub-space can be rewritten in the form noted in* Table 5.1.
In the set of equations of Table 5.1, *a and c are two dependent variables as defined below:*

(1) a is dependent on b and e, that is, $a \oplus b \oplus e$=0 in all the vectors $v_i \in V$; and
(2) c is dependent on d, that is, $c \oplus d = 0$ in all $v_i \in V$.

Table 5.1: System of Linear Equations representing a set of Vectors

Vector	Corresponding Linear Equation
00000	$0 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 0 \cdot e = 0$
01001	$0 \cdot a + 1 \cdot b + 0 \cdot c + 0 \cdot d + 1 \cdot e = 0$
10001	$1 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 1 \cdot e = 0$
11000	$1 \cdot a + 1 \cdot b + 0 \cdot c + 0 \cdot d + 0 \cdot e = 0$
00110	$0 \cdot a + 0 \cdot b + 1 \cdot c + 1 \cdot d + 0 \cdot e = 0$
01111	$0 \cdot a + 1 \cdot b + 1 \cdot c + 1 \cdot d + 1 \cdot e = 0$
10111	$1 \cdot a + 0 \cdot b + 1 \cdot c + 1 \cdot d + 1 \cdot e = 0$
11110	$1 \cdot a + 1 \cdot b + 1 \cdot c + 1 \cdot d + 0 \cdot e = 0$

In the context of the above illustrative example (*Table 5.1*) we next introduce the terms *Dependency Vector* and *Dependency String.*

Definition 5.3 Dependency Vector (DV) *represents each individual linear dependency relationship supported by all the elements in the vector sub-space (V). The two* dependency vectors *for the illustrative* Example 5.4 *are* $< 11001 >$ *&* $< 00110 >$. *Each of the n bits signify the variables in that order - that is, the bits in the DV represents the variable in the sequence* $< abcde >$. *The 1's in the DV specify the dependent variables; the summation (xor) of the corresponding variables in all $v_i \in V$ = 0, that is,*

$$DV \cdot v_i^t = 0 \qquad \forall v_i \in V \tag{5.12}$$

Definition 5.4 Dependency string (DS) *represents the multiple linear dependency, if it exists in the vector sub-space* (V). *The Dependency String in the* Example 5.4 *is* [11221] *where '1' indicate the relationship between a, b & e while '2' indicate the relationship between c and d. In essence, the two Dependency Vectors* (DV) *are merged together to form the string.*

A few basic concepts are noted for subsequent analysis.

- For an $(n-1)$ dimensional vector subspace, *Dependency Vector* and *Dependency String* are synonymous.
- *The* 0 *basin of an* $MACA(n, 1)$ *is a subspace of dimension* $(n-1)$. *So there is one dependent variable in the vector subspace and thus the characterization of the* 0 *basin can be solely done by a dependency vector.* Whereas the 0 basin of an $MACA(n, m)$ is a subspace of dimension $(n - m)$. Therefore, there are m dependent variables in the subspace. A *dependency string* representing multiple DV is necessary for characterization of $MACA(n, m)$.

- Let a dependency vectors involve $\tilde{w}$ number of variables, that is, the number of $1's$ (that is weight) of the vector is $\tilde{w}$. The number of w-weighted pattern in the vector subspace generated by the dependency vector is $|P(w, n, 1, \tilde{w})|$.
- The pattern distribution of an $(n-1)$ dimensional vector subspace (V) is determined by weight of the *dependency vector* (DV) representing the subspace. The relationship is expressed through the following lemma.

Lemma 5.7 The value of $|P(w, n, 1, \tilde{w})|$ is given by

$$|P(w, n, 1, \tilde{w})| = \sum_{k=0,2,\cdots}^{\tilde{w}} {}^{\tilde{w}}C_k . {}^{n-\tilde{w}}C_{w-k} \tag{5.13}$$

Proof : Let V be a vector subspace. In any vector $(v \in V)$, since $\tilde{w}$ bits are dependent, the summation of the corresponding $\tilde{w}$ bits in v is zero. Therefore, in those $\tilde{w}$ bits, the number of 1's is even. Hence the *relation 5.13* follows. □

5.3.1 Computation of $|P(w, n, 1)|$

The computation of $|P(w, n, 1)|$ is based upon the concept of legal dependency vector. A 3-neighborhood $MACA$ whose next state depends on itself, its left neighbor and right neighbor, cannot produce 0 basin supporting all the variations of *dependency vector*. In the present context, the *dependency vector* which can be generated by 0 basin of $MACA(n, m)$ is termed as *legal dependency vector* respectively. The following *Theorem* sets the guideline for determination of *legal dependency vector*.

Theorem 5.8 *The vector subspace of the* 0 *basin of a three neighborhood n cell MACA with two attractor basins cannot generate a Dependency Vector* $(\hat{DV})$ *of the form [1*$\cdots$ *1* $\cdots$ *(* k $0's$ *)* $\cdots$ *1* $\cdots$*1] with k number of 0's between a pair of 1's, where* $k \geq 2$.

Proof : The proof is presented for the case of k $(= 2)$ numbers of $0's$ between a pair of $1's$. It can be easily generalized for any value of k. The $\hat{DV}$ as per the theorem, is $\hat{DV} = [\ 1 \quad \cdots \quad 1_i \quad 0 \quad 0 \quad 1_{i+3} \quad \cdots \quad 1\ ]$, where the 1's at either end ensures its uniqueness for a particular value of n. (0 at any end would imply that the $\hat{DV}$ is supported by smaller values of n and hence the proof can proceed accordingly)

The characteristic and minimal polynomial of an n-cell $MACA$, as noted in [9], is $x^d(1+x)$(*Result III Sec IIA*), where d is the depth of its attractor basin, and $d = n$ - 1. Rank of the characteristic matrix T is given by

$$Rank(T) = n - 1 \tag{5.14}$$

From Cayley Hamilton Theorem [27], the following result follows :

$$T^d(I + T) = 0 \quad i.e. \quad T^d \cdot T = T^d \tag{5.15}$$

where T is the $n \times n$ $MACA$ matrix and

$$T^d \cdot v = 0, \forall v \in 0 - basin \tag{5.16}$$

The theorem characterizes a two attractor basin $MACA$ with its characteristic matrix T. So the number of elements in 0-basin is half the total number of elements in the vector space. Hence, there is only one independent row in T^d. and consequently, the rank of T^d is 1[9]. This row can be viewed as an n-dimensional dependency vector($\hat{DV}$) generating the vector sub-space of the $MACA$ 0-basin.

In the following proof, we show that the rank of the T matrix will not be equal to $(n-1)$ if the independent row of T^d is equal to the $\hat{DV}$ having k number of $0's$ in between a pair of $1's$, where $k = 2$. The proof is given by contradiction. We show that the $\hat{DV}$, as noted in the theorem, makes the rank of T matrix as $(n-2)$, that contradicts the *relation 5.14*. The proof is divided in four phases. The result of each phase is used to develop the next phase.

Phase 1 : Identifies the '0' elements of the tri-diagonal T matrix of the 3 neighborhood MACA

In a tri-diagonal matrix, all the elements other than main and two off diagonals are necessarily $0's$. So we concentrate on identifying the elements in this band.

The independent row of the T^d matrix can be conceived as $1 \times n$ dimensional vector where T^d is denoted as $\hat{DV} = [\ 1 \ \cdots \ 1 \ 0 \ 0 \ 1 \ \cdots \ 1\ ]$. Hence, to satisfy the *relation 5.15*, we concentrate on the independent row of $\hat{DV}$ and consequently we have

$$[\ 1 \ \cdots \ 1 \ 0 \ 0 \ 1 \ \cdots \ 1\ ] \times \begin{bmatrix} & & \\ \cdots & \mathbf{T} & \cdots \\ & & \end{bmatrix} = [\ 1 \ \cdots \ 1 \ 0 \ 0 \ 1 \ \cdots \ 1\ ] \tag{5.17}$$

where T is the 3-neighborhood band matrix representing the $MACA$ and having characteristic polynomial $x^d(1+x)$.

In order to satisfy the *relation 5.17*, the T matrix should be of the form

$$T = \begin{bmatrix} \cdot & i^{th}col & \cdot & \cdot & (i+3)^{th}col & \cdot \\ \cdot & \downarrow & \cdot & \cdot & \downarrow & \cdot \\ \cdot & * & [0] & 0 & 0 & \rightarrow i^{th}row\cdot \\ \cdot & * & * & * & 0 & \cdots \\ \cdot & 0 & * & * & * & \cdot \\ \cdot & 0 & 0 & [0] & * & \rightarrow i+3^{th}row\cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$ (* indicate either 0 or 1)

That is, in the $T_{i,i+1}th$ & $T_{i+3,i+2}th$ position (specified within [ ]), the T matrix can have only '0' element. Presence of these two $0's$ guide the remaining phases of the proof. Specifically, $T_{i,i+1} = 0$ guides the proof & conclusion of *Phase 2 & 3*, while the *Phase 4* summarizes the identical results for the case of $T_{i+3,i+1} = 0$.

Phase 2 : Identifies the nature of dependency of the i^{th} bit on the T matrix rows for evaluation of relation 5.16

Let any pattern $v \in 0{-}basin$ be represented by $v = [\ b_1 \quad \cdots \quad b_i \quad b_{i+1} \quad b_{i+2} \quad b_{i+3} \quad \cdots \quad b_n\ ]$.

Therefore, the *relation 5.16* - $T^d \cdot v = 0$, specifies that we obtain the all 0 vector by multiplying v by the T matrix successively d times.

Considering the bit b_i, the successive product of i^{th} row of T matrix produces $b_i|t_d = 0$, where (t_d) indicates the state of the bit after d time steps. The computation of b_i after d time steps depends on other rows of T in addition to its i^{th} row. We next probe this nature of dependency.

We first illustrate the nature of dependency for $t = 2$, then generalize the result for $t = d$.

In the i^{th} row, only the elements $T_{i,i-1}$, $T_{i,i}$ & $T_{i,i+1}$ can be non-zero, hence the following relations can be written.

After one time step,

$$b_i|t_1 = T_{i,i-1} \cdot b_{i-1} + T_{i,i} \cdot b_i + T_{i,i+1} \cdot b_{i+1} \tag{5.18}$$

After two time steps,

$$b_i|t_2 = T_{i,i-1} \cdot (b_{i-1}|t_1) + T_{i,i} \cdot (b_i|t_1) + T_{i,i+1} \cdot (b_{i+1}|t_1) \tag{5.19}$$

The *relation 5.19* can be substituted with *relation 5.18* and can be written as follow

$$\begin{aligned} b_i|t_2 = \quad & T_{i,i-1} \cdot (T_{i-1,i-2} \cdot b_{i-2} + T_{i-1,i-1} \cdot b_{i-1} + T_{i-1,i} \cdot b_i) + T_{i,i} \cdot (T_{i,i-1} \cdot b_{i-1} + \\ & T_{i,i} \cdot b_i + T_{i,i+1} \cdot b_{i+1}) + T_{i,i+1} \cdot (T_{i+1,i} \cdot b_i + T_{i+1,i+1} \cdot b_{i+1} + T_{i+1,i+2} \cdot b_{i+2}) \end{aligned} \tag{5.20}$$

The *relation 5.20* can be represented in a concise form as

$$b_i|t_2 = \sum_{j=i-1}^{i+1} (T_{ij} \cdot (T_{j,j-1} \cdot b_{j-1} + T_{j,j} \cdot b_j + T_{j,j+1} \cdot b_{j+1})) \tag{5.21}$$

For subsequent generalization, we define a function $\mathcal{J}$ where $\mathcal{J} = 1(0)$ if $T_{i,j} = 1(0)$. Consequently,

$$b_i|t_2 = \sum (\mathcal{J} \cdot (T_{j,j-1} \cdot b_{j-1} + T_{j,j} \cdot b_j + T_{j,j+1} \cdot b_{j+1})) \tag{5.22}$$

From the relation it is clear that the state of b_i depends for its evaluation after 2 time steps on j^{th} row only if the value of $\mathcal{J} = 1$. We denote $[dj]$ as the set of rows on which the evaluation of b_i depends. Consequently, the membership criteria of any j to be in $[dj]$ is determined by the value of corresponding $\mathcal{J}$ being one.

Generalizing the above formulation, after d time steps we have

$$b_i|t_d = T_{i,i-1} \cdot (b_{i-1}|t_{d-1}) + T_{i,i} \cdot (b_i|t_{d-1}) + T_{i,i+1} \cdot (b_{i+1}|t_{d-1}) = 0 \tag{5.23}$$

Following the same methodology expressed through (*relation 5.20 & 5.22*) the *relation 5.23* can be written in concise form as

$$b_i|t_d = \sum(\mathcal{J}(T_{j,j-1} \cdot b_{j-1} + T_{j,j} \cdot b_j + T_{j,j+1} \cdot b_{j+1}) = 0, \quad \mathcal{J} = 1(0) \quad \text{if} \quad \prod_{a=1}^{d-1} T_{l_a,k_a} = 1(0) \tag{5.24}$$

where

$$\{l_1 = i, k_{d-1} = j, l_{a+1} = k_a, where \quad k_a = \{l_a \ or \ l_a - 1 \ or \ l_a + 1\}\} \tag{5.25}$$

As per *Phase 1* result, $T_{i,i+1} = 0$. Therefore, from *relation 5.24 & 5.25*, the value of $\mathcal{J}$ can be 1 only if k_1 is either $\{i-1, i\}$. Subsequently, from *relation 5.25* k_2 has to be either $\{i-2, i-1, i\}$. In this fashion, we can show that the upper limit of k_{d-1} is i. *Hence,* $\forall j \in [dj]$, $j \leq i$, *that is evaluation of* b_i *as per the* relation 5.24 *depends on rows having index less than or equal to* i

Phase 3 : Establishes the rank of the set of $[dj]$ **rows as** $|dj|$ **- 1** ($|dj|$ indicates the cardinality)

We reorient the relation by clubbing each of the $b_j s$ together. The *relation 5.20* is rearranged in this line and reproduced below.

$$\begin{aligned} b_i|t_2 = \ & T_{i,i-1} \cdot T_{i-1,i-2} \cdot b_{i-2} + (T_{i,i-1} \cdot T_{i-1,i-1} + T_{i,i} \cdot T_{i,i-1}) \cdot b_{i-1} + (T_{i,i-1} \cdot T_{i-1,i} + T_{i,i} \cdot T_{i,i} \\ & + \ T_{i,i+1} \cdot T_{i+1,i}) \cdot b_i + (T_{i,i} \cdot T_{i,i+1} + T_{i,i+1} \cdot T_{i+1,i+1}) \cdot b_{i+1} + (T_{i,i+1} \cdot T_{i+1,i+2}) \cdot b_{i+2} \end{aligned} \tag{5.26}$$

The relation can be rewritten in closed form as

$$b_i|t_2 = \sum_{k=i-2}^{k=i+2} \Big(\sum_{j=max(k-1,i-1)}^{min(k+1,i+1)} (T_{i,j} \cdot T_{j,k}) \Big) \cdot b_k \tag{5.27}$$

For subsequent generalization, we use the function $\mathcal{J}$ where $\mathcal{J} = 1(0)$ if $T_{i,j} = 1(0)$.

$$b_i|t_2 = \sum_{k=i-2}^{k=i+2} \Big(\sum_{j=max(k-1,i-1)}^{min(k+1,i+1)} (\mathcal{J} \cdot T_{j,k}) \Big) \cdot b_k \tag{5.28}$$

Generalizing for d time steps

$$b_i|t_d = \sum_{k=i-d}^{k=i+d} \Big(\sum_{j=max(k-1,i-d+1)}^{min(k+1,i+d-1)} (\mathcal{J} \cdot T_{j,k}) \Big) \cdot b_k = 0 \quad \mathcal{J} = 1(0) \quad if \quad \prod_{a=1}^{d-1} T_{l_a,k_a} = 1(0) \tag{5.29}$$

where as per *relation 5.25*,

$$\{l_1 = i, k_{d-1} = j, l_{a+1} = k_a, where \quad k_a = \{l_a \ or \ l_a - 1 \ or \ l_a + 1\}\}$$

that is, k^{th} columns of the $[dj]$ rows are clubbed together to form the coefficient of b_k. (Since it is a three neighborhood T matrix, hence the non-zero rows of the k^{th} column are $k-1$, k & $k+1$, so only those three rows are significant.) As per the result of Phase 2, the maximum value of $j \in [dj]$ is i. Hence from *relation 5.29* maximum value of $k = i$.

The *relation 5.29* is a set of simultaneous equation, the variables of which are denoted by b_k. The value of the simultaneous equation can be zero if each of the coefficient $(\sum(\mathcal{J} \cdot T_{j,k}))$ is zero or there exists a dependency among the b_k bits. However, a dependency will not exist because according to definition, the dependency vector binding the vector subspace has dependency between b_i & b_{i+3} and b_{i+3} is not present in *relation 5.29*.

Therefore, the relation follows

$$(\sum(\mathcal{J} \cdot T_{j,k})) = 0, \quad \forall k \tag{5.30}$$

The expression $(\sum(\mathcal{J} \cdot T_{j,k}))$ implies summation of all the elements of the dependent $[dj]$ rows over the k^{th} column.
Hence from relation 5.30*, it directly follows that among the* $[dj]$ *rows there is at least one dependent row.*

Phase 4 : Sums up and shows the contradiction

Similarly, with the help of the information $T_{i+3,i+2} = 0$ (as noted in Phase 1 of the proof), we can show that the rows upon which b_{i+3} is dependent after d time steps are $\geq i+3$ and among these rows, there is at least one dependent row.

So from the above two instances there are at least two separate dependent rows in the T-matrix and therefore, we can conclude $rank(T) \leq n-2$. But the rank, as per *relation 5.14*, is $n-1$. Hence there is a contradiction.

Therefore, dependency vector of the form $\hat{DV} = [\ 1 \ \cdots \ 1 \ 0 \ 0 \ 1 \ \cdots \ 1\ ]$ cannot exist. □

Example 5.5 The examples of Dependency Vectors which cannot be generated by a three neighborhood $MACA$ are $< 100011 >$,$< 1001001 >$. In both the cases, there are two or more zeroes between a pair of 1's. Two legal dependency vectors are $< 101011 >$,$<$ $11010111 >$ etc. □

The value of $|P(w,n,m)|$ for $m = 1$ can be derived from the results of following two lemmas. The first lemma calculates the total number of $MACA(n,m)$. The second lemma enumerates the total number of w-weighted patterns these $MACA(n,m)$ can produce. Subsequently, we perform the average function.

Lemma 5.9 *The total number of Dependency Vectors* $(\hat{N}(n,1))$ *for two attractor basins*

(m=1) MACA is given by

$$\hat{N}(n,1) = \sum_{\tilde{w}=1}^{n} |D(\tilde{w})| \tag{5.31}$$

where $|D(\tilde{w})|$ *denotes the set of* Legal Dependency Vector *of weight* $\tilde{w}$.

Proof : As per the *Theorem 5.8*, in all the *legal dependency vector* $D(\tilde{w})$, the $1's$ are placed in such a way so that there are no two or more consecutive zeroes in between a pair of 1's. So the number of legal $|D(\tilde{w})|$ is given by

$$|D(\tilde{w})| = \sum_{k=0}^{\acute{k}} C_k^{\tilde{w}-1}.(n-\tilde{w}-k+1) \tag{5.32}$$

where $\acute{k} = Min\{(\tilde{w}-1),(n-\tilde{w}+1)\}$.

Hence the total number of Dependency Vectors ($\hat{N}(n,1)$) - that is, the number of $MACA(n,1)$ each having a different zero basin configuration is given by

$$\hat{N}(n,1) = \sum_{\tilde{w}=1}^{n} |D(\tilde{w})|$$

□

$|P_{\hat{N}(n,1)}(w,n,1)|$, as defined under **Terminology**, denotes the total number of patterns with weight w in $\hat{N}(n,1)$ number of n-bit vectors of $(n-1)$ dimensional subspace.

Lemma 5.10 *In two attractor basins MACA, the total number of patterns with weight w denoted as* ($|P_{\hat{N}(n,1)}(w,n,1)|$) *is given by*

$$|P_{\hat{N}(n,1)}(w,n,1)| = \sum_{\tilde{w}=1}^{n} |D(\tilde{w})| \cdot |P(w,n,1,\tilde{w})| \tag{5.33}$$

Proof : As per *Lemma 5.7*, each member of $D(\tilde{w})$ has $|P(w,n,1,\tilde{w})|$ (*Lemma 5.7*) number of patterns with weight w. So, in all possible vector space $\hat{N}(n,1)$, the number of patterns with weight w will be

$$|P_{\hat{N}(n,1)}(w,n,1)| = \sum_{\tilde{w}=1}^{n} |D(\tilde{w})| \cdot |P(w,n,1,\tilde{w})|$$

□

Theorem 5.11 *The value of* $|P(w,n,1)|$ *is given by*

$$|P(w,n,1)| = \frac{\sum_{\tilde{w}=1}^{n} |D(\tilde{w})| \cdot |P(w,n)|(d(\tilde{w}))}{\sum_{\tilde{w}=1}^{n} |D(\tilde{w})|} \tag{5.34}$$

Proof : The value of $|P(w,n,1)|$ is given by $|P(w,n,1)|=\frac{|P_{\hat{N}(n,1)}(w,n,1)|}{\hat{N}(n,1)}$. The rest of the proof directly follows from that of *Lemma 5.9* and *5.10*, - that is, from *Equation 5.31* and *5.33*. □

5.3.2 Computation of $|P(w,n,m)|$

Based upon the results of $|P(w,n,1)|$, the computation of $P(w,n,m)$ is initiated. The concept of Legal Dependency String in place of Legal Dependency Vector to enumerate the number of 2^m attractor n-cell $MACA$. The following lemma characterizes Legal Dependency String.

Lemma 5.12 *A* legal dependency string *has the constituent* dependency vector *placed in non-overlapping positions.*

Proof : From *Lemma 5.3* & *Fig. 5.3*, an $MACA(n,m)$ is so formed that the constituent T_i matrix, each representing an $MACA(n_i,1)$ are placed in block diagonal form. So each T_i has no dependency or overlap with any T_j. Since each T_i represents a 2-attractor $MACA$, corresponding to each T_i, there is a dependency vector. Hence, the $MACA(n,m)$ is concatenation of m non-overlapping dependency vector. □

Example 5.6 An example legal *dependency string* is [101012020], whereas the following *dependency string* [12010], where 1 & 2 are interleaved is illegal. The vectors of the 0 basin of an $MACA$ will not generate such illegal *dependency string.* □

To calculate $P(w,n,m)$, we first calculate the number of legal dependency string. It specifies the number of $MACA(n,m)$ with distinct 0 basins and is denoted by $(\hat{N}(n,m))$. Further, $|P_{\hat{N}(n,m)}(w,n,m)|$ - the total number of w-weighted pattern formed in a 0-basins of these $MACA(n,m)$. Finally, we measure the average.

Computation of $\hat{N}(n,m)$: The essential steps are elaborated below.

- Since a *Legal Dependency String(LDS)* is formed from m non-overlapping *Legal Dependency Vector*, at a particular partition configuration, the total number of LDS is the product of distinct LDV in each partition.
- For example, let us take an LDS comprising of 3 LDV of length 10, then a particular partition configuration is (3, 3, 4). Let the number of distinct LDV at each partition be 7, 7, 9 respectively. Therefore, the total number of $LDV = 7 \times 7 \times 9 = 441$. We now elaborate the concept of Distinct LDV.
- *Distinct Dependency Vector :* An example is given to clarify the nature of the problem. Suppose a *Dependency String* is [1010202], the partition point can be considered as $(4,3)$ as well as $(3,4)$.
- In order to ensure uniqueness, we impose the condition that given a partition an unique $MACA$ is one which has a 1 in the last position. Therefore, the above *Dependency String* will be considered as a member of partition $(3,4)$ only. Such restriction is not needed in the last partition.

- *Number of n_i-bit LDV in the last position :* It will be equal to $\hat{N}(n_i, 1)$ - $\hat{N}(n_i - 1, 1)$, where $\hat{N}(n_i, 1)$ is all possible LDV of size n_i and $\hat{N}(n_i - 1, 1)$ is all possible LDV with size n_i - 1, that is, it doesn't have 1 in the last position. Since the last position doesn't have the restriction.

- Therfore,

$$\hat{N}(n, m) = \sum_{Perm(n)} (\prod_{l=1}^{m-1} \hat{N'}(n_l, 1) \times \hat{N}(n_m, 1)) \tag{5.35}$$

where

$$\hat{N'}(n_l, 1) = \hat{N}(n_l, 1) - \hat{N}(n_l - 1, 1) \tag{5.36}$$

and $Perm(n)$ is all possible partition configuration.

Computation of $|P_{\hat{N}(n,m)}(w, n, m)|$:

- At each partition configuration, the number of unique $MACA(n_i, 1)$ - $\hat{UN}(n_i, 1)$ at each partition has $|P_{\hat{UN}(n_i,1)}(w_j, n, m)|$, $j = 0 \cdots n$ number of w_j weighted pattern.

- Let w_1, w_2, $\cdots$ $w_m = w$. Since each w weighted pattern is formed from concatenation of w_j patterns (*Theorem 5.5*, therefore, the total number of w-weighted pattern is the cross product of each $|P_{\hat{UN}(n_i,1)}(w_j, n, m)|$

- That is, number of w-weighted pattern in the 0-basins of $MACA$, given a particular partition configuration and a particular combination of $w_j s$, such that $\sum w_j = w$ is $\prod_{i=1}^{m} |P_{\hat{UN}(n_i,1)}(w_j, n, m)|$.

- This is summed over all possible combination of w such that $\sum w_j = w$. Therefore, number of w-weighted pattern in the 0-basins of $MACA$, given a particular partition configuration is $\sum_{Comb(w)} \prod_{i=1}^{m} |P_{\hat{UN}(n_i,1)}(w_j, n, m)|$.

- Hence, summing over all partition configuration,

$$|P_{\hat{N}(n,m)}(w, n, m)| = \sum_{Perm(n)} \sum_{Comb(w)} \prod_{i=1}^{m} |P_{\hat{UN}(n_i,1)}(w_j, n, m)| \tag{5.37}$$

The average number of w weighted patterns in each 0-basin of an $MACA(n, m)$ is specified by the following theorem.

Theorem 5.13 *The value of $|P(w, n, m)|$ is given by*

$$|P(w, n, m)| = \frac{\sum_{Perm(n)} \sum_{Comb(w)} \prod_{i=1}^{m} |P_{\hat{UN}(n_i,1)}(w_j, n, m)|}{(\hat{N}(n, m))} \tag{5.38}$$

Proof : The value of $|P(w, n, m)|$ is given by $|P(w, n, m)| = |P_{\hat{N}(n,m)}(w, n, m)| / \hat{N}(n, m)$. The rest of the proof directly follows from *Equation 5.37*. □

5.4 Synthesis of MACA through Genetic Algorithm

The previous two sections have detailed the behavior of the $MACA$ basins. It has been seen that the $MACA$ basins behave as natural clusters. This section details the methodology synthesizing a desired $MACA$ (T matrix) that can memorize the given set of patterns in its attractor basins. The number of attractor basins (say k) of the $MACA$ may be fixed or it can vary depending upon the design requirement. We have employed evolutionary algorithm to design the desired $MACA$. The following discussion reports the methodolgy to evolve $MACA$ through Genetic Algorithm. Genetic Algorithm performs directed search through the pool of $MACA(n, m)$. We discusses the directed search process for two scheme - *Scheme 1* - where m is fixed and *Scheme 2* - where m varies.

The basic structure of GA revolves around the concept of evolving the successive solutions according to its fitness. The fitness function is defined according to the functionality for the purpose of which $MACA$ is evolved. In the next chapter, we elaborate two different functions of $MACA$ - classification and associative memory modeling. The fitness functions of each individual cases are elaborated. Moreover, each solution (the solution is an $MACA$) has to be encoded in bit string format (chromosome) for the purpose of evolution. Hence, the prerequisite for GA evolution is to encode an $MACA$ in bit string format.

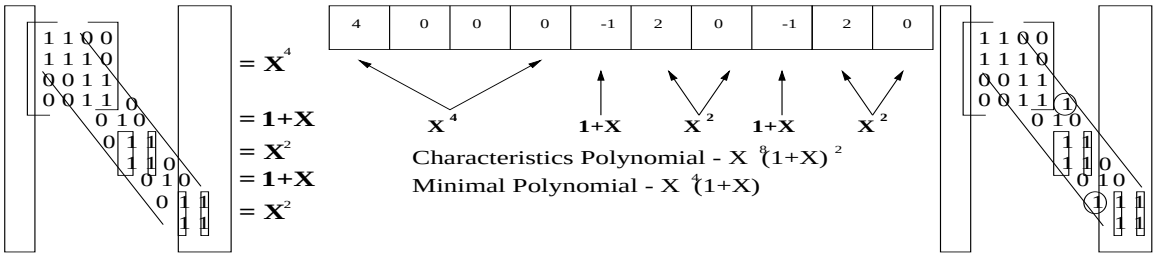
The most obvious method of encoding an $MACA$ would be to encode the binary counterpart of CA rules employed for each cell in bit string. Researcher while combining GA with CA has applied this encoding scheme [36, 5, 26, 25, 33]. But the problem which lies ahead is that since $MACA$ is a very specialized class of Linear CA, a different sequencing of the same rules produces CA which do not preserve $MACA$ characteristics. For example, a 5 cell CA with Rule vector [102 60 60 90 204] forms an $MACA$ (*Fig 5.2*) while [204 102 90 60 60] is not an $MACA$ even though both employ the same set of rules. Hence, encoding the rules of $MACA$ would result in the undesirable effect of taking the solution set out of the $MACA$ domain. In order to circumvent this problem we have introduced a special encoding scheme also termed as pseudo-chromosome format to generate the solution of the problem with GA framework.

The encoding scheme is described next.

5.4.1 The Encoding Scheme

The encoding scheme encodes an n-bit $MACA$ having 2^m attractors. The characteristic polynomial of the T matrix of an $MACA$ can be represented by a sequence of $x^{d_i}{}'s$ and $(1+x)'s$. The T matrix of *Fig 5.8(a)* represents an $MACA$. Only tri-diagonal elements of the matrix is shown in the figure since the remaining elements are zero. The figure shows the blocks of x^d & $(1+x)'s$ derived from T matrix.

The sequence of $x^{d_i}{}'s$ and $(1+x)'s$ is encoded in a chromosome like format. It gives a semblance of the chromosome and hence termed as *pseudo-chromosome format*. It is a string of n bits where (a) d_i positions occupied by a x^{d_i} is represented by d_i followed by $(d_i - 1)$ zeros (for example, $x^3 = [300]$), and (b) $(1+x)$ is represented by -1. The *pseudo-chromosome format* of the $MACA$ is illustrated in *Fig. 5.8b*.



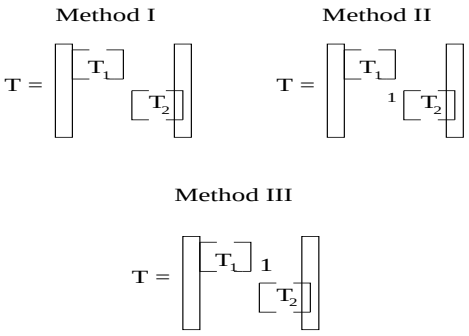
(a). T Matrix(1) & its Elementary Divisor Sequence (b). Pseudo-Chromosome Format (c). T Matrix(2)

Figure 5.8: MACA encoding for GA formulation

The three major functions of *GA* - Random Generation of Initial Population, Crossover and Mutation operate on this encoded format. Each of the function is elaborated one by one. Variations in each function with respect to *Scheme 1* - Scheme where the number of attractor basin is fixed and *Scheme 2* - Scheme where the number of attractor basin varies is simultaneously highlighted during the discussions.

5.4.2 Random Generation of the Initial population(IP)

To form the population, it must be ensured that each solution (randomly generated) is an n-bit $MACA$ with 2^m number of attractors where $m \geq log_2(k)$. The value of m may be fixed (*Scheme 1*) or it can be randomly generated (*Scheme 2*). Based on the value of m we generate a sequence of the elementary divisors and consequently synthesize the $MACA$. The following example illustrates the underlying principle.



[$\hat{T}_1$]& [$\hat{T}_2$] in Block Diagonal Form each represents an Elementary Divisor (either x^d or $(1+x)$)

Note: While using Method II, III, one restriction to be maintained - $\hat{T}_2$ & $\hat{T}_1$ both cannot be $(1+x)$.

Figure 5.9: Different methods to arrange two matrices

Therefore, the characteristic polynomial $x^8(1+x)^2$ is randomly arranged as $x^4 \cdot (1+x) \cdot x^2 \cdot (1+x) \cdot x^2$. In the synthesis scheme, each elementary divisor is separately synthesized (*Algorithm 4.7*). The T matrix is finally synthesized by randomly using alternative schemes of arranging submatrices in block diagonal form/block triangular form [49], each submatrix corresponds to an elementary divisor. The schemes are illustrated in *Fig 5.9. Method II* & *III* places two sub-matrices in block triangular form while *Method I* places two sub-matrices in *Block Diagonal Form. Method II* & *III* cannot be used if each sub-matrices represent $(1+x)$. This restriction ensures our construction policy (*Fig. 5.3*) which specifies each block represents a two attractor $MACA$, hence contains a single $(1+x)$ factor.

Adopting different schemes, the same sequence of the elementary divisors produces two different T matrices as (*Figure 5.8 a* & c - the positions where they are differing is encircled in *Figure 5.8 (c)*). However, the pseudo-chromosome format of both of them are same.

[Note : The structure of the attractor basins of the $MACA$ changes with the change of the sequencing of the elementary divisor, as well as the synthesis of T matrix from the same sequence of elementary divisor. In this connection. along with the pseudo chromosome format, each solution stores the underlying T matrix.]

The algorithm to randomly generate an $MACA$ is elaborated next. Any steps specific to *Scheme 1* or *2* is categorically mentioned.

Algorithm 5.1 Rand_gen_MACA

Input: n - the size of $MACA$, m: where 2^m is the number of attractors. - **Scheme 1**

Input: n - the size of $MACA$ - **Scheme 2.**

Output: T matrix representing an $MACA$ with 2^m attractor. - **Scheme 1**

Output: T matrix representing an $MACA$. - **Scheme 2**

Step : Randomly Generate the value of m. - **Scheme 2**

Step 1: Randomly divide $(n-m)$ into $d_1, d_2 \cdots d_p$ to form the corresponding $d_i's$.

Step 2: Synthesize $\hat{T}_i$ matrix from each elementary divisors x^{d_i}.

Step 3: Synthesize $\hat{T}_j$ matrix from each elementary divisors $(1+x)$.

Step 4: Randomly arrange the sequence of $\hat{T}_i$ and $\hat{T}_j$s, forming the *pseudo-chromosome format.*

Step 5: Select either *Method I, II* or *III* of *Fig. 5.9* to arrange a T_i and T_j to form the final T matrix representing the output $MACA$

The crossover and mutation algorithms implemented are similar in nature to the conventional one normally used for GA framework. We use single point crossover technique and single bit mutation technique. However, it demands some repair of the chromosome after mutation/crossover to maintain the integrity of the format of the chromosome (*Fig. 5.8 (b)*) generated. The repair algorithm differs in case of cross-over algorithm for *Scheme 1* & *2*. However, it is same in case of mutation. The modification scheme is next outlined for mutation.

5.4.3 Mutation Algorithm

The mutation algorithm selects a chromosome of higher fitness from the Present Population (PP) and generates a new chromosome for the for the population of the next generation(NP). The $MACA$, as illustrated in *Fig. 5.10*, is mutated at a single point. In mutation algorithm, an $(1+x)$'s position is altered. Some anomaly crops up due to its alteration. The anomaly is resolved to ensure that after mutation the new CA is also an $MACA$ that preserves the pseudo chromosome format illustrated in *Fig. 5.8.* The inconsistent format, as shown in the *Fig 5.10b* is the mutated version of *Fig 5.10a.* The inconsistency of the *pseudo-chromosome format* of *Fig 5.10b* can be resolved by replacing '4' and '2' with '3'. This leads to the consistent format of *Fig 5.10(c).* The similar

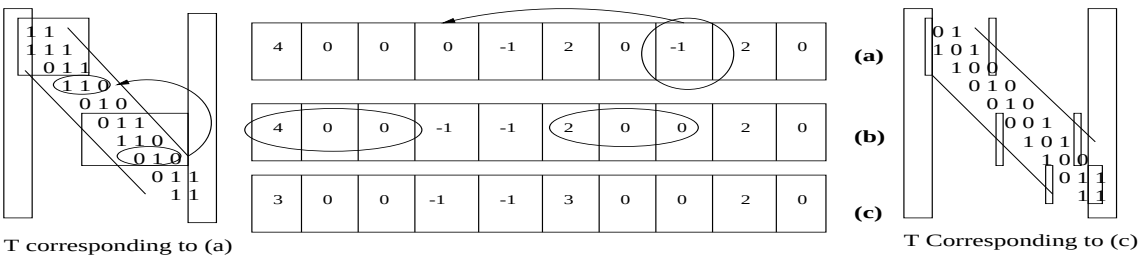


Figure 5.10: An Example of Mutation Technique

inconsistencies occur in the underlying matrix. The inconsistencies arising in the matrix corresponding to pseudo-chromosome format of *Fig 5.10(a)* in position are encircled. They are correspondingly resolved and we obtain *Matrix* corresponding to *Fig 5.10(c)*.

The algorithm is elaborated below.

Algorithm 5.2 Mutation Algorithm
Input: Randomly select one $MACA$ of PP.
Output: New $MACA$ for NP.
Step 1: Take the *pseudo-chromosome format* of the $MACA$.
Step 2: Randomly select a $(1+x)'s$ position (p_1) and put it in a non-$(1+x)$ position (p_2).
Step 3: Make necessary modifications at mutation points to maintain *pseudo-chromosome format* (*Section 7.4.1*) for the resulting $MACA$.

5.4.4 Crossover Algorithm

The crossover algorithm implemented is similar in nature to the conventional one normally used for GA framework with minor modifications as illustrated below. The algorithm takes two $MACA$ from the present population (PP) and forms the resultant $MACA$. Like a single point crossover, it sets a crossover point and each half about the crossover point is selected from the two respective $MACA$. Similar to mutation, after crossover reparing of the chromosome has to be done to maintain the integrity of the solution. The repair algorithm differs for *Scheme 1 & 2*. A single example is developed in two steps to illustrate the repair algorithm of both *Scheme 1 & 2*.

The *Fig 5.11* shows an example of the crossover process. Two $MACA$ in *pseudo-chromosome format* are shown in *Fig 5.11a & Fig 5.11b*. The T matrix corresponding to *Fig. 5.11a & Fig 5.11b* are also shown in the figure. Each of the participating $MACA$ has 2 $(1+x)$ factors, hence has 2^2 attractors. The first 6 symbols are taken from $MACA_1$, while the rest 4 symbols are taken from $MACA_2$. The violation in *pseudo chromosome format* is first explained for *Scheme 2*.

The resulting CA, as explained below violates the *pseudo-chromosome format* in 6^{th} position. (encircled in *Fig 5.11c.*). The *pseudo-chromosome format* has x^{d_i} represented by d_i followed by $(d_i - 1)$ zeros. But in the case of *Fig 5.11c*, we have 2 without any zero following it. This is a violation since the property of $MACA$ is not maintained. The

violation in the corresponding T matrix is also shown in the figure. So we take out the symbol and form a CA of elementary divisor x and adjust it according to the method suggested in *Fig. 5.9*. The adjustment in the chromosome and T-matrix is accordingly shown in *Fig 5.11d*. The resultant $MACA$ has three $(1+x)$ factors, that is, the resultant is an $MACA$ with 2^3 attractors. This conforms with requirement of *Scheme 2* but in *Scheme 1* there is one more restriction - the resultant $MACA$ should have 2^2 attractors. Hence, to satisfy this constraint, further repairment of the chromosome is needed. The repairment process is discussed next.

Since there is one extra $(1+x)$ factor, we randomly select a $(1+x)$. In the figure the $(1+x)$ in the last cell is selected (*Fig. 5.11d*) and it is converted into a factor of x^d. The last three cells are selected in the figure, and [2, 0, -1] is converted to [3, 0, 0] in *Fig. 5.11e*. The corresponding changes in the matrix are highlighted simultaneously.

The algorithm formalizes the above discussions.

Algorithm 5.3 Algorithm Cross-over
Input: *Randomly selected two $MACA$ ($MACA_1$ & $MACA_2$) from PP each having 2^m attractor.* - **Scheme 1**
Randomly selected two $MACA$ ($MACA_1$ & $MACA_2$) from PP. - **Scheme 2**
Output: *New $MACA$ having 2^m attractor.* - **Scheme 1**
Output: *New $MACA$.* - **Scheme 2**
Step 1: *Randomly generate a number q between 1 & n.*
Step 2: *Take the first q symbols from $MACA_1$ and the last $n-q$ symbols from $MACA_2$.*
Step 3: *Make the necessary modifications in the partition points to conform* pseudo-chromosome format *(*Section 7.4.1*) for the resulting $MACA$.*
Step 4: *Make the necessary modifications to make the number of attractor 2^m.*
- Scheme 1

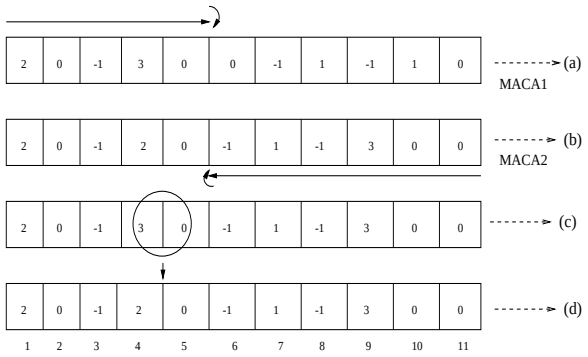


Figure 5.11: An Example of Cross-over Technique

5.4.5 Selection, Crossover and Mutation Probability

From the study of GA evolutions, we have set the associated parameters to derive the NP out of PP. The population size at each generation is set to 50. The crossover probability

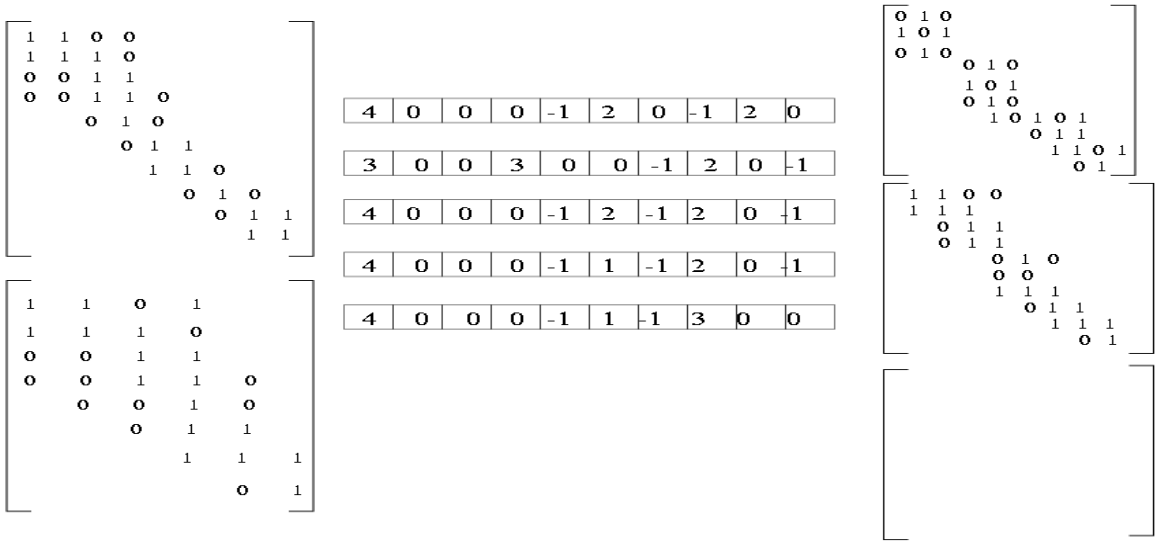


Figure 5.12: An Example of Cross-over Technique

(p_c) is set to 0.8, while the probability of mutation (p_m) is set as 0.001. We follow the elitist model of carrying forward 10 best solutions to the next generation. The Fitness Function is determined according to the requirement of the problem, two such requirement will be discussed in the next chapter.

5.5 Conclusion

Chapter 6

MACA Based Pattern Recognizer and Pattern Classifier

6.1 Introduction

This paper reports a new scheme of modeling an associative memory with linear Cellular Automata (CA). A special class of linear CA, Multiple Attractor Cellular Automata ($MACA$), has been used for the design. We introduce a concept of Hamming Hash Family (HHF) and realize the structure with $MACA$. The properties of HHF are explored to ease the efficient design of CA based associative memory. The complete analytical framework coupled with extensive experimental results has been provided. The following considerations have encouraged us to undertake the research.

The last few decades of the twentieth century have encountered massive advancement in computing technology. The computing speed has leaped from kilo to mega to giga and is now reaching the unthinkable terra flops. However, in spite of such leaps, machines have failed to emulate the stability of human brain and to match the ease with which human brains recognize patterns. Associative Memory organization of human brain is one of the most important ingredients which imparts this superiority. Consequently, researchers, practitioners, entrepreneurs from diverse fields are engaged themselves to develop sophisticated techniques to emulate associative memory.

Associative Memory model provides an elegant solution to the problem of identifying the closest match to the patterns learnt/stored [2]. The model, as shown in *Fig.8.1*, divides the entire state space into some pivotal points (say) a,b,c. The pivotal points (patterns) are assumed to be learnt by the machine during its training phase. A state close to a pivotal point is the noisy vector (pattern) associated with that specific pivotal point. The process of recognition of a pattern, with or without noise, amounts to traversing the transient path (*Fig.8.1*) from the given pattern to the closest pivotal point learnt. In effect, the time to recognize a pattern in associative memory model is independent of the number of patterns stored [2].

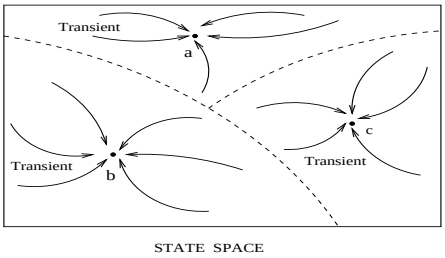


Figure 6.1: Model of associative memory with 3 pivotal points.

Since early 80's the model of associative memory has attracted considerable interest among the researchers [3, 30]. Both sparsely connected machine (Cellular Automata) and densely connected network (Neural Net) have been explored to design the associative memory model for pattern recognition [3, 5, ?]. The seminal work of Hopfield [?, ?] made a breakthrough by modeling a *recurrent*, *asynchronous*, *neural net* as an *associative memory* system. However, the complex structure of neural net with non-local interconnections has partially restricted its application for design of high speed low cost pattern recognition machine. As a result, there has been a series of work attempting to minimize the connectivity of the Hopfield net[][][], which resulted in the introduction of Cellular Neural Network [?, ?, ?]. In this light, solving the associative memory problem through a sparse network like Cellular Automata(CA) would be extremely helpful. This would not only make the underlying circuit simple but also have a natural digital $VLSI$ implementation of the whole circuitry providing the necessary speed and scalability required to design associative memory.

But although the search for alternative model around the simple structure of Cellular Automata (CA) [5, ?] continued there hasn't been any notable development. Recently a series of works on associative memory model designed with 3-neighborhood CA, using non-linear rules, have been reported [25, ?, 33, ?]. However, the results in these papers are largely experimental. The current work shows that associative memory can be modeled using even a smaller set of rules. The paper as well gives a solid analytical foundation regarding the capacity, basins of attraction of the CA based memory.

This work employs only the linear CA, using XOR logic to generate its next state function [?, ?], to model an associative memory. The design with linear CA is of special interest to the research community due its available analysis and synthesis tools based on linear algebra [?] and its simplest hardware architecture [?].

The powerful associative memory modeling tool. The entire model is built around a special class of CA, Multiple Attractor Cellular Automata ($MACA$), introduced in [?]. The detailed analysis of $MACA$ has provided the foundation to propose a new concept of hash family referred to as Hamming Hash Family (HHF). The property of the hash family - the probability of collision between a pair of patterns, hashed by a member of HHF, varies inversely with their hamming distance, - is exploited for the current design of associative memory model that performs complex computation like pattern recognition.

The preliminaries of CA based sparse network are introduced in *Section II*. In *Section III*, we report the basic scheme of design $MACA$ based associative memory model. The detailed analysis of the properties of Hamming Hash Family, employed for the design of associative memory, is reported in *Section IV*. Subsequently, in *Section V*, the performance of the proposed design is evaluated analytically. The *Genetic Algorithm* framework for evolution of $MACA$ is described in *Section VI*. *Section VII* reports the extensive experimentation performed to support and reinforce the theoretical analysis. We conclude the paper in *Section VIII*.

The internetworked society has been experiencing data explosion that is acting as an impediment in acquiring knowledge. Meaningful interpretation of these data is increasingly becoming difficult. Consequently, researchers, practitioners, entrepreneurs from diverse disciplines are trying to develop sophisticated techniques for knowledge extraction. Data classification forms one of the important arena of such research.

Data classification is the process of identifying common properties among a set of objects in a database. A classification model is built based on a predefined set of data classes. A sample set from the database, each member/tuple belonging to one of predefined classes, is used to train the model. The training is also termed as supervised learning of the classifier. Each member/tuple may have multiple features/attributes. The classifier is trained based on a specific feature. Subsequent to training, the model performs the task of correctly recognizing the class of any incoming pattern, that may or may not have been used during the training phase, Recognition of an input sample is done based on some metric, typically distance metric.

The essential prerequisite of designing the classifier for current information age is - it should have high throughput with less storage requirements. Further, low cost hardwired implementation of the scheme is becoming a very important criterion for on line real time applications. The CA based classifier proposed in this paper displays the following characteristics :

- A special class of CA referred to as Multiple Attractor Cellular Automata ($MACA$) is employed for design of the classifier.
- The desired $MACA$ has been evolved with a special type of Genetic Algorithm(GA) formulation.
- Theoretical analysis of the $MACA$ attractor basins establishes the capacity of the classifier to accommodate noise.
- The experimental results confirm that its classification efficiency is better than decision tree models.
- The classifier employs the simple computing model of three neighborhood Additive CA having very high throughput. Further, compared to conventional techniques [11], [30], [**?**], [**?**] developed for classification, the simple, regular, modular and local neigh-

borhood sparse network of Cellular Automata suits ideally for low cost $VLSI$ implementation.

The Cellular Automata (CA) preliminaries follow in the next section.

6.2 MACA as Associative Memory

An associative memory model is trained to get familiarized with some specific pattern set $\{\mathcal{P}_1, \cdots, \mathcal{P}_i, \cdots \mathcal{P}_k\}$. It can recognize an incoming pattern as $\mathcal{P}_i$ even if the pattern is distored with limited noise. If a new pattern $\mathcal{P}_{incom}$ is input to the system, the model identifies it as $\mathcal{P}_i$, where $\mathcal{P}_{incom}$ is the closest match to $\mathcal{P}_i$. The hamming distance between $\mathcal{P}_{incom}$ and $\mathcal{P}_i$ ($HD(\mathcal{P}_{incom}, \mathcal{P}_i)$) is the least and this is viewed as the measure of noise in a generic sense.

This section reports the design of associative memory with $MACA$. If an $MACA$ has to function as an associative memory storing $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_i, \cdots, \mathcal{P}_k\}$, each of its attractor basin should have one and only one of the given patterns of $\mathcal{P}$. While the $MACA$ is run for some time steps (maximum d) with $\mathcal{P}_{incom}$ as an input, it should return the pivot point $\mathcal{P}_i$ closest to it. That is, both $\mathcal{P}_{incom}$ and $\mathcal{P}_i$ should lie in the same α-basin of the $MACA$. The design guidelines for such an $MACA$ for associative memory are specified by the following two relations:

R1: Each attractor basin of the $MACA$ should contain one and only one pattern ($\mathcal{P}_i \in \mathcal{P}$) to be learnt.

R2: The Hamming Distance ($HD(\mathcal{P}_{incom},\mathcal{P}_i)$) between each state $\mathcal{P}_{incom} \in \mathcal{P}_i$-basin and $\mathcal{P}_i$ is lesser than the $HD(\mathcal{P}_{incom},\mathcal{P}_j)$ where $\mathcal{P}_j \in \mathcal{P}$, $\forall j = 1, 2, \cdots k, \&\ \ j \neq i$.

To satisfy **R1**, the design demands the construction of an $MACA$ with k attractors where no two pattern $\mathcal{P}_i$ & $\mathcal{P}_j$ lie in the same attractor basin att_i. In order to ensure the placement of patterns in separate attractors, the following relation has to be met.

$$T^d \cdot X \neq 0 \tag{6.1}$$

where the pattern set $X = \{x_l \mid x_l = P_i \oplus P_j\ \forall_{i,j=1}^{k}\ \ \&\ \ i \neq j\}$. T is the desired $MACA$ with k attractors, and d is the depth.

Relation 6.17 directly follows from *Theorem 5.2*. If two patterns P_i and P_j fall in different attractors, the pattern $x_l = P_i \oplus P_j$ (modulo-2 sum) should necessarily fall in a non-zero attractor basin. In effect x_l while multiplied by T^d yields a non-zero attractor.

An $MACA$ can act as an effective hash function. The total pool of $MACA$ forms a hash family and exhibits an interesting property - *the hash family maintains an inverse relationship between the collision of a pair of patterns while hashed and their hamming distance.*

The hash family having this property is referred to as *Hamming Hash Family* (HHF). The concept of HHF imparts an important property to the attractor basin of $MACA$. Since keys (patterns) that collide lie in the same attractor basin, the keys (patterns) which

lie close to each other in terms of hamming distance belong to the same basin. That is, if a set of patterns $\mathcal{P}$ satisfies **R1**, then **R2** naturally follows.

Therefore, the design of an $MACA$ based associative memory for a pattern set $\mathcal{P}$ boils down to design of a method of deriving T satisfying **R1** - that is, relation 6.17 for the given pattern set. The characterization of HHF in respect of the probability of collision and hamming distance between patterns is in the next section.

6.3 Performance Analysis

This section reports the performance analysis of $MACA$ based associated memory. The performance is measured with respect to two major aspects (i) Stability and (ii) Basins of Attraction.

6.3.1 Stability

Stability or *Memorizing Capacity (MC)* is defined as the number of patterns the machine can store. *Stability* gives the measure of the number of patterns (say k) for which all possible combination of patterns can be distinctly remembered.

To memorize any random pattern set $\mathcal{P}$, the prerequisite is that there should be at least one $MACA$ in which each pattern $\mathcal{P}_i \in \mathcal{P}$ falls in separate attractor basins. It implies that the entire set X, the set formed from XORing each pair of members of the pattern set $\mathcal{P}$, should satisfy *Relation 6.17*

$$T^d \cdot x \neq 0 \quad \forall x \in X$$

That is, any $x \in X$ should not lie in zero-basin. If the cardinality of P is k, then the maximum cardinality of the set X is $\tilde{k} = \frac{k \cdot (k-1)}{2}$

Let p_i be the probability of a pattern $x_i \in X$ to fall in zero basin. Then (1 - p_i) = q_i is the probability that the x_i is not in zero-basin. Therefore, the probability - patterns of X are not in the zero basin, is

$$Q = q_1 \times q_2 \times \cdots \times q_{\tilde{k}}. \tag{6.2}$$

To memorize k patterns, the candidate $MACA$ should have 2^m number of attractors, where $2^{m-1} < k \leq 2^m$. Let the number of all possible $MACA(n, m)$ be $\hat{N}(n, m)$. Then the number of $MACA$, capable of memorizing the pattern set, is given by

$$\tilde{N} = Q \cdot \hat{N}(n, m) \tag{6.3}$$

If each of the pattern sets $\mathcal{P}$ of cardinality k has to be stable, then for all values of Q, $\tilde{N}$ should be greater than 1. So in order to determine the lower limit, the task is to find the least value of Q.

The probability of weight w occurring in zero basin is

$$p_w = \frac{|\hat{P}(w, n, m)|}{2^{n-m}} \tag{6.4}$$

where 2^{n-m} is the number of patterns in the zero basin. $|\hat{P}(w,n,m)|$ is the expected number of patterns with weight w in the zero basin of $MACA(n,m)$. Consequently, the probability of non-occurrence is given by

$$q_w = 1 - p_w \tag{6.5}$$

The least value of Q is directly determined from the least value of q_w - that is, the least probability of a pattern with weight w not in the zero basin. It is observed that q_w is least while w is around $n/2$. Therefore, the *equation 6.3* can be rewritten as

$$\tilde{\mathcal{N}} = (least(q_w))^{\tilde{k}} \cdot \hat{N}(n,m) \tag{6.6}$$

The value of $\tilde{k}$ for which $\tilde{\mathcal{N}} > 1$ gives us the absolute stability value (k) of the machine.

Table 6.1: Stability Analysis for different value of n

No of cell (n)	No of Random Patterns
10	9
20	13
30	17
40	22
50	25
60	28
70	30
80	32
90	34
100	36

Table 6.1 reports the nature of absolute stability. The absolute stability is very high for smaller values of n. However, at larger values it stabilizes roughly at $0.2n$.

6.3.2 Basins of Attraction

For effective pattern association, the patterns stored in an associative memory must act as the attractors. Ideally a given initial state will relax to the nearest attractor with which it has the least hamming distance. Consequently, the basin of attraction of an attractor includes the pattern set which are closest to that attractor.

In an $MACA$, the basin of attraction is not ideal. The amount of noise that can be recovered is estimated from the Expected Occurrence Graph. The Expected Occurrence reflects the occurrence of pattern set with weight w in the zero basin and is denoted as

$$EO(P(w,n,m)) = \frac{|\hat{P}(w,n,m)| \times 2^m}{{}^nC_w}$$

$|\hat{P}(w,n,m)|$ is the expected number of patterns with weight w in the zero basin of $MACA(n,m)$.

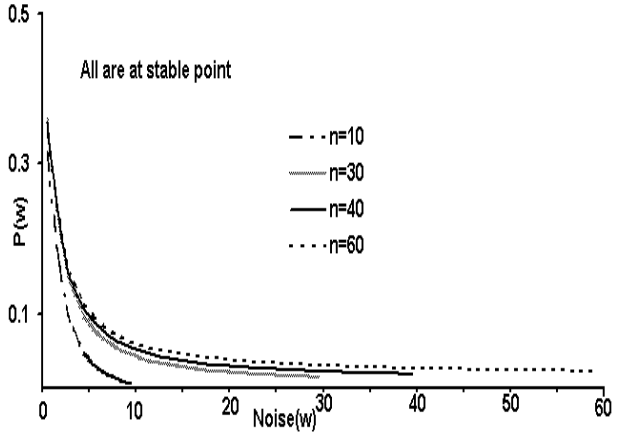


Figure 6.2: Theoretical Estimation of Noise Accommodating Capacity Of MACA at stable point for n = 10, 30, 40, 60

The total number of n-bit patterns with weight w is ${}^nC_w$. Therefore, the fraction of patterns having weight w present in the zero basin is given by

$$pn_w = \frac{|\hat{P}(w,n,m)|}{{}^nC_w} = \frac{EO(P(w,n,m))}{2^m} \tag{6.7}$$

Fig.6.2 shows the basin of attraction (theoretical value) of the $MACA$ with respect to the set of patterns at stable point for n= 10, 30, 40 & 60. It is observed that any attractor $\mathcal{P}_i$ has an ideal dominance behavior, that is, it has better attraction capability for patterns $\mathcal{P}_{incom}$ if it is close to it in terms of Hamming Distance.

Fig.6.3 depicts the basin of attraction (theoretical value) of the $MACA$ of size n = 40, for different k (number of patterns stored). Since an $MACA$ basin only breaks into 2^m attractor ($m = 1, 2, \cdots, n$), therefore pattern set of cardinality k, where $2^{m-1}+1 < k \leq 2^m$, will have same error recovery capability. However, as we decrease the number of patterns to be stored beyond 2^m, the error recovery capability increases.

The efficiency of basin of attraction is further extended by exploring multiple $MACA$. The next subsection elaborates the scheme.

6.3.3 Enhancing Performance - Multiple $MACA$

In this section, we present a technique for enhancing the error recovery capability. For pattern recognition instead of using a single $MACA$ we use multiple $MACA$ each classifying the same set of patterns. Although the percentage of patterns classified has a strong bias towards lower hamming distance pattern but the rate of recovery is less mainly because a single basin hence the zero basin has only 2^{n-m} patterns. That is, for example, a pattern $\mathcal{P}_{incom}$ may have HD($\mathcal{P}_{incom}$, $\mathcal{P}_1$) =1 and HD($\mathcal{P}_{incom}$, $\mathcal{P}_2$) =2, where ($\mathcal{P}_1$,$\mathcal{P}_2$) are pivot points. Therefore, $p_1 = 0.3$ while $p_2 = 0.2$, hence have more chance of colliding with $\mathcal{P}_{incom}$. But the pattern $\mathcal{P}_{incom} \oplus \mathcal{P}_1$ may not be present in the zero basin, hence the

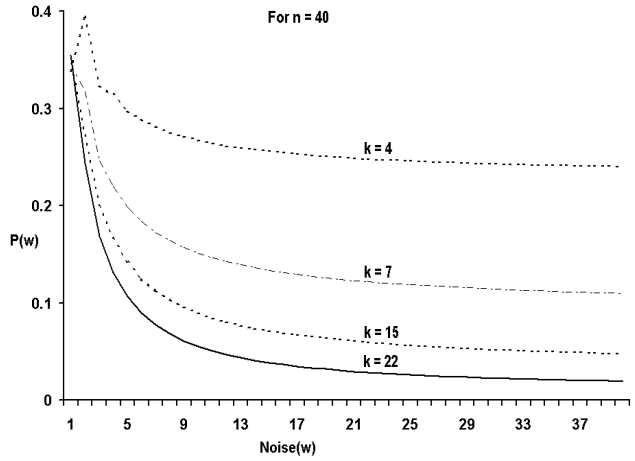


Figure 6.3: Theoretical Estimation of Noise accommodating capacity of MACA($n = 40$) at various k

probabilistic advantage will be lost.

To take advantage of the probabilistic approach, using of multiple $MACA$ where each CA classifies the same set of patterns is pursued. The decision of the pivot of the pattern is decided by the majority function. It thus enlarges the zero basin and allows the bias of smaller weights to take advantage.

The algorithm to design Multiple $MACA$ proceeds in the following manner. Let the number of $MACA(n, m)$ taken be $\mathcal{N}$, where each $MACA$ has classified the patterns in distinct attractors. Any incoming pattern $\mathcal{P}_{incom}$ is hashed upon all the $\mathcal{N}$ $MACA$. At each instance of hashing, $\mathcal{P}_{incom}$ collides with a pattern $\mathcal{P}_i \in \mathcal{P}$. Each $\mathcal{P}_i \in \mathcal{P}$ records the count of the number of times it collides with $\mathcal{P}_{incom}$.

Once the process of hashing in all the $\mathcal{N}$ $MACA$ is completed, the maximum number of times with which $\mathcal{P}_{incom}$ has collided is recorded as the identifiable pattern.

Stating it in Algorithmic form

Algorithm 6.1 Find Majority
Input : Incoming Pattern : $\mathcal{P}_{incom}$
Output : Pivot Point : $\mathcal{P}_i$
for j *= 1 to* $\mathcal{N}$
Hash $\mathcal{P}_{incom}$ *with* $MACA_j$.
Pick $\mathcal{P}_i$ *with which* $\mathcal{P}_{incom}$ *has collided the most number of times in the process of hashing.*
Output $\mathcal{P}_i$.

The *Figure 6.4* shows the enhanced capacity of the pattern recognizer for $n = 40$, $k = 22$. The graph shows that designing the Pattern Recognizer System using multiple CA provides an ideal basin of attraction. It is seen that as we increase the value of $\mathcal{N}$, the

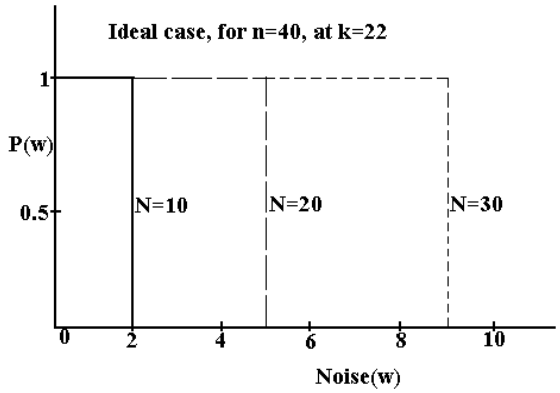


Figure 6.4: Number of MACA vis-a-vis Noise

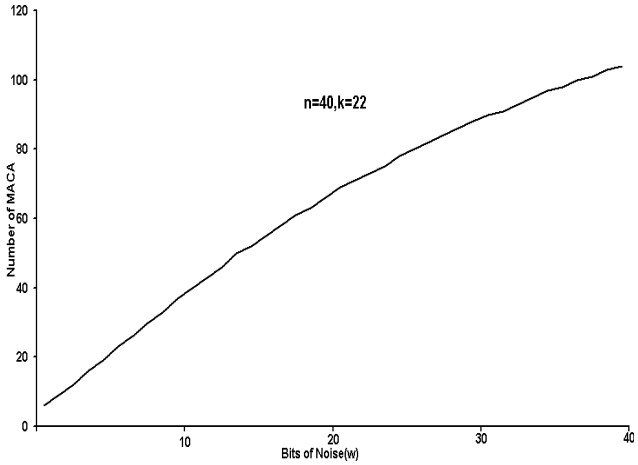


Figure 6.5: Number of MACA required to accommodate fixed bit noise(w = 1, 2, 3)

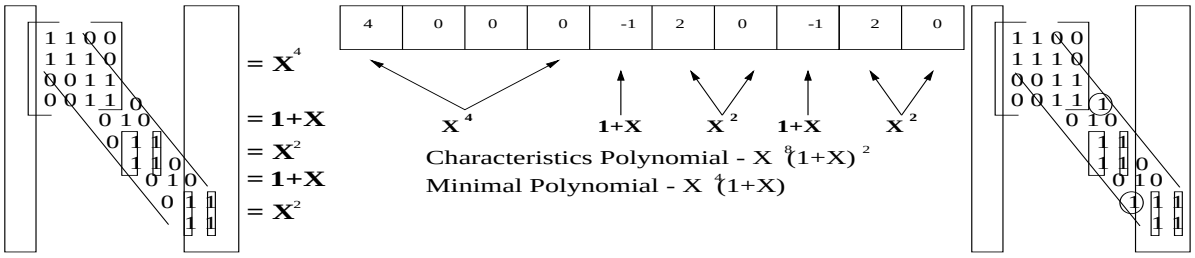
radius of basin of attraction increases for example, we can recover upto .... bits of noise if that attractor is closest.

Figure 6.5 shows the number of $MACA$ - $\mathcal{N}$ required to recover w (= 1, 2, 3 $\cdots$ 40) bits of noise respectively for $n = 40$, $k = 22$. It is seen that the number of $MACA$ required to recover the noise slowly increases with the increase in value of w.

The method by which we obtain the graphs of *Fig. 6.4 and 6.5* is next discussed.

Computing the value of $\mathcal{N}$

Suppose the basin of attraction of the machine is w, that is it can recover noise upto weight w. To design such machine, we have to ensure that in order to recover the proper pivot of a pattern $\mathcal{P}_{incom}$, which is at a distance w from the nearest attractor $\mathcal{P}_i$, the number of collision of $\mathcal{P}_{incom}$ with $\mathcal{P}_i$ should be more than ones. To ensure that the collision is at



(a). T Matrix(1) & its Matrix(2) (b). Pseudo-Chromosome Format Elementary Divisor Sequence (c). T

Figure 6.6: MACA encoding for GA formulation

least two number of times, so that it can be distinguished from other pattern set $\mathcal{P}_j \in \mathcal{P}$, (which has less collision probability, the number of collision with $\mathcal{P}_j$ will be less than 2) the following relation has to hold.

$$pn_w \times \mathcal{N} = 2 \tag{6.8}$$

Hence from the equation both the value of w(*Figure 6.4*) as well as $\mathcal{N}$(*Figure 6.5*) is computed keeping the other constant.

Moreover, with the introduction of Multiple CA, the recovery of noise beyond the threshold limit w cannot be done at all because the incoming pattern collides once with all the pivot point, making it impossible to differentiate by majority function. Hence the *graph 6.4* shows the noise recovery capacity slides drastically.

Let $n = 10$ and $m = 2$.

6.4 Experimental Observation

The experiments are performed to evaluate the stability point and the Basins of Attraction both using single and Multiple $MACA$. The experimental results are reported next.

6.4.1 Stability

The experiment for finding out the memorizing capacity was performed for different value of n. The *stable* point for different value of n was enumerated through experimentation. Random Pattern Set $\mathcal{P}$ is taken as input. The cardinality of $\mathcal{P}$ is increased progressively. At each cardinality point 30 random sets are taken as input. If more than one fails to generate $MACA$ for the selected $\mathcal{P}$ of a particular cardinality (k), it is assumed that cardinality $(k-1)$ is stable point of the $MACA$. The $MACA$ is evolved by the process of Genetic Algorithm.

The *Table -6.3* reports the experimental results for various values of n. *Column 2* shows the maximum number of random patterns(P) the set of $MACA$ has been able to memorize. *Column 3* shows the average number of generations required to memorize. The memorizing capacity is seen to be almost same as the theoretical limit. The slight difference occurs

because the individual $MACA$ chosen to classify the random patterns has slight different probability of non-occurrence(least q - *relation 6.6*) than the theoretical measure which is enumerated through probabilistic approach.

Table 6.2: Collision mean and Variance of sample pair with a fixed hamming distance

No of cell (n)	No of Random Patterns	No of Generation
10	8	5
20	13	11
30	17	13
40	22	17
50	24	29
60	27	30
70	27	31
80	31	39
90	33	43
100	37	47

6.4.2 Basins of Attraction

To evaluate the noise recovery capacity of the $MACA$, extensive experiments are performed. The experiments are performed with both single $MACA$ as well as Multiple $MACA$. For each case, experiment is done on 30 different set of classified patterns. The results based on single $MACA$ is reported next. The experimental setup is as follows.

Experimental Setup

For each set of patterns $\mathcal{P} = \{\mathcal{P}_1,\ \mathcal{P}_2 \cdot \mathcal{P}_k\}$, 300 random pattern $\mathcal{P}_{incom}$ are picked, each pattern $\mathcal{P}_{incom}$ having a fixed hamming distance(d) from any of the randomly chosen patterns $\mathcal{P}_i$. The value of d is increased from 1 to n. The error recovery at each d is the percentage of patterns which converge to the pivot pattern($\mathcal{P}_i$) from which it is generated.

Single MACA - Noise Recovery Capacity

For single $MACA$, the recovery is calculated on different cardinality (k) of the pattern set. The following results of pattern recognition capability are based on single $MACA$.

Figure 6.7 & 6.8 shows the experimental results illustrating the noise recovery capacity of $MACA$. *Fig. 6.7* shows the results for different value of k(= 7, 15, 17, 21) for a fixed value of n(=40). *Fig. 6.8* shows the result for different value of n(= 10, 30, 40, 60) at the stable point of each n. The results are the average of 30 different set.

The nature of the graph is similar to the one predicted through theoretical analysis. Only in *Fig. 6.7*, we see that there is difference in the nature of the noise accommodating capacity of the $MACA$ for two k =(17, 22). In the theoretical analysis, we have predicted for all values of k which need same number of attractors to classify, for example (both 17 & 22 need 32 attractors), they have the same basin of attraction. However, in experiment we see that in fact the basin of attraction for $k = 17$ is better than $k = 22$. This is because, the degree of freedom of classifying $k = 17$ is more (we have 15 attractor unused). This acts

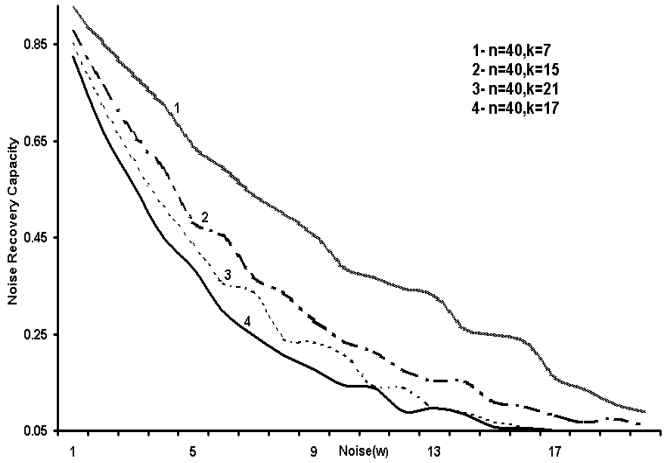


Figure 6.7: Experimental Results of Noise Recovery capacity of MACA($n = 40$) at various k

as an advantage and we subsequently end up in selecting $MACA$ with better recognizable capacity. However, the most important observation in the experimental results is that there is a significant gap in the theoretical and experimental observation regarding the error correcting capacity of the $MACA$.

Figure 6.9 shows the result of the abnormal difference in the theoretical limit as well as the experimental observation($n = 40$, $k = 22$). We explain this abnormality by taking a look at the $MACA$ basin and the methodology of $MACA$ space analysis undertaken in *Section V*.

Reason Behind the Gap in Theoretical & Experimental Result :
The entire theoretical analysis in *Section V* regarding Basin of Attraction is developed taking into consideration the statistical property of the $MACA$ space. The $MACA(n, m)$ space will form a $(n+2)$ - variable frequency distribution where each of the independent variables indicates $|\hat{P}(w, n, m)|$ $\forall_{w=0}^{n}$, while the dependent variable is the number of $MACA(n, m)$ - $\hat{N}(n, m)$, which displays the property of the corresponding $(n + 1)$ independent variables. Defining in functional form

$$\hat{N}(n, m) = f(|\hat{P}(0, n, m)|, |\hat{P}(1, n, m)|, \cdots |\hat{P}(n, n, m)|) \tag{6.9}$$

The *Fig. 6.10* displays a simplified 3-dimensional representation of the $MACA$ space, the x and y axis representing the $|\hat{P}(w, n, m)|$ for $w = 1$ & 2, the assumption being that there are only 3 variables - the independent variables $|\hat{P}(w, n, m)|$ for $w = 1$, 2 and $\hat{N}(n, m)$ - the dependent variable represented in the z-axis.

While exploring the $MACA$ space, the statistic upon which the analysis of error correcting capacity is done is mean. That is we have shown our result in respect to the behavior of an average $MACA$.

However, with only a small percent of $MACA$, classification of all the distinct config-

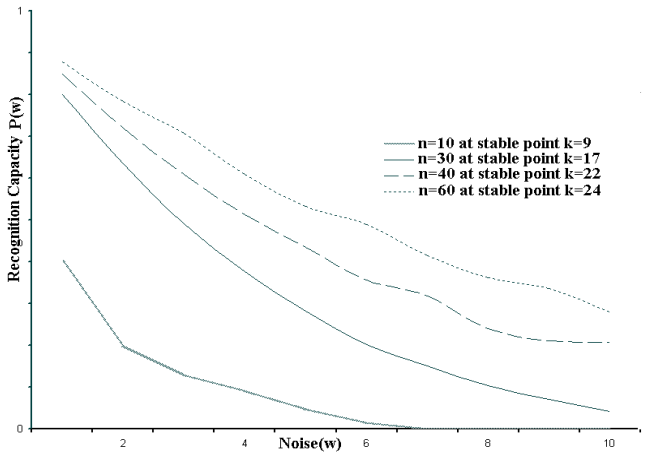


Figure 6.8: Experimental Results of Noise Recovery Capacity Of MACA at stable point for $n = 10, 30, 40, 60$

uration of the pattern set $\mathcal{P}$ having cardinality k can be done. We explain the statement through the following discussion.

The number of all possible pattern set of cardinality k chosen from all possible 2^n patterns is

$$[\mathcal{K}] =^{2^n} C_k \tag{6.10}$$

Let the cardinality k of the pattern set $\mathcal{P}$ follow the relation where $2^{m-1} < k \leq 2^m$. An n-celled $MACA(n, m)$ can classify $[\hat{\mathcal{K}}]$ number of k set of patterns that is, in the $MACA$ each member of the set $\mathcal{P}$ falls in different attractors - $[\mathcal{K}]$ is given by

$$[\hat{\mathcal{K}}] =^{2^m} C_{\mathcal{K}}(2^{n-m})^{\mathcal{K}} \tag{6.11}$$

The relation is obtained by selecting an element from each of the selected $\mathcal{K}$-basin.

Therefore, the number of $MACA$ required to classify the whole set of combination is given by

$$\tilde{N} = \frac{[\mathcal{K}]}{[\hat{\mathcal{K}}]} \tag{6.12}$$

The number of $MACA$ forms a small fraction of the of the entire space of the $MACA$, the fraction is given by

$$fracMACA(n, m) = \frac{\tilde{N}}{\hat{N}(n, m)} \tag{6.13}$$

where $\hat{N}(n, m)$ is the total number of $MACA(n, m)$ which can be formed.

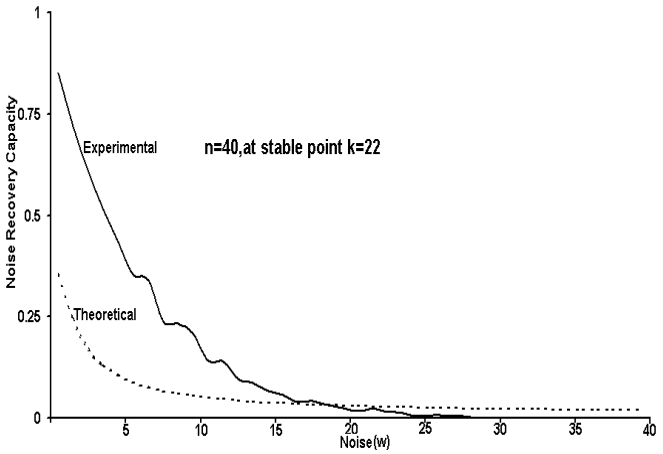


Figure 6.9: Noise Distribution $n = 40$ (Experimental & Theoretical)

We show the fraction of $MACA$ - $fracMACA(n, m)$ - required to classify the total set of patterns for different values of n at the stable point in *Table 6.3*. For example, in the case of $n = 40$, we see that at stable point, the fraction is around 10^{-10}, which implies that to classify each patterns, such a meager percentage of the whole population (10^{-8} %) is sufficient to classify the whole range of pattern.

So when during evolution we employ a fitness preference to those $MACA$ whose pattern recognition capacity is more, we reach the zone of $MACA$ which has the best pattern recognition capacity. Since it is a meager percentage of the entire population, selected $MACA$ displays above average noise recovery capacity.

We present a mathematical formulation of the above statement for one bit noise that is $w = 1$. In order to present the mathematical model, we assume that the frequency distribution is Gaussian in nature and simplify the model to a 2-dimensional normal distribution (*Fig. 6.11*). Through the model, we illustrate how the enhancement of pattern recognition occurs.

The x dimension of the distribution is taken as $|P(w, n, m)|$ for $w = 1$ and the y axis represent the number of $MACA$ - $\hat{N}(n, m)$ - corresponding to the a particular $|P|$.

Table 6.3: Average Stability

No of cell (n)	No of Random Patterns	Frac	z(Frac)
10	9	2.4×10^{-3}	2.8
20	13	5.4×10^{-5}	3.9
30	17	2.3×10^{-9}	5.9
40	22	3.8×10^{-10}	6.2
50	25	4.7×10^{-11}	6.5
60	28	2.4×10^{-11}	6.6
70	30	8.3×10^{-12}	6.8
80	32	1.8×10^{-11}	6.6

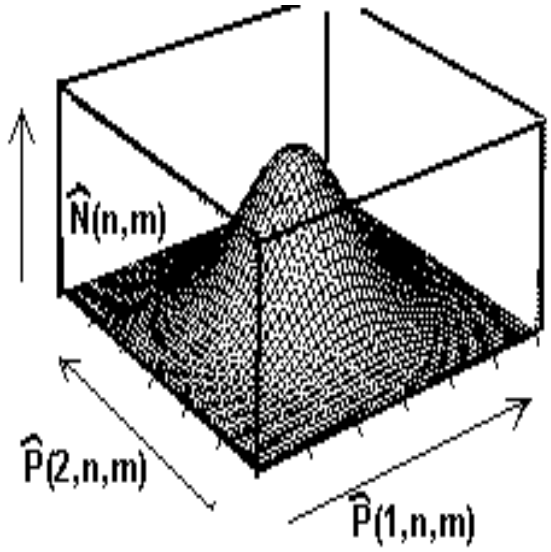


Figure 6.10: Three Dimensional Frequency Distribution

We calculate the mean (μ) and standard deviation (σ) of the frequency distribution formed from *relation* **??**. The relation is reproduced for convenience

$$|\hat{P}(w,n,m)| = \frac{\sum_{Perm(n)} f(w,i) \cdot \hat{N}(n(i),m)}{\hat{N}(n,m)}$$

The relation can be rewritten as

$$|\hat{P}(w,n,m)| = \sum_{Perm(n)} X \cdot p(x) \tag{6.14}$$

where

$$X = f(w,i) \quad \& \quad p(x) = \frac{\hat{N}(n(i),m)}{\hat{N}(n,m)}$$

Assuming the distribution as a standard normal distribution, we find the value($\mathcal{Y}$) of $|\hat{P}(1,n,m)|$ so that the $MACA$ having number of 1 bit patterns greater than $\mathcal{Y}$ covers the (3.8×10^{-8} %)of the entire space. (The space is shaded in *Fig. 6.11*). From the formula of standard normal distribution, we find the value of $\mathcal{Y}$ where

$$\frac{\mathcal{Y} - \mu}{\sigma} = z(fracMACA) \tag{6.15}$$

where μ and σ implies the mean and standard deviation of the distribution respectively and

z is the function defining the value corresponding to the $fracMACA$ area. The value of the function is obtained from the standard normal distribution table. μ is average while σ is the standard deviation of the $MACA$ distribution.

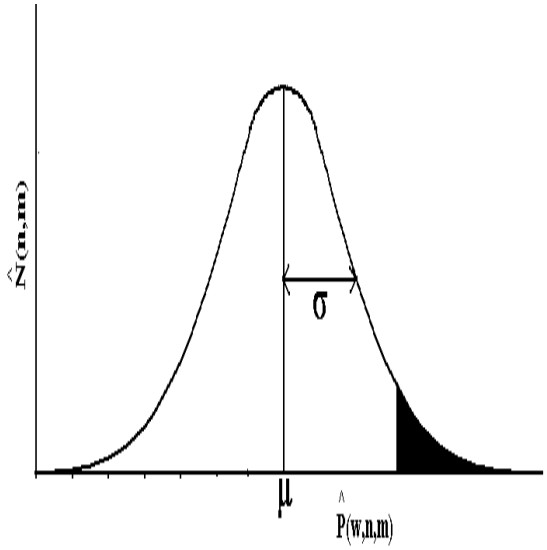


Figure 6.11: Gaussian(Normal) Probability Distribution Function

From *relation 6.15*, it is quite clear that the value of $\mathcal{Y}$ is much more than the mean μ. We illustrate the calculation for $n = 40$ at stable point 22. The mean μ and standard deviation (σ) are 14.20 & 3.25 respectively and $fracMACA = 3.8 \times 10^{-10}$. Therefore, the value of $\mathcal{Y}$ will be

$$\frac{\mathcal{Y} - \mu}{\sigma} = z(3.8 \times 10^{-10}) = 6.2 \tag{6.16}$$

Consequently, the value of $\mathcal{Y}$ is 34. Hence $EO(P(1, n, m)) = \frac{34}{\frac{{}^{n}C_1}{2^m}}$ which is roughly equal to the experimental observation.

Multiple MACA - Noise Recovery Capacity

The experimental results of noise recovery capacity of systems developed through the use of Multiple $MACA$ is illustrated through analysis of *Fig. 6.12*. The *Fig. 6.12* shows a typical representative experimental result. The pattern width ($n = 40$) and the system is trained with $k = 22$ patterns.

The line diagram 1 shows the hamming distribution of the pattern set $\mathcal{P}$. The distribution is enumerated by taking 300 random patterns ($\mathcal{P}_{incom}$) of hamming distance d for each ($d = 1, 2, \cdots 40$) from any of the random patterns $\mathcal{P}_i$, $\in \mathcal{P}$ and seen whether $HD(\mathcal{P}_{incom}, \mathcal{P}_i) < HD(\mathcal{P}_{incom}, \mathcal{P}_j)$ $\forall_{j=1}^{k}$ & $j \neq i$. The graph shows the percentage of case for which it follows the relation. It is seen that for hamming distance = 6, it always follow the relation beyond which it begins to slowly degenerate. At Hamming Distance 18, none of the pattern $\mathcal{P}_{incom}$

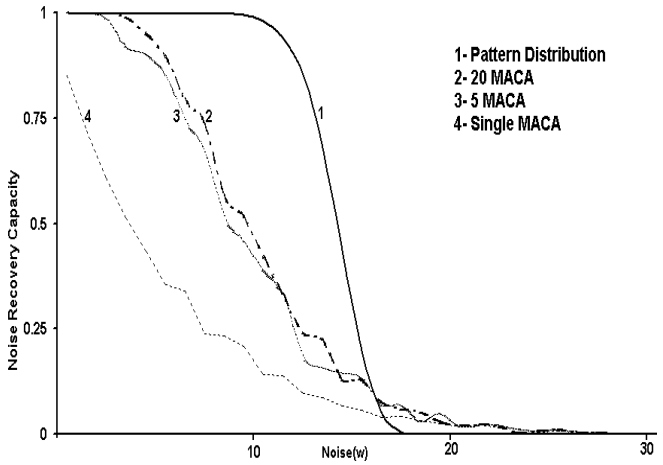


Figure 6.12: Experimental Results of Noise Accommodating Capacity of MACA($n = 40$) for Multiple $MACA$

is close to the pivot point $\mathcal{P}_i$ from which it is generated.

Through evolution we have obtained 20 different $MACA$ each classifying the input pattern set $\mathcal{P}$. The line diagram 4 shows the noise recovery capability of a single $MACA$. The line diagram 3 shows the error recovery capacity of the system developed with 6 $MACA$. The value 6 is taken because in *Fig. 6.5* we have shown theoretically that 6 $MACA$ is needed to recover the 1-bit noise. However, in this case we see that we can fully recover 3-bits of noise as well as we can recover most of the noise upto 8 bits. Moreover, the error recovery capacity doesn't degenerate as sharply as has been theoretically predicted. However, the rate of degeneration is much sharper compared to the behavior exhibited by single attractor. Around bit 20, the recovery capacity is less than that of single $MACA$. This also shows that the multiple $MACA$ system exhibit much more accuracy in recovering noise because according to the pattern distribution curve if an α-basin of an $MACA$ attracts patterns of more than 18 bit, it is attracting a pattern from a different class.

However, if we think in terms of the error correcting capacity of individual $MACA$ which is much better than theoretical limit, then the capacity enhancement through multiple $MACA$ is not as good as that predicted by the theoretical limit. This is because the theoretical formulation has made the assumption that all patterns of a certain weight occurs uniformly whereas in actual experimentation there are more repetition of some patterns than others. The sharpness in fall has mellowed down due to the same reason.

The non-uniformity of pattern distribution of a particular weight gets more reflected as we increase the number of $MACA$ to design the system. The rate of improvement in error recovery capacity also slows down as we introduce more and more $MACA$ to the system.

6.5 $MACA$ - As a Classifier

An n-bit $MACA$ with k-attractors can be viewed as a natural classifier. It classifies a given set of patterns into k-distinct classes, each class containing the set of states in the attractor basin.

To enhance the classification accuracy of the machine, most of the works [8, 48, 43] have employed $MACA$ to classify patterns into two classes (say I & II). Multi-Class Classifier is built by recursively employing the concept of two class classifier. We have also employed the same strategy. **MACA based two class classifier:** The design of the $MACA$ based Classifier for two pattern sets P_1 and P_2 should ensure that elements of one class (say P_1) are covered by a set of attractor basins that do not include any member from the class P_2. Any two patterns $a \in P_1$ and $b \in P_2$ should fall in different attractor basins. According to *Theorem* **??**, the pattern derived out of *modulo-2* sum of a and b ($a \oplus b$) should lie in a non-zero attractor basin. Let X be a set formed from *modulo-2* sum of each member of P_1 with each member of P_2 that is, $X = \{x_l \mid x_l = (a_i \in P_1) \oplus (b_j \in P_2)\ \forall_{i,j}\}$. Therefore, all members of X should fall in non-zero basin. This implies that the following set of equations should be satisfied.

$$T^d \cdot X \neq 0 \tag{6.17}$$

where T is a valid $MACA$ to be employed for designing two class classifier and d (Definition 5.1) is the depth of the $MACA$. But none of the earlier works have progressed beyond this point, that is none have proposed any efficient algorithm on how to find the T Matrix satisfying *equation 6.17*.

The strategy is illustrated with an example.

Example 6.1 *Let us have two 5 bit pattern sets P_1 and P_2, where P_1 = {P_{11} = 11111, P_{12} = 01111,P_{13} = 11010} and P_2 = {P_{21} = 01000, P_{22} = 01010} respectively. In order to classify these two pattern sets into two distinct classes - Class I and II respectively, we have to design an $MACA$ such that the patterns of each class falls in distinct attractor basins of an $MACA$.*

The $MACA$ of Fig. 5.2(a) *is able to classify them into distinct attractor basins where Class I (C_1) is represented by one set of basins with attractors as {00001, 10001 & 10000}, while Class II (C_2) represented by the 4^{th} basins with {00000} as the attractor.*

When a CA is loaded with an input pattern (say x = 00010) and is allowed to run in autonomous mode for a number of cycles equal to the depth of CA, it travels through a number of states and ultimately reaches an attractor state. The attractor to which the input pattern should reach depends on the average Hamming Distance (HD) of the pattern from the elements of two classes as illustrated next. *The theoretical foundation of this observation is noted in* Section IV.

The Hamming Distance (HD) between x and Class I = ($HD(x, C_1)$) is measured by averaging the HD with each member of the class. For example, ($HD(x, P_{11})$ = 4 , $HD(x,$

P_{12}) = 3, HD(x, P_{13}) = 2). Hence average (HD(x, C_1)) = $\frac{4+3+2}{3}$ = 3. Similarly, (HD(x, C_2)) = 1.5.

Therefore, (HD(x, C_2)) < (HD(x, C_1)). Hence, the input pattern x = (00010) *settles to attractor* 00000 *- the attractor representing Class II.*

For an ideal classifier, to distinguish between two classes, we would need one bit. The classifier of *Fig 5.2(b)* requires four attractors hence two bits to distinguish between two classes which is a memory overhead of the design.

[Note: The two class classifier identifies the class of an element in a constant time (d) - d is the depth of $MACA$. If we didn't have a classifier and yet wanted a constant time identification, we would have needed a hash table of size 2^n, n being the width of the pattern. This is impractical even for a moderate value of n. Further, there would not be any proper formulation of the method of prediction of noisy patterns.

Design of Multi-class Classifier: A two class classifier can be viewed as a single stage classifier. For designing a multi-class classifier, this scheme of single stage classification will be repetitively employed leading to a multi-stage classifier consisting of multiple CA, each CA corresponds to a single stage (*Fig.6.13*).

Hence, in the rest of the paper, we concentrate on designing an efficient two class classifier. The exponential complexity of the design satisfying *relation 6.17* has provided the motivation to fall back on the evolutionary program of Genetic Algorithm.

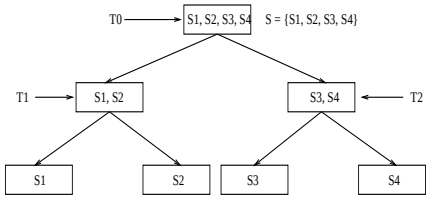


Figure 6.13: $MACA$ based multi-class classifier
Note : A leaf node represents a class in input set $S = \{S_1, S_2, S_3, S_4\}$

For an ideal classifier, to distinguish between two classes, we would need one bit. The classifier of *Fig* **??** requires two bits to distinguish between two classes which is a memory overhead of the design. A k-attractor two class classifier needs $log_2(k)$ bits.]

The above discussions set our design perspective. *In order to reduce the memory overhead, the design will strive to find an $MACA$ that would have least number of attractor basins.* That is, to classify pattern set into two classes, we should ideally find an $MACA$ with two attractor basins - each basin having the members of one specific class. Even if this ideal situation is not attained, the algorithm should design an $MACA$ based classifier having minimum number of attractor basins - while one subset of basins accommodates the elements of one class, the remaining subset houses the elements of the other class.

6.6 Evolving MACA

6.6.1 Fitness function

The fitness $\mathcal{F}(MACA)$ of a particular $MACA$ in a population is determined by the weighted mean of two factors - F_1 and F_2. While F_1 reflects satisfiability of *relation 6.17* in *Section 7.2* by the currently evolved $MACA$ (T Matrix), the number of attractors of the $MACA$ gets reflected in the factor F_2.

Determination of F_1 *:* X, as noted in *relation 6.17* of *Section 7.2*, is the set formed from *modulo-2* summation of each pattern of P_1 with each pattern of P_2. With 30 randomly chosen patterns from X, we test fraction of X satisfying the $T^d X \neq 0$ *Relation 6.17 in Section* . The fitness criteria F_1 of the $MACA$ is determined by the percentage of patterns satisfying the *Relation 6.17*. For example, if 15 out of 30 chosen patterns from X satisfies *relation 6.17*, $F_1 = 0.5$.

Determination of F_2 *:* The factor F_2 varies inversely with the number of attractors required to classify the pattern set. It has been defined as -

$$F_2 = 1 - [(m-1)/n]^l \tag{6.18}$$

where 2^m denotes the number of attractor basins for the n cell CA, and l is set to 1·8.

Subsequent to exhaustive experimentation, we have set the relative weightage of F_1 and F_2 as follows.

$$\mathcal{F}(MACA) = .8 \cdot F_1 + .2 \cdot F_2 \tag{6.19}$$

The experimental results reported in *Section 6.7* confirms that the *relation 6.18 and 7.6*, although evolved empirically, provides the direction for fast GA evolution to arrive at the desired $MACA$.

The capability of the evolved $MACA$ to classify a pattern with noise is next analyzed along with the characterization of the Attractor Basins of the evolved $MACA$.

6.7 Performance Analysis of $MACA$ based Classifier

The characterization of $MACA$ basins reported in the previous section has established the fact that a basin has patterns which are close to each other in terms of hamming distance. Hence patterns satisfying this criterion have high probability of getting classified into minimum number of attractors. Patterns not covered by the training sets, but satisfying the criteria, will also have high probability of being predicted as the right class. Moreover, the efficiency of classifying two classes of patterns in distinct attractor basins increases if the average distance between them is high. The classifier is assumed to be designed to classify n-bit patterns into k classes.

In the above context, to characterize the data sets to be classified in two classes, we introduce two terms D_{min} and d_{max}. D_{min} is the minimum hamming distance (inter-cluster distance) between two patterns $P_i \in C_1$ and $P_j \in C_2$ in two classes C_1 & C_2 respectively;

while d_{max} is the maximum distance (intra-cluster distance) between two patterns within a class.

6.7.1 Distribution of Patterns

For the sake of convenience of performance analysis, the distributions of patterns in two classes are assumed as shown in *Fig 6.14*. Each pair of sets on whom classifiers are run are characterized by the curves ($a-a'$, $b-b'$, $c-c'$, $d-d'$). The ordinate of the curves represents the number of pairs of patterns having the specified hamming distance. For example, the point A (on the curve for a) has y number of pairs of patterns which are at hamming distance x. The abscissa has been plotted in both direction, from left to right for *Class I* while from right to left for *Class II*. The curves of *Class I* & *II* overlap if $D_{min} < d_{max}$. An ideal distribution $a - a'$ is represented by the continuous line without any overlap of two classes.

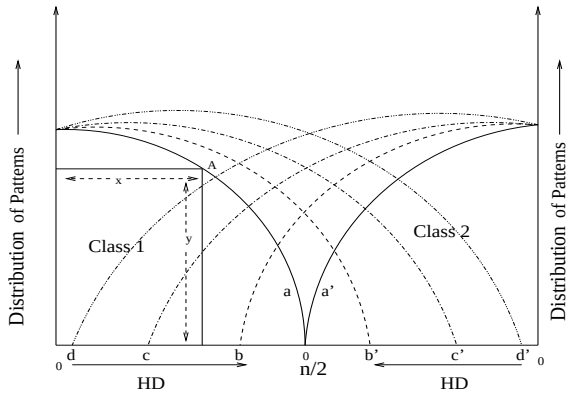


Figure 6.14: Distribution of patterns in class 1 and class 2
(HD denotes Hamming Distance)

6.7.2 Experimental Setup

In each of the above distribution various values of n are taken. For each value of n, 2000 patterns are taken for each class. Out of this set, 1000 patterns are taken from each class to build up the classification model. The rest 1000 patterns are used to test the prediction accuracy of the model. For each value of n, 10 different pairs of pattern sets are built. The *Genetic Algorithm* (GA) has been evolved for 100 generations.

6.7.3 Experimental Results

The *Tables 6.4- 6.6* represent the classification efficiency for the data set $a - a'$, $b - b'$, $c - c'$. *Column II* represents the different values of m for which GA finds the best possible solution; 2^m is the number of attractor basins of the $MACA$ designed as the classifier for the given pattern set. *Column III* and *IV* represent the classification efficiency of training and test data set respectively. Classification efficiency of training set is the percentage of patterns which can be classified in different attractors while that of test data implies the percentage of data which can be correctly predicted. The classifier predicts the class of a pattern in constant time. The best result of classification efficiency corresponding to each

m in the final generation is taken. This is averaged over for the 10 different pairs of pattern set taken for each value of n. The average is shown in *Column II & III.*

6.7.4 Analysis of Classifier Performance

The experiment, noted below, validate the theoretical foundations laid in the earlier sections.

Expt 1: Study of GA Evolution - The GA starts with various values of m. The GA parameters, as reported in *Section 8.4.3 and 7.4.2*, are so formulated that it soon begins to get concentrated in certain zone of values. The genetic algorithm is allowed to evolve for 50 generations. In each case 80% of the population in the final solution assumes the two or three values of m noted in *Column II.*

This shows that our scheme of GA is able to hit the correct zone of classification, the values of m where classification efficiency and design overhead is optimum is reported. It has been tested that even if we increase the value of m than the ones reported, the classification performance improves at a very slow gradient.

Table 6.4: Experiment to find out the value of m (Ideal Distribution($a - a'$))

Size (n)	Value of m	Performance (%)	
		Training	Testing
20	2	85.40	85.60
	3	96.10	94.35
40	3	98.20	97.75
	4	98.35	98.85
60	3	98.55	97.75
	4	98.50	98.00
80	3	98.81	98.65
	4	99.15	99.20
	5	99.75	99.70
100	3	99.65	99.25
	4	99.67	99.35

Expt 2. Experiment done on ideal set $a - a'$ and $b - b'$ with low overlap - It shows that classification accuracy is above 98% in most of the cases of the ideal situation (curve a - a'). Moreover, the prediction accuracy is also almost equal to the classification accuracy. In fact, it is better in some of the cases. The entire experiment is repeated with less than ideal sets characterized by the curve $b - b'$. There is some overlapping area between the data set of two classes. *Table 6.5* represents results on overlapping data sets. It is seen that even with the relaxation of conditions, the classification accuracy doesn't deteriorate much.

This validates the theoretical foundation that $MACA$ basins form natural clusters.

Expt 3: Experiment on data set $c - c'$ - With increasing overlap of data set (curve $c - c'$), classification efficiency of test data set falls below that of training data set. *Table 6.6* shows the result. But even in this case the classification efficiency and the number of attractors required to represent the data set doesn't deteriorate appreciably.

In the theoretical analysis, it has been shown that the expected behavior of $MACA$

Table 6.5: Experiment to find out the value of m (Curve $b - b'$)

Size (n)	Value of m	Performance (%)	
		Training	Testing
20	2	83.20	82.00
	3	92.20	93.35
40	3	97.60	96.80
	4	97.20	97.45
60	3	96.90	96.05
	4	96.90	96.05
80	3	98.70	97.70
	4	98.70	97.70
	5	98.30	97.30
100	3	98.30	97.45
	4	98.40	97.30
	5	97.65	97.55

Table 6.6: Experiment to find out the value of m (Curve $c - c'$)

Size (n)	Value of m	Performance (%)	
		Training	Testing
20	2	81.20	72.40
	4	92.20	83.35
40	2	77.60	66.80
	4	94.28	87.45
60	3	86.98	77.55
	4	91.90	86.60
80	3	81.70	76.35
	4	88.79	87.30
	5	91.65	87.10
100	3	86.40	77.95
	4	83.10	80.35
	5	93.40	91.50

is that its basin forms natural clusters. But since it is a statistical behavior, there are some (small amount) of $MACA$ whose basins doesn't conform with the condition. In view of availability of such $MACAs$, the results for data set $c - c'$ do not deteriorate to an appreciable extent. Naturally, the prediction capacity deteriorates as there is no such metric to bind each individual class of the data set.

Expt 4a: Experiment with data sets having implicit clusters - Even if the data set between two classes may have a fair amount of overlap ($d - d'$), the classifier functions well if the classes have implicit clusters among them. The following experiment is performed to validate this observation.

To perform this experiment, we first randomly generate four pivot patterns which are separated by a fixed hamming distance (say D_{min}). Then around each pivot, we randomly generate a cluster of p number of patterns within d_{max} distance (maximum hamming distance between a pivot and an element in the associated cluster); where $d_{max} <= D_{min}/2$.

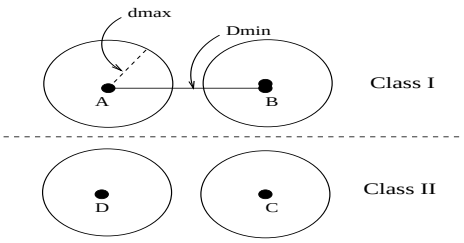


Figure 6.15: Clusters Detection by two-class classifier

Table 6.7: Clusters Detection by MACA based Classifier,
$n = 100$, $p = 2000$, $D_{min} = 20$, $d_{max} = 5$

Combi^n of Clusters	Value of m	Performance (%)	
		Training	Testing
A & *B*, *C* & *D*	2	95.90	92.30
	4	99.82	97.10
A & *C*, *B* & *D*	2	94.50	92.30
	4	98.70	96.62
A & *D*, *B* & *C*	2	94.60	90.40
	4	99.20	96.82

The *Fig 6.15* represents four pivots **A, B, C, D** which are separated by hamming distance D_{min} . Around each pivot, we generate some patterns within d_{max} distance. Four clusters are generated in the process. Then combining two clusters each, we generate two classes. For example, the data set of clusters **A** and **B** is represented by $Class1$; whereas $Class2$ represents the data set of clusters **C** and **D**. When two clusters together form an individual class, the elements of the new class do not maintain (D_{min}, d_{max}) relationship - a variant of the form denoted by the curve ($d - d'$) in *Fig 6.14*.

It is seen that the $MACA$ based classifier perform classification task very well even though the distribution of patterns of each class is not ideal. The *Table 6.7* reports the detail result of cluster detection. The *Column III* and *IV* depict the classification efficiency of the training and testing data set respectively at different values of m.

Expt 4b: The last experiment is subjected to further study. The contents of each attractor basins are probed. For example, the first row in *Table 6.7* has employed 2^2 attractors to classify the patterns. *Clusters A & B* form a class (Class I) and occupies two attractors while *Cluster C & D* occupies the other two attractors. In the two attractors (say *a* & *b*) which *Cluster A & B* occupy, it is found that 99% of patterns of *Cluster A* occupies attractor *a*, while 99% of patterns of *Cluster B* occupies the attractor *b*. It is similar in case of the class formed from *C* and *D*.

This is in line with the theoretical formulation that each attractor basin generates a cluster of patterns having lesser hamming distance. Thus the $MACA$ in this case have been able to identify clusters within a class and have put them in separate basins.

The experimental observations illustrates beyond doubt that the classifier proposed is

very robust one and has the capacity to execute a wide variety of practical classification task.

6.8 Conclusion

The paper presents the detailed design and analysis of Cellular Automata (CA) based Pattern Classification technique. Theoretical formulation supported with experimental results have established the $MACA$ (*Multiple Attractor CA*) as an efficient machine to solve the problems of real life pattern classification.

Chapter 7

An Evolutionary Design of Pseudo-Random Test Pattern Generator Without Prohibited Pattern Set (PPS)

7.1 Introduction

The pseudo-random pattern generators ($PRPG$s) are widely used in $VLSI$ circuits as the efficient test pattern generators [**?**][**?**]. The $PRPG$ generates a large volume of patterns to test the different CUTs (*Circuit under Test*) of a $VLSI$ chip that may be accessed through a full or partial scan path. However, there are applications where some patterns are declared prohibited to a CUT. If the CUT is subjected to a pattern of the prohibited pattern set (PPS), the CUT may be placed to an undesirable state and even may get damaged. The manufacturers do face the problem while testing the chip equipped with $PRPG$.

The prohibited pattern set (PPS) can be of two different types.
Type I : The first one are random patterns regarding which the test designer has gathered information in the process of testing. These patterns are random in nature and do not display any co-relation whatsoever. The test engineer doesn't have any idea why those patterns are creating problem. Consequently, the cardinality of the prohibited pattern set is small in this case.
Type II : The second type of PPS are those formed out of some prohibited functions (PFs) on a subset of *primary input (PIs)*. Test designer identifies the functions from analysis of the circuit. Generally, a few number of PIs are involved in PF. However, the prohibited function (PF) produces PPS of cardinality much larger than that of *Type I*.

The design of a CA based test pattern generator (TPG) enriched to generate large sequence of patterns without any pattern from the given PPS has been first proposed in [**?**].

This paper refines the earlier work with specific emphasis on real life problems encountered by test engineers in a typical semi conductor industry. The design methodology proposed in the paper ensures PPS free TPG considering both the Types, I and II. The Type II PPS, due to some prohibited function, was not considered in [?]. Moreover, the solution in the current paper is based on a novel scheme which combines the guided search technique provided by Genetic Algorithm (GA) and the analytical framework of vector subspaces generated by a Cellular Automata (CA). The class of CA referred to as the group CA [?] are used for the design of TPG.

The proposed methodology can also be implemented for the $LFSR$ (linear feedback shift register) based TPG. However, the modular, local neighborhood cascadable structure of cellular automata (CA) suits ideally for $VLSI$ applications [?].

The next section briefly introduces the CA preliminaries. *Section III* presents the TPG design scheme followed by GA based evolution strategy of the proposed design in *Section IV*. The experimental results are subsequently reported in *Section V* which clearly establish the proposed design of TPG as the most desirable solution to address the real problem semiconductor industries encounter.

7.2 Cellular Automata Preliminaries

This section presents the preliminaries of Cellular Automata (CA) theory that are necessary to follow the proposed design of TPG without prohibited pattern set (PPS). The detailed theoretical foundation has been reported in the book [?]. A *Cellular Automaton (CA)* can be viewed as an autonomous finite state machine (FSM) consisting of a number of cells. The next state of a cell depends on the present states of its neighbors. In a 3-neighborhood dependency, the next state q_i^{t+1} of a cell is assumed to be dependent only on itself and on its two neighbors (left and right). It is denoted as

$$q_i^{t+1} = f(q_{i-1}^t, q_i^t, q_{i+1}^t),$$

where q_i^t represents the state of the i^{th} cell at t^{th} instant of time and *'f'* is the next state function, referred to as the rule of the automata. The decimal equivalent of the next state function, as introduced by Wolfram [?], is the rule number of the CA cell. For example,

Rule 90 : $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$

Rule 150 : $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$

where $\oplus$ (XOR function) denotes *modulo-2* addition. Since f is a function of 3 variables, there are 2^{2^3} i.e., 256 possible next state functions (rules) for a CA cell. Out of 256 rules there are 7 rules with XOR logic. The CA employing only the XOR rule is referred to as Linear CA [?]. In the present work we will concentrate on linear CA.

7.2.1 Characterization of Linear CA

A linear CA is represented by an $n \times n$ characteristic matrix T, where

$$T_{ij} = \begin{cases} 1, & \text{if the next state of the } i^{th} \text{ cell depends} \\ & \text{on the present state of the } j^{th} \text{ cell} \\ & i,\ j\ =\ 1,\ 2,\ ...,\ n \\ 0, & \text{otherwise.} \end{cases}$$

If we restrict to the 3-neighborhood dependency, then T[i,j] can have non zero values only for $j = (i-1),\ i,\ \ and\ \ (i+1)$. Thus, T becomes a tri-diagonal matrix. The state $S_t(x)$ of an n-cell CA at t^{th} instant of time is a string of n binary symbols. The next state (pattern) $S_{t+1}(x)$ of the CA is given by :

$$S_{t+1}(x) = T.S_t(x)$$

The CA can also be represented by a polynomial referred to as the characteristic polynomial of the CA. The following subsection provides some of the important issues related to the polynomial representation of CA which are relevant for the current design.

7.2.2 Polynomial Representation

The polynomial $p(x)$ of which the characteristic matrix T is a root is referred to as the *characteristic polynomial* of the CA. The characteristic polynomial $\phi(x)$ is derived from T by computing $\det(T + Ix)$. The $\phi(x)$ comprises of invariant polynomials $\phi_i(x)^{n_i}$, where $\phi_i(x)$ is irreducible. The polynomials $\phi_i(x)^{n_i}$, invariant to the linear operator T, are referred to as elementary divisors. The elementary divisors and characteristic polynomial maintain the following properties. These are relevant for the proposed TPG design.

Property 1: Different characteristic polynomials produce different vector subspace.

Property 2: Multiple Ts (characteristic matrices) can have the same characteristic polynomial [**?**].

Property 3 : Elementary divisors arranged in different orders produce different vector subspaces.

If all the states in the state transition diagram of a CA lie in some cycles, it is a group CA whereas a non-group CA state transition graph has both cyclic and non-cyclic states. In this paper we have employed group CA. The characteristic polynomial as well as the characteristic matrix of a CA are used to analyze the state transition behavior of the CA.

7.2.3 Properties of Group CA

All the states of a group CA lie on some cycles. For a group CA, the T matrix is nonsingular - that is, the $det[T] \neq 0$. The group CA can be classified as maximal-length and non-maximal length CA. The maximal length CA *(Fig. 7.1)* is the special class of group CA having a cycle of length $2^n - 1$ with all non-zero states, where n is the number of cells in the CA. Maximum length CA generates excellent pseudo-random sequence [**?**]. The characteristic polynomial of an n-cell maximal length CA is the n^{th}-degree primitive polynomial.

For a non-maximal length CA *(T_1 of Fig. 7.2)*, the characteristic polynomial gets factored to invariant polynomials (elementary divisors). The characteristic polynomial $\phi(x)$ of such a CA can be represented as

$$\phi(x) \ = \ \phi_1(x)^{n_1}\phi_2(x)^{n_2}\cdots\phi_K(x)^{n_K}.$$

Each of the elementary divisors $\phi_i(x)^{n_i}$ forms cyclic subspace - which leads to the generation of multiple cycles in a group CA. The entire state space V of a non-maximal length group CA is the direct sum

$$V = I_1 + I_2 + \cdots + I_K,$$

where I_i is the cyclic subspace generated by the divisor $\phi_i(x)^{n_i}$.

The proposed design of TPG employs non-maximal length group CA. An example non-maximal length group CA is shown in *Fig.7.2* with its T matrix (*Fig.7.2a*) and its four cyclic subspaces(*Fig.7.2b*). The subsequent discussions report the enumeration of cyclic sub-spaces of a CA and the synthesis of CA with desired specifications.

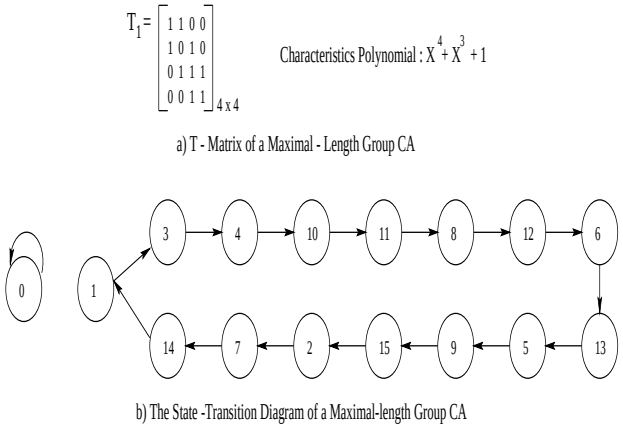


Figure 7.1: A 4-cell maximal length group CA

7.2.4 Enumerating Cycle Structure of a CA

The cyclic subspace divides the entire state space of the group CA into discrete cycles (cycles I,II,III, and IV of *Fig.7.2(b)*). The method of enumerating the cycle structure for linear CA are detailed through the following theorems. The proofs of the theorems are reported in [**?**].

Theorem 7.1 *A CA of size n with characteristic and minimal polynomial $\phi(x)$, where $\phi(x)$ is irreducible, has cycle structure [1(1), $\mu(k)$]; where $\mu \times k = 2^n - 1$. μ is referred to as cyclic component and k is the cycle length of each cyclic component.*

Theorem 7.2 *A CA with characteristic and minimal polynomial $\phi(x)^t$, where $\phi(x)$ is irreducible, has cycle structure $[1(1), \mu_1(k) \cdots \mu_m(2^m \cdot k)]$;*
where $2^{m-1} < t \leq 2^m$, and $\mu_i = 2^{r(t-1)}(2^r - 1)/2^i \cdot k$. k is the primary period, r is the degree of $\phi_i(x)$ and t refers to the power to which polynomial $\phi_i(x)$ is raised.

Theorem 7.3 *If a CA with characteristic polynomial $\phi_1(x)$ and $\phi_2(x)$ has the cycle structures A_1 and A_2 respectively, where A_1 = [1(1), $\hat{\mu}_1(\hat{k})$, $\hat{\mu}_2(2 \cdot \hat{k}), \cdots \hat{\mu}_i(2^i \cdot \hat{k}) \cdots \hat{\mu}_{\hat{m}}(2^{\hat{m}} \cdot \hat{k})$] and A_2 = [1(1), $\tilde{\mu}_1(\tilde{k})$, $\tilde{\mu}_2(2 \cdot \tilde{k}), \cdots \tilde{\mu}_i(2^i \cdot \tilde{k}) \cdots \tilde{\mu}_{\tilde{m}}(2^{\tilde{m}} \cdot \tilde{k})$], then*
*the cycle structure of the CA with characteristic polynomial $\phi(x) = \phi_1(x) \cdot \phi_2(x)$ is the product of each i^{th} term of A_1 with j^{th} term of A_2, i = 1,2,$\cdots, n_1$, j = 1,2,$\cdots, n_2$. That is, the product of $\hat{\mu}_i(\hat{k})$ and $\tilde{\mu}_j(\tilde{k})$ produces cyclic component μ of cycle length k [**?**], where*

$$\mu = \hat{\mu}_i.\tilde{\mu}_j.gcd(2^i\hat{k}, 2^j\tilde{k}) \text{ and}$$

$$k = lcm(2^i \hat{k}, 2^j \tilde{k})$$

7.2.5 Synthesis of Group CA

This subsection details the technique of synthesizing a group CA from a given characteristic polynomial $\phi(x) = \phi_1(x)^{n_1} \cdot \phi_2(x)^{n_2} \cdots \phi_K(x)^{n_K}$, where each $\phi_i(x)$ is an irreducible polynomial. The following three theorems provide the framework for synthesis scheme [**?**][**?**] reported in this section.

Theorem 7.4 *Corresponding to each irreducible polynomial $\phi_i(x)$ with degree p, a linear CA (tri-diagonal $[T_i]_{p \times p}$ matrix) can be synthesized.*

Theorem 7.5 *Given a tri-diagonal matrix T_i with the characteristic polynomial $\phi_i(x)$, the matrix*

$$T_i(n_i) = \begin{bmatrix} T_i & 1 & & \\ 0 & T_i & 1 & \\ . & . & . & \\ 0 & 0 & T_i & \end{bmatrix}_{n_i \times n_i}$$

has both characteristic and minimal polynomials as $f_i(x) = \phi_i(x)^{n_i}$.

Theorem 7.6 *If the matrices T_i, i=1,2,3,$\cdots$, K, with characteristic polynomial $f_i(x)$ are arranged in block diagonal form which results in*

$$T = \begin{bmatrix} T_1(n_1) & 0 & \\ 0 & T_2(n_2) & 0 \\ . & . & . \\ 0 & 0 & T_K(n_K) \end{bmatrix}$$

then $\phi(x) = f_1(x) \cdot f_2(x) \cdots f_K(x)$ is the characteristic polynomial of the resultant matrix T.

We next outline the synthesis algorithm for group CA.

Algorithm 7.1 Synthesis of Group CA
Input: *Characteristic Polynomial $\phi(x) = \phi_1(x)^{n_1} \cdot \phi_2(x)^{n_2} \cdots \phi_K(x)^{n_K}$.*
Output: *Tri-diagonal matrix (CA).*
*1. For each $\phi_i(x)$ generate T_i matrix (*Theorem 7.4*).*
*2. Generate $T_i(n_i)$ corresponding to $\phi_i(x)^{n_i}$ (*Theorem 7.5*).*
*3. Generate T matrix corresponding to $\phi(x)$ by combining the $T(n_i)$s (*Theorem 7.6*).*

With this theoretical background we now propose our TPG design scheme employing non-maximal length group CA.

7.3 TPG Design Scheme

The patterns generated from a maximal length CA display better "randomness" quality and hence used as Pseudo Random Pattern Generators ($PRPG$s) for testing of $VLSI$ circuits. However, the n-cell ($n \geq 16$) non-maximal length group CA having cycle length greater than 2^{n-1} doesn't suffer in respect of randomness qualities and also can be used as the TPG for a CUT. This observation has been validated through exhaustive experimental results reported in *Section V*.

In the proposed design, the state space of a CA considered for TPG is divided into multiple cycles. At least one of the cycles has large cycle length, fairly close to the maximal cycle length $(2^n - 1)$. The large cycle is designated to generate the pseudo-random test patterns free from the given PPS.

The proposed design, as noted in *Section 7.1*, takes into consideration of two types of PPS. *Type I PPS* with random vectors (patterns). For such a case, the cardinality of PPS is very small. *Type II PPS* displaying some prohibited function on a subset of $PI's$ has larger cardinality. The *Type I* is typically prevalent in circuits with small number of PIs, whereas the second case is applicable for a circuit having large number of PIs.

We introduce the following terminologies to designate the cycles of a non-maximal length group CA to be designed as a TPG.

Target Cycle (TC): The cycle of largest length generated by the CA and used for generating test patterns.

Redundant Cycle (RC): The cycles other than TC - these are redundant in the sense that these are not used for generation of pseudo random test patterns.

Design Objectives: The basic design objectives are:

- Cover the PPS as far as possible with RCs;
- Maximum length of TC should be free from PPS not covered by RCs; and
- The patterns generated by the TC without the PPS should display good pseudo-random qualities.

The following observations provide the guideline for the proposed design.

- If the number of primary inputs (PIs) n of a CUT is large, the number of test patterns needed to test the circuit is then comparatively much lesser than $2^n - 1$, the total number of patterns generated by the n-cell maximal length CA.
- In small circuits (less number of PIs), maximal space of the *Target Cycle (TC)* should be free from PPS so that the number of test patterns is close to the maximal length $(2^n - 1)$.
- For a circuit with large number PIs, a large space, but small compared to the maximal length, should be free from PPS, so that it can be used for testing the CUT.

Although a small percentage of maximal length is required to test the CUT, the TC should be of length close to maximal length, as far as possible, to provide good *pseudo-random* patterns.

Let us consider the 7-cell group CA of *Fig.7.2* with characteristic matrix $[T_1]_{7\times 7}$ *(Fig.7.2(a))*. Its characteristic polynomial $x^7+x^5+x^3+x^2+1$ can be factored as (x^4+x+1) and (x^3+x+1). So as per the theory noted in the *theorems 7.1* -7.3, the states generated by the CA are divided into three small cycles of length 1, 7 & 15 and only one large cycle of length 105 *(Fig.7.2(b))*. The CA (*Fig.7.2(a)*) can be selected as the desired TPG for a CUT with 7 PIs for some given set of prohibited patterns either supplied in the form of random PPS (Type I) or in the form of a prohibited function (PF) (Type II).

Type I : Let us assume that the PPS of the CUT contains 10 prohibited patterns as shown in *Fig.7.2(a)*. Out of the given PPS, the cycle II ($length = 7$) contains 3-prohibited patterns PPS_1={0110100, 1101101, 1011001}, whereas the prohibited patterns PPS_2 = {0000110, 0000010, 0001001, 0000111, 0001111} fall in cycle III ($length = 15$). The rest 2 prohibited patterns {0010001, 0100100} covered by the cycle IV are separated by a distance of 10 time steps - that is $T_1^{10}(0010001)^t = (0100100)^t$. (t denotes the transpose of the vector). To avoid these two prohibited patterns, the CA is loaded with 0100100 and can run for L=94 time steps to generate test pattern sequence starting from 1111110 - that is, the state following 0100100. In effect, the group CA (T_1) with 0100100 as the seed is a desired TPG, which avoids generation of the PPS while generating pseudo random patterns of length 94 which is close to maximal length - approximately 75% of maximal length 127.

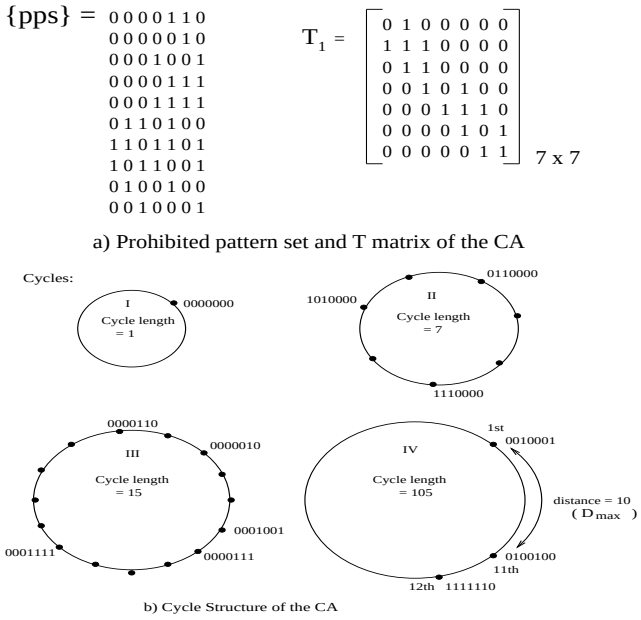


Figure 7.2: $PRPG$ without the Prohibited Patterns

Type 2: For this illustration let the prohibited pattern set is given in the form of patterns

satisfying a prohibited function (PF) on PIs marked as $a, b, c \cdots g$. For example, let out of the 7 PIs of the CUT three PIs - a,b, and c are associated with the given prohibited function defined as $a\overline{bc} + abc = 1$. So the number of prohibited patterns resulted from the PF is $2 \times 2^4 = 32$. Out of the Target Cycle (TC) we should generate a pattern sequence of sufficient length (k) without encountering the PPS. For circuits with large number of PIs, the number of test patterns k is small compared to the length of TC.

The task of designing the TPG for a CUT without the PPS is a computationally hard partition problem. For Type I PPS, we reported some heuristic in [**?**]. However, on detailed analysis we came to the conclusion that design of efficient heuristics to realize the design objectives (noted in the initial part of *Section III*) is extremely difficult. So, we employ genetic algorithm to evolve the desired CA satisfying the design objectives. The next section reports an efficient scheme for fast convergence of GA evolution leading to the desired TPG.

7.4 Evolution of Group CA Based TPG

The basic structure of GA encodes a solution in bit string format referred to as chromosome. It evolves the successive solutions according to its fitness. For the current application, the evolved solution is a *group* CA and the fitness of the solution depends on its capability to meet the design objectives specified as a set of constraints for GA evolution.

The following subsections report the technique to encode a solution (chromosome) and the formation of fitness function for the proposed evolution process. Subsequent subsections report the scheme for implementation of the three major functions of GA - random generation of Initial Population, Crossover, and Mutation, as developed in the current GA formulation.

7.4.1 Pseudo-Chromosome

The pseudo-chromosome is a string of n symbols representing the CA. It gives a semblance of the chromosome and hence termed as *pseudo-chromosome format*. The structure represents a *group* CA with respect to the arrangement of the cyclic subspace. In this format, the representation of each elementary divisor $\phi_i(x)^{n_i}$ of a CA with characteristic polynomial $\phi(x)$ is done by R_i number of cells. The value of R_i can be derived, following the *theorems 7.1* & 7.2, as

$$log_2(\mu_i \times k_i + 1) \times n_i = R_i \tag{7.1}$$

where μ_i is the cyclic component and k_i is the cycle length of $\phi_i(x)$. The first cell contains the elementary triplet $< \mu_i, k_i, n_i >$. For example, the $\phi_i(x)$=$x^4 + x^3 + x^2 + x + 1$ is encoded as $[< 3, 5, 1 >, *, *, *]$. The significance of * symbol is discussed in *Subsection C*.

The *pseudo-chromosome format* is illustrated in *Fig.7.3* for the *group* CA having its characteristic polynomial ϕ= $(x^4 + x^3 + x^2 + x + 1) \cdot (x^2 + x + 1)^2 \cdot (x^3 + x + 1)$. It has been explained in details in *Subsection C* of this section.

7.4.2 Fitness Function

The fitness function of the evolution scheme differs for the two types *(Type I and Type II)* of prohibited patterns defined in the earlier sections. The following discussions provide the framework of the formation of fitness function for each of the cases.

Fitness function for Type 1: The fitness $\mathcal{F}$ of a *group CA* in a population is determined by weighted mean of the following factors.

- F_1 : The amount of PPS free test patterns available.
- F_2 : The length of the Target Cycle (TC).

Calculation of F_1 : The ideal situation is the total space of TC is available for generation of test patterns (maximum possible value is $2^n - 1$ for an n-cell CA). The fitness deteriorates while the available space goes down. Let k_1 be the length of the space available. Then

$$F_1 = k_1/(2^n - 1) \tag{7.2}$$

Calculation of F_2 : As already mentioned, the maximum length CA is an excellent pseudo random generator. So the best situation would be if the TC is equal to the maximal length cycle. Let k_2 be the actual length of the Target Cycle (TC) of a population (CA), then

$$F_2 = k_2/(2^n - 1) \tag{7.3}$$

Subsequent to exhaustive experimentation, we have set the relative weightage of F_1 & F_2 and the fitness function is fixed as follows:

$$\mathcal{F} = 0.85 \cdot F_1 + 0.15 \cdot F_2 \tag{7.4}$$

Fitness function for Type 2: In case of prohibited function the objective is - rather than finding the maximum available free space, find a free space of size (say) k as the length of TC for such cases is very large. A typical value of k is assumed to be greater than 1,00,000. The fitness depends upon the two factors $\tilde{F}_1$: The amount of available space as a factor of k and $\tilde{F}_2$: the length of the target cycle (TC).

Calculation of $\tilde{F}_1$: To find the available space, we perform a non-deterministic test. We generate randomly 10 seeds (S_i) and run the CA for length p_i ($\leq k$) until and unless we encounter a prohibited pattern. The fitness function is

$$\tilde{F}_2 = \forall_i max(p_i/k) \tag{7.5}$$

Calculation of $\tilde{F}_2$: It is same as defined in *Relation 7.3*

The relative weightage of $\tilde{F}_1$, $\tilde{F}_2$ are set as follows:

$$\tilde{\mathcal{F}} = 0.85 \cdot \tilde{F}_1 + 0.15 \cdot \tilde{F}_2 \tag{7.6}$$

7.4.3 Generation of Initial Population

To form the initial population, it must be ensured that the each solution, randomly generated, is an n-cell *group CA*. The elementary divisor form of a *group CA* is $\phi_1(x)^{n_1} \cdot \phi_2(x)^{n_2} \cdots \phi_K(x)^{n_K}$ - that is, the *group CA* has K elementary divisors each of which forms cyclic group. The random generation of a *group CA*, therefore, implies the generation of K elementary divisors. To implement this, the following steps are executed.

- Randomly generate the value of K.
- Randomly generate the degree (R_i) of each elementary divisor, that is degree of $\phi_i(x)^{n_i}$. Therefore, $R_1 + R_2 + \cdots + R_K = n$.
- Randomly generate the degree (r_i) of each irreducible polynomial $\phi_i(x)$ following the relation $r_i \times n_i = R_i$, where n_i is the power to which it is raised.
- Randomly generate the cycle structure of each irreducible polynomial $\phi_i(x)$. The cycle structure according to *Theorem 7.1* will be [1(1), $\mu_i(k_i)$], where $\mu_i \times k_i = 2^{r_i} - 1$. Therefore, to determine the cycle structure, randomly factor $2^{r_i} - 1$ into two components μ_i and k_i respectively.
  At this point, corresponding to each elementary divisor we have a elementary triplet $< \mu_i, k_i, n_i >$ representing cyclic component, cycle length of $\phi_i(x)$ and the power to which $\phi_i(x)$ is raised.
- The polynomial $\phi_i(x)$ corresponding the cycle structure $[1(1), \mu_i(k_i)]$ is enumerated as noted in [**?**].
- The characteristic polynomial $\phi(x)$ is determined once all the elementary divisors are enumerated.
- The CA corresponding to $\phi(x)$ is synthesized using *Algorithm 7.1.*
- The cycle structure of the CA is then enumerated (*Theorem 7.1-7.3*) and the length of the Target Cycle (TC) is determined.

Fig.7.3 shows a CA that is generated randomly. For this example,

- The value of $n = 11$ and K=3.
- The degree (R_i) of elementary divisors are 4, 4, and 3 respectively.
- The degree (r_i)s of the irreducible polynomials are 4, 2 and 3 while the powers(n_is) to which they raised are 1, 2, and 1 respectively.
- The cycle structures of $\phi_i(x)$s are [1(1),3(5)], [1(1),1(3)] and [1(1),1(7)].
- The polynomials $\phi_i(x)^{n_i}$s are $(x^4 + x^3 + x^2 + x + 1)$, $(x^2 + x + 1)^2$ and $(x^3 + x + 1)$ respectively.

- The CA synthesized from them is represented by the T matrix of *Fig.7.3*.
- The cycle structure of the CA (T) is shown in *Fig.7.3*, while the length of the Target Cycle is 210.

A departure from the conventional form of chromosome representation demands associated modifications, as introduced below, for the cross-over process of GA.

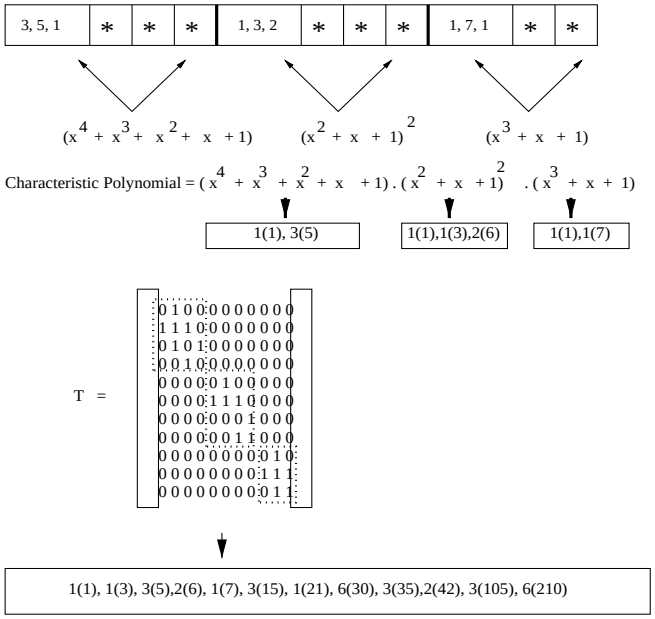


Figure 7.3: Group CA in pseudo-chromosome form

7.4.4 Crossover Algorithm

The crossover algorithm implemented for the present application is similar in nature to the conventional one with minor modifications as illustrated below. The algorithm takes two *group CA* from the present population (PP) to form the resultant *group CA* for the next generation. We implement single point crossover.

Fig.7.4 shows an example of the crossover process. Two *group CA* CA_1, & CA_2 in *pseudo-chromosome format* are shown in *Fig.7.4a* & *Fig.7.4b*. The first 6 symbols are taken from CA_1, while the rest 5 symbols are taken from CA_2 to form the CA for the next generation. At the cross over point 6, the resulting CA (CA') violates (explained below) the *pseudo-chromosome format* (*positions 6 - 8* encircled in *Fig.7.4c*).

The *pseudo-chromosome* of $\phi(x)$ is represented by the triplet $< \mu_i, k_i, n_i >$ followed by $R_i - 1$ * symbols, where $R_i = log_2(\mu_i \times k_i + 1) \times r_i$. In the representation of CA' (*Fig.7.4c*), we have the triplet $< 3, 5, 1 >$ followed by two *s - that is, R_i=3. This violates the basic format of the pseudo-chromosome, since $log_2(3 \times 5 + 1) = 4$. Hence, the property of *group CA* is not maintained for CA'. We compensate the violation by randomly generating an elementary divisor $\phi_2(x)$ of degree 3. For the present example, the cycle structure of $\phi_2(x)$ is [1(1),1(7)]

and the power (n_2) of $\phi_2(x)$ is 1. Therefore, the new elementary triplet is $< 1, 7, 1 >$. The resultant *group CA* for the next generation is shown in *Fig. 7.4d*. The formal algorithm for cross over operation is next noted.

Algorithm 7.2 Algorithm Cross-over
Input: Randomly selected two *group CA* (CA_1 & CA_2) from present population(PP).
Output: group CA for the next generation.
Step 1: Randomly generate a number q between 1 & n. q is the cross over point.
Step 2: Take the first q symbols from CA_1 and the last $(n - q)$ symbols from CA_2 and concat to form CA'.
Step 3: Make the necessary modifications of CA' at the cross over point to conform with the *pseudo-chromosome format* (*Section 7.4.1*). It generates *group CA* for the next generation.

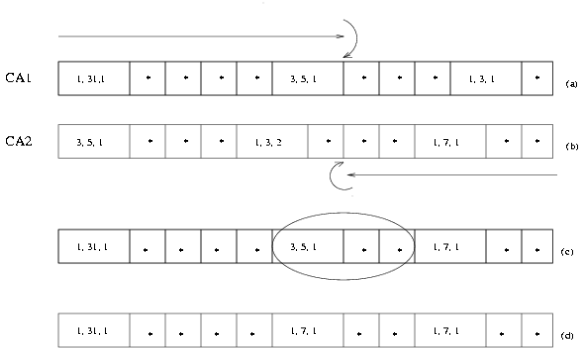


Figure 7.4: An example of cross-over technique

7.4.5 Mutation Algorithm

The mutation algorithm emulates the normal mutation scheme. It makes some minimal changes in the existing *group CA* of PP (Present Population) to form a new *group CA* for next generation. We employ single point mutation.

In the mutation algorithm, an elementary divisor $\phi_i(x)^{n_i}$ is replaced with $\hat{\phi}_i(x)^{\hat{n}_i}$, keeping the degree as it is. To accomplish this, the $< \mu_i, k_i, n_i >$ representing $\phi_i(x)^{n_i}$ is randomly changed to $< \hat{\mu}_i, \hat{k}_i, \hat{n}_i >$ while maintaining the relation $[\mu_i \times k_i + 1] \times n_i = [\hat{\mu}_i \times \hat{k}_i + 1] \times \hat{n}_i]$. *Fig. 7.5* shows two examples of mutation. In the first case, the elementary triplet changes from $< 3, 5, 1 >$ to $< 1, 15, 1 >$. The powers of the elementary divisors remain same after mutation. In the second case, the elementary triplet changes from $< 1, 3, 2 >$ to $< 3, 5, 1 >$ while the power changes from 2 to 1.

The formal mutation algorithm is given below.

Algorithm 7.3 Mutation Algorithm
Input: Randomly select one *group CA* of PP.
Output: group CA for the next generation.

Step 1: Take the *pseudo-chromosome format* of the *group CA*.
Step 2: Randomly select an elementary divisor $\phi_i(x)^{n_i}$.
Step 3: Generate an elementary divisor $\hat{\phi}_i(x)^{\hat{n}_i}$ and synthesize $\hat{T}_i(\hat{n}_i)$ corresponding to the elementary divisor.
Step 4: Replace the existing $T_i(n_i)$ with the $\hat{T}_i(\hat{n}_i)$.

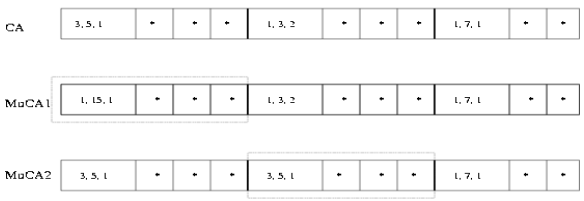


Figure 7.5: An example of mutation technique

Selection, Crossover and Mutation Probabilities: From the study of GA evolutions, we have set the associated parameters to derive population for the next generation out of PP (Present Population). The population size at each generation is set to 50. The crossover probability (p_c) is set to 0.8, while the probability of mutation (p_m) is set as 0.001. We follow the elitist model to carry forward 10 best solutions for the next generation.

7.5 Experimental Results

This section reports a detailed study on the design of proposed TPG. The first part of this section contains the findings of the feasibility study pertaining to this TPG design. The study is performed with different classes of prohibited patterns and prohibited functions for a large number of CUTs. Finally, we provide the fault coverage of the proposed TPG for $ISCAS$ benchmark circuits.

7.5.1 Feasibility Study

Real life data in respect of PPS for a CUT is proprietary in nature and not usually available. In the absence of real life data the experiments are conducted with randomly synthesized data.

The number of prohibited patterns, structure of the prohibited functions (PFs) - that is, the number of prohibited PIs responsible for the formation of PF and the number of minterms in the PF affect the success rate of getting the desired TPG for a CUT with n PIs. The efficiency of the scheme, proposed, can be evaluated in respect of the percentage of free space available for test generation, success rate of the design, the randomness quality of the patterns generated by the TC, and the complexity of the design.

The complexity of the design is measured as the number of generations of the evolution scheme is required to find a TPG for a given PPS. The *free space* is computed as
$freespace\ percent = \frac{\#\text{ test patterns in TC without PPS}}{\text{Total no. of } 2^n \text{ patterns}} \times 100.$
The number of prohibitive patterns for a CUT is expected to be very small, it is typically

around 15. for experimentation we have considered randomly generated 100 different sets of PPS for a CUT with n PIs. Then the scheme is employed to arrive at the desired TPG for each of the sets. The number of times the desired TPG is resulted out of those 100 trials is the success rate of the proposed scheme for this particular value of n.

The following are the summary of the experimental results showing the success rate, percentage of free space achieved for a design and the complexity of the scheme to arrive at the final TPG for a given PPS or the given PF of the CUT.

1. *Table 8.1* depicts the summary of the success rate in designing the TPG that generates good quality pseudo random patterns while avoiding generation of the given PPS. The value of n and $|PPS|$, the cardinality of PPS, are noted in the *columns* 1 & 2. *Column 3* denotes the length of TC (Target Cycle) used to generate the test patterns. The free space % available, without encountering any prohibited patterns, for test generation is shown in *Column 4*. The results of *Table 8.1* are shown for $|PPS|$=10 and 15. It is found that the available free space reduces with the increase of number of prohibited patterns. The more comprehensive result of this fact is shown in *Fig.7.6*. It depicts the effect of increasing the $|PPS|$ in achieving the free space for a particular value of n (number of PIs of a CUT).

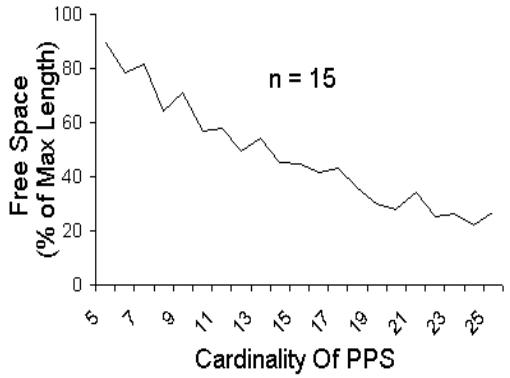


Figure 7.6: Variation of free space with the cardinality of PPS.

2. The experimentation with Prohibited Functions (PFs) are performed for the CUTs having number of PIs $>$ 20. The results of these experiments are noted in the graphs *Fig.7.7 - Fig.7.10*.

(a). *Fig.7.7* depicts the success rate of the design for a desired amount of free space (say 3,00,000) without prohibited patterns. Assuming a fixed number of PIs are participating to form the prohibited function (with fixed number of minterms). The success rate of the design increases with the number of PIs. For the example presentation, it achieves 100% when number of PIs>30. However, the complexity of the proposed design reduces with the increase of $\#PI$s (*Fig.7.8*).

(b) For a fixed number of PIs (n) if the number of prohibited PIs increases, then the number of generations to achieve the desired TPG also increases for a specific amount of

Table 7.1: Success rate of the TPG design

#Cell	$\|PPS\|$	TC	$FreeSpace$ %
8	10	217	59.76
9	10	381	62.30
10	10	1023	56.73
11	10	1953	72.31
12	10	3255	63.96
13	10	8001	67.78
14	10	15841	55.02
15	10	27559	51.53
16	10	63457	64.16
17	10	131071	57.78
18	10	262143	70.41
19	10	458745	57.70
20	10	1040257	50.55
21	10	2097151	58.41
22	10	4063201	65.62
8	15	225	44.14
9	15	465	48.63
10	15	889	45.41
11	15	1905	33.64
12	15	3937	39.11
13	15	8191	39.13
14	15	15841	41.00
15	15	31705	50.00
16	15	59055	44.90
17	15	82677	34.80
18	15	259969	37.70
19	15	458745	45.70
20	15	1040257	45.54
21	15	1966065	49.51
22	15	3138051	42.65

free space. *Fig. 7.9* shows the number of generations required in designing the 25-cell TPG for a CUT. The number of prohibited minterms is assumed to be 2^{p-1} - p being the number of prohibited PIs.

(c) For fixed value of n (number of PIs) and p (# prohibited PIs), the complexity increases with the number of prohibited minterms that are involved to form the PF. *Fig. 7.10* illustrates the fact 25-cell TPG, where $p = 8$ and the the target free space is 3,00,000.

3. **Study of randomness property:** The randomness property of the patterns generated by the TC (Target Cycle), for different values of n, are studied, based on the metric proposed in [**?**] and *DiehardC* [1]. The comparative studies on randomness quality of the patterns generated by the TC and the corresponding maximal length CA are presented in the *tables 8.3* & 8.2.

DiehardC 1.01 is a public domain tool which supports randomness testing of a set of patterns. It consists of 15 different tests. The results of 10 tests are noted in *Table 8.2*.

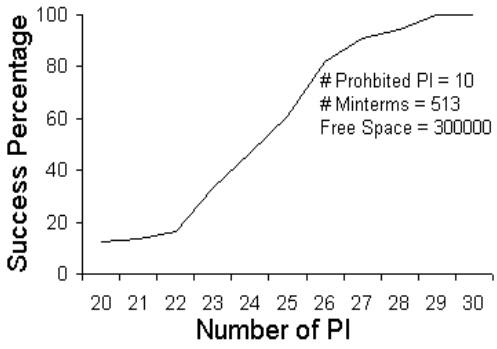


Figure 7.7: Variation of success rate with the number of PIs.

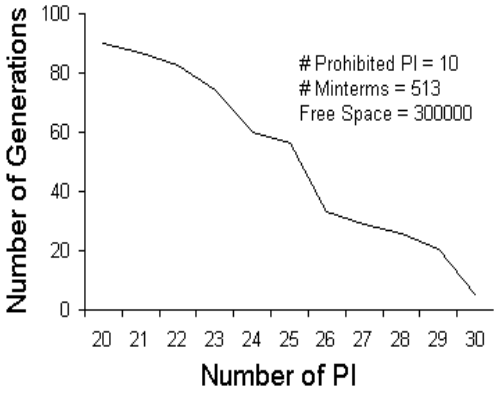


Figure 7.8: Variation of complexity with the number of PIs.

Each test produces a set of 'p' values. For a pattern set with good randomness quality, the values of ps will be uniformly distributed between 0.001 and 0.999.

The first columns of the tables depict the names of the tests. The columns under the heading of 'Max' specify the test results for an n-cell maximal length CA, while the columns under TPG signify the result out of the patterns generated by the proposed design. Each of the tests is performed for a number of runs with different seeds. The results noted for maxlength CA and the proposed TPG are the average of the results produced with different seeds.

In *Table 8.3*, the 'pass' implies that the test succeeds at least for 75% cases while for *Table 8.2* it means - the 'p'value is evenly distributed on [0,1] for at least 75% of seeds.

The results reported in *Table 8.3* & 8.2 establish the fact that the randomness quality of the patterns generated by the TC (proposed TPG) is as good as that of maximal length CA.

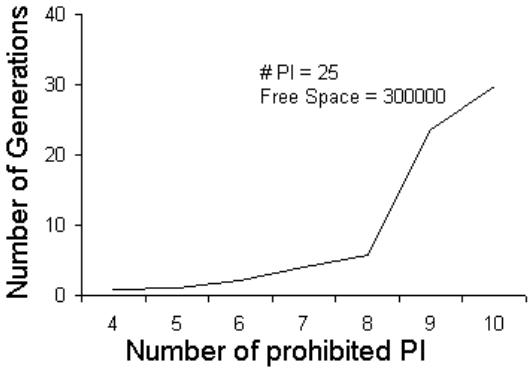


Figure 7.9: Complexity of design with increase of # Prohibited PIs.

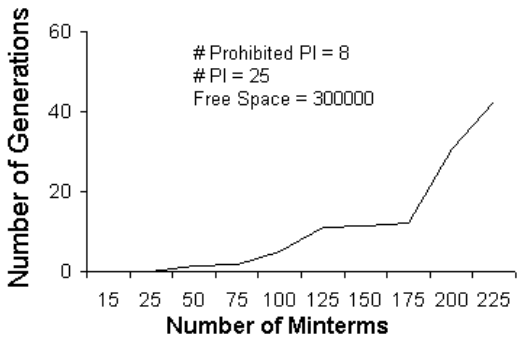


Figure 7.10: Complexity of design with increase of # Minterms.

7.5.2 The fault coverage

It is observed that the TPG designed with the proposed scheme is as powerful as the corresponding maximal length CA based test pattern generator in respect of fault coverage and number of test patterns required to achieve the desired fault coverage. The fault simulation is done for large number of *ISCAS benchmark* circuits in the framework of *Cadence* fault simulator $verifault$. *Table 8.4* compares the fault coverage shown by maximal length CA and the TC of the proposed TPG in the *columns* 4 and 5 respectively. The fault coverage figures are expressed in terms of

$$faultcoverage = \frac{\text{Total no. of detected faults}}{\text{Total no. of faults in the CUT}},$$

while the FFs of the sequential circuits are assumed to be initialized to 0. A benchmark circuit is tested with the same number of test vectors for both the designs. *Column 3* indicates the number of test vectors applied for the test.

Table 8.4 reports the results of testing of 21 combinational and sequential circuits with maximal length CA and as well as with the patterns generated by the TC of the TPG designed with the algorithm proposed in this paper. It can be observed that out of 21

Table 7.2: Randomness Test I

Random Test	$n = 9$		$n = 15$ *to* 20	
	Max	TPG	Max	TPG
Gap test	pass	pass	pass	pass
Run test	pass	fail	pass	pass
Serial corr test	pass	pass	pass	pass
Equidist. test	fail	fail	fail	fail
Auto-corr test	pass	pass	pass	pass
Cross-corr test	pass	fail	pass	pass

cases, the fault coverage of the proposed TPG:

(i) is same or better for 8 cases (marked with *), and

(ii) worse for 13 cases

than the result obtained with maximal length CA. The difference in fault coverage between the two schemes is marginal.

Table 7.3: Randomness Test II

Random Test	$n = 24$		$n = 32$		$n = 48$	
	Max	TPG	Max	TPG	Max	TPG
Overlap Sum	pass	pass	pass	pass	pass	pass
Run	pass	pass	pass	pass	pass	pass
3Dsphere	pass	pass	pass	pass	fail	fail
Parking lot	fail	fail	fail	fail	fail	fail
B'day spacing	fail	fail	fail	fail	fail	fail
Craps	pass	pass	pass	pass	pass	pass
Minimum Dist	fail	fail	fail	fail	fail	fail
Overlap 5-permut	fail	fail	fail	fail	pass	pass
DNA	fail	fail	fail	fail	pass	fail
Squeeze	fail	pass	fail	fail	pass	fail

7.6 Conclusion

The paper presents an elegant solution for the problem of designing a TPG that generates good quality pseudo random test patterns while avoiding generation of Prohibited Pattern Set (PPS) for a given CUT. The design methodology proposed in the paper ensures PPS free TPG considering both the random prohibited patterns and the functions of PIs declared prohibited for a CUT. The reported solution, evolved through GA, does not incur any extra area overhead than the conventional $CA/LFSR$ based TPG. Exhaustive experimentation confirms that the the TPG maintains the fault efficiency in a CUT that could be achieved through a maximal length $CA/LFSR$ based design.

Table 7.4: Comparison of Test Results

Circuit Name	# PI	# Test Vector	Fault Coverage (%)	
			Max Len	*TPG*
s349	9	400	84.00	84.00 *
s344	9	400	84.21	84.21 *
s1196	14	12000	94.85	94.04
s1238	14	10000	89.67	89.08
s967	16	9000	98.22	98.12
s1423	17	15000	56.50	53.60
s1269	18	1200	99.18	99.48 *
s3271	26	10000	98.99	98.99 *
c6288	32	60	99.51	99.43
c1908	33	4000	99.41	99.41 *
s5378	35	8000	67.63	67.72 *
s641	35	2000	85.63	85.08
s713	35	2000	81.41	80.72
s35932	35	14000	61.91	59.82
c432	36	400	98.67	99.24 *
c432m	36	4000	83.57	83.96 *
c499	41	600	98.95	98.68
c499m	41	2000	97.78	97.22
c1355	41	1500	98.98	98.11
c1355m	41	12000	92.23	92.17
s3384	43	8000	91.78	91.60

Chapter 8

DESIGN AND CHARACTERIZATION OF CELLULAR AUTOMATA BASED ASSOCIATIVE MEMORY FOR PATTERN RECOGNITION

8.1 Introduction

This paper reports a Cellular Automata (CA) based model of Associative Memory designed to recognize patterns. Characterization of the model in respect of its pattern recognition capability along with other associated parameters has been reported from extensive study of the model. The storage capacity of the model has been found to be higher than $0.2n$ for pattern size of n bits.

Pattern recognition is the study as to how the machines can learn to distinguish patterns of interest from their background. The *Associative Memory* model provides an elegant solution to the problem of identifying the closest match to the patterns learnt/stored [2]. The model, as shown in *Fig.8.1*, divides the entire state space into some pivotal points (say) *a*,*b*,*c*. The pivotal points (patterns) are assumed to be learnt by the machine during its training phase. The states close to a pivotal point are the noisy vectors (patterns) associated with that specific pivotal point. The process of recognition of a pattern with or without noise, amounts to traversing the transient path (*Fig.8.1*) from the given pattern to the closest pivotal point learnt. As a result, the time to recognize a pattern is independent of the number of patterns stored.

Since early 80's the model of associative memory has attracted considerable interest among the researchers [3, 30]. Both sparsely connected machine (Cellular Automata) and densely connected network (Neural Net) have been explored to design the associative memory model for pattern recognition [3, 5, **?**]. The Hopfield's neural net [**?**, **?**, **?**] models a 'general content addressable memory', where the state space is categorized into a number of locally stable points referred to as attractors (*Fig.8.1*). An input to the network initiates flow to a particular stable point (pivot). However, the complex structure of neural net with non-local interconnections has partially restricted its application for design of high speed low cost pattern recognition machine.

The associative memory model around the simple structure of Cellular Automata has been discussed by a number of authors [5][36][**?**]. Most of the CA based associative memory designs concentrated around uniform CA [5, **?**, 36] with same rule applied to each of the

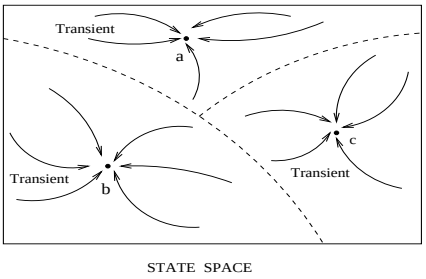


Figure 8.1: Model of associative memory with 3 pivotal points.

CA cells. This structure has restricted the CA based model to evolve as a general purpose pattern recognizer [5]. Further, estimation of memory capacity of the CA in relation to its architectural parameters such as number of cells (n), rules of the CA cells, etc has not been explored.

In the above background, this paper proposes an efficient CA model of associative memory for pattern recognition. It employs non-linear rules for different CA cells. This class of hybrid CA is referred to as *Generalized Multiple Attractor CA* ($GMACA$) [25]. It is the generalization of $MACA$ (Multiple Attractor Cellular Automata) that employs only *additive* rules with $XOR/XNOR$ logic [10].

We employ *Genetic Algorithm* (GA) to arrive at the desired $GMACA$ configurations. The non-linear rule space of $GMACA$ has been investigated. The diverse parameters λ, $\mathcal{Z}$, entropy, G-density etc., proposed by the researchers [**?**][**?**][58] [57] to study CA behavior, are employed to characterize $GMACA$ designed for pattern recognition. The results derived from the study confirm the following facts : (i) the $GMACA$ lies in between order and chaos defined as the *edge of chaos* [**?**]; (ii) the complex computation like pattern recognition occurs only at the *edge of chaos*; and (iii) the memorizing capacity of $GMACA$ is more than 20% of its lattice size and is better than Hopfield network.

The design of $GMACA$ based associative memory and its application for pattern recognition are outlined in *Section IV* preceded by the design specification noted in *Section III*. The characterization of the model in respect of different parameters is reported in *Section V*. A brief introduction to *Cellular Automata* follows.

8.2 Cellular Automata

Cellular Automata (CA) are the simple model of spatially extended decentralized systems made up of a number of cells [39]. Each individual cell of a CA is in a specific state which changes over time depending on the states of its neighbors. In this paper we will concentrate on 3-neighborhood (left, self and right) one dimensional CA, each CA cell having two states - 0 or 1. In a two state 3-neighborhood CA, there can be a total of 2^{2^3} i.e, 256 distinct next state functions referred to as the *rule* of CA cell [55]. If the same rule is applied to all the cells, then the CA is referred to as *uniform CA*, else it is a *hybrid CA*. The rule tables for

two such rules, 90 and 150 are illustrated below:

Neighborhood :	111	110	101	100	011	010	001	000	$RuleNo$
(i) NextState :	0	1	0	1	1	0	1	0	90
(ii) NextState :	1	0	0	1	0	1	1	0	150

The first row lists the possible combinations of present states of the neighbors (left, self and right) of the i^{th} cell at time t referred to as $q_i(t)$. The next two rows list the next states of i^{th} cell at time instant $(t+1)$ denoted as $q_i(t+1)$. Decimal equivalent of the 8-bit binary number is referred to as the rule number (90/150) associated with the next state function of CA cell i as defined below:

Rule 90: $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$

Rule 150: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$

A non-linear CA employs all possible 256 rules as next state function, while additive CA [10] employs only additive rules with $XOR/XNOR$ logic. The example additive CA of *Fig.8.2* with rule vector $< 150, 102, 60, 150 >$ is a *Multiple Attractor CA* ($MACA$).

The entire state space of an $MACA$ are divided into disjoint trees rooted at some attractor cycles. The length of an attractor cycle of the example $MACA$ of *Fig.8.2* is 1. An inverted tree is also called attractor basin. States other than the cyclic state of attractor are referred to as Transient States. A transient state when loaded as a seed for the CA, it reaches the attractor cycle after some time steps. The maximum number of time steps needed for any state to reach the attractor cycle is termed as the depth or Transient Length of $MACA$.

This research work generalizes the concept of $MACA$ to *Generalized MACA* ($GMACA$) with the following characteristics : (i) it employs non-linear rules; (ii) its attractor cycle length is greater than or equal to 1; (iii) its tree structure, unlike $MACA$, is not uniform. A $GMACA$ is illustrated in *Fig.8.3*.

Based on different dynamical behavior, Wolfram [55] reported a specific class (called class IV) of CA displaying complex patterns of localized structure (attractor) with long transients. Wolfram predicted that class IV CA with attractors and transients, are capable of doing non-trivial computations. The term *edge of chaos* is the critical point of a system, where a small change can push the system into chaotic behavior or lock the system into a fixed behavior. The logical and complex computations are likely to occur at edge of chaos [**?**]. Packard highlighted that the state transition behavior of class IV CA exhibits the property of a system at the *edge of chaos* [**?**]. Further, Langton [**?**] defined the range of a parameter value (λ) to identify the CA rules that exhibit complex computation.

The above observations motivated us to explore $GMACA$ based associative memory model for pattern recognition. Design guidelines next follows.

8.3 Design Specification of GMACA Model For Pattern Recognition

A pattern recognizer is trained to get familiarized with some specific pattern set $\{\mathcal{P}_1, \cdots, \mathcal{P}_i, \cdots \mathcal{P}_k\}$ so that it can recognize patterns with or without noise. If a new pattern $\acute{\mathcal{P}}_i$ is input to the

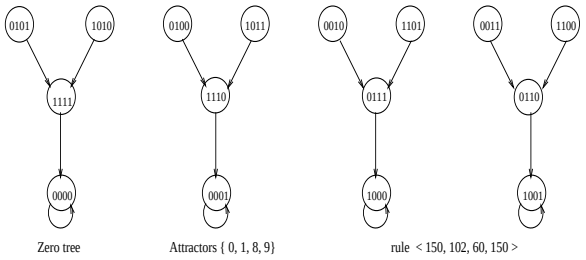


Figure 8.2: State space of a 4-cell $MACA$ with attractors - 0,1,8,9

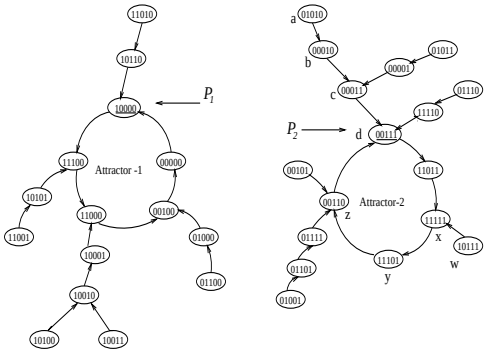


Figure 8.3: State space of a 5 cell GMACA $< 89, 39, 87, 145, 91 >$

system, the pattern recognizer identifies it as $\mathcal{P}_i$, where $\acute{\mathcal{P}}_i$ is the closest match to $\mathcal{P}_i$. The hamming distance between $\acute{\mathcal{P}}_i$ and $\mathcal{P}_i$ ($HD(\acute{\mathcal{P}}_i, \mathcal{P}_i)$) is the least and can be viewed as the measure of noise.

If a $GMACA$ has to function as a pattern recognizer, it has to learn/store the given pattern set $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_i, \cdots \mathcal{P}_k\}$. While the $GMACA$ is run for some time steps with $\acute{\mathcal{P}}_i$ as an input, it returns $\mathcal{P}_i$. Hence, $\acute{\mathcal{P}}_i$ is a transient state close to $\mathcal{P}_i$. Therefore, the design guidelines for the $GMACA$ can be specified by the following two relations:

R1: Each attractor cycle of the $GMACA$ should contain only one pattern ($\mathcal{P}_i \in \mathcal{P}$) to be learnt.

R2: The Hamming Distance ($HD(\acute{\mathcal{P}}_i,\mathcal{P}_i)$) between each state $\acute{\mathcal{P}}_i \in \mathcal{P}_i$-basin and $\mathcal{P}_i$ is lesser than the $HD(\acute{\mathcal{P}}_i,\mathcal{P}_j)$, where $\mathcal{P}_j \in \mathcal{P}$, $\forall j = 1, 2, \cdots k, \&\ \ j \neq i$.

A CA which satisfies both **R1** & **R2** is the desired $GMACA$ for pattern recognition. The 5-cell $GMACA$ of *Fig.8.3* can learn two patterns, $\mathcal{P}_1 = 10000$ and $\mathcal{P}_2 = 00111$. It maintains both the relations R_1 and R_2. The state $\acute{\mathcal{P}} = 11010$ has the hamming distances 2 and 3 from $\mathcal{P}_1$ & P_2 respectively. If $\acute{\mathcal{P}}$ is given as the input to be recognized, then the recognizer must return $\mathcal{P}_1$. The $GMACA$ of *Fig.8.3* if loaded with the pattern $\acute{\mathcal{P}}$= 11010, returns the desired pattern $\mathcal{P}_1 = 10000$ after two time steps.

The search to arrive at a rule vector of a $GMACA$, satisfying both **R1** and **R2**, from all possible combinations of hybrid CA rules is of exponential time complexity. So we fall back on *Genetic Algorithm* (GA) to arrive at the desired $GMACA$ with pattern recognition

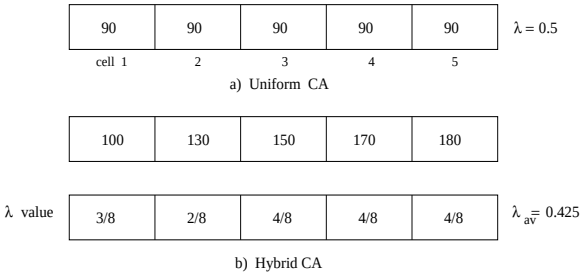


Figure 8.4: λ parameter values of uniform and hybrid CA

capability.

8.4 Evolution of GMACA

The aim of this evolution scheme is to arrive at the $GMACA$ (rule vector) that can perform pattern recognition task. The rule vector of $GMACA$ is viewed as the chromosome for the current GA formulation. The following subsection reports some novel techniques of enhancing the fitness of initial population which ensure fast convergence of the evolution algorithm.

8.4.1 Selection of Initial Population (IP)

Three elegant schemes to construct the IP with better fitness value are proposed next.

IP from λ_{pr} Region

In this scheme the IP is developed from the study of CA rule configuration. The rules are popularly characterized by Langton's parameter λ [**?**]. The number of 1's in a rule is quantified by λ and defined as λ= (number of 1's in the rule number)/8. For example, λ value of Rule 90 (01011010) is 0.5 (*Fig.8.4(a)*). The present research work explores hybrid CA for which we introduce a parameter called λ_{av}, the average value of λ for all the cells in a CA. For example, the λ_{av} for the hybrid CA of *Fig.8.4(b)* is 0.425.

It has been observed that the $GMACA$ rules, acting as efficient pattern recognizers, lie within a specific range of λ_{av} value referred to as λ_{pr} region. So the initial population from that region ensures fast convergence of GA. The following hypothesis characterizes the range of λ_{pr}.

λ_{pr} settles around 0.46 and 0.54 that are roughly equidistant from 0.5.

The IP for the $GMACA$ evolution are picked up from the rule set satisfying λ_{pr} region (*Hypothesis 8.4.1*). The analytical foundation of the hypothesis is reported next followed by experimental validation.

Analytical Foundation : In a $GMACA$, the state $\acute{\mathcal{P}}_i$ while traversing towards $\mathcal{P}_i$ assumes new state $\acute{\mathcal{P}}_i(t)$ at time step t. During traversal, it is not that there is a continuous decrease in hamming distance $HD(\acute{\mathcal{P}}_i(t), \mathcal{P}_i)$ at each time step. In *Fig.8.3*, $\acute{\mathcal{P}}_i(t)$=w=10111 is associated with pivot d=00111 (*Attractor-2*). The hamming distance between w and d is 1. After one time step, $\acute{\mathcal{P}}_i(t+1) = x = 11111$, where the hamming distance $HD(x, d) =$

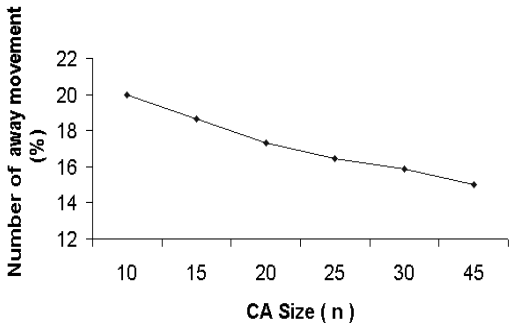


Figure 8.5: Away movements for different values of n

2. During its movement towards d the seed w exhibits an oscillatory motion in terms of hamming distance e.g. 1, 2, 3, 1. Therefore, the $GMACA$ rule configuration has to have the capacity of bringing back its corresponding cell state from $1 \rightarrow 0$ and as well as $0 \rightarrow 1$, as the case may be. The rule cinfigurations having an equal balance of 0 & 1 as the next state function can ensure the oscillatory motion. In otherwords, rule configurations having λ_{av} value around 0.5 are the better candidates for the IP.

This fact is also validated through following experimentations done on the sets of evolved pattern recognizer ($GMACA$). We randomly generate 100 seeds and observe the movements of patterns towards a pivot $\mathcal{P}_i$ in a $GMACA$. Let assume $d' = HD$ ($\acute{\mathcal{P}}_i$(t), $\mathcal{P}_i$) at time step 't', while $d'' = HD(\acute{\mathcal{P}}_i(t+1), \mathcal{P}_i)$. If $d' < d''$, then it refers to as *'away movement'* of pattern from the pivot. The occurances of *away movements* for different values of n (pattern size) are shown in *Fig.8.5*. For each n, we have taken 10 different $GMACA$ and observed movements of 100 patterns for each $GMACA$. Out of 100×10 runs for an n the number of cases we encounter away (%) movement.

It is observed that the *away movement*, noted in *Fig.8.5*, is 15% to 20% to reach a pivot. This validates the analysis that the hamming distance of $\acute{\mathcal{P}}_i$ from $\mathcal{P}_i$ undergoes oscillations in its movement towards $\mathcal{P}_i$.

Let us assume that during the traversal from seed $\acute{\mathcal{P}}_i$ to $\mathcal{P}_i$ the away movement first occurs at time step 't' and continues till $(t+m)$. The minimum number of bits that flip away from the pivot point $\mathcal{P}_i$ during this *away movement* is therefore M =[$HD(\acute{\mathcal{P}}_i$(t+m),$\mathcal{P}_i)$ - $HD(\acute{\mathcal{P}}_i$(t),$\mathcal{P}_i)$]. The maximum value of M for all such durations $(t, t+m)$ for the traversal from a seed $\acute{\mathcal{P}}_i$ to $\mathcal{P}_i$ is the magnitude of 'away movement' for $\acute{\mathcal{P}}_i$. Magnitude of 'away movement' for a $GMACA$ is computed as the average of away movements of a large number of seeds. *Fig.8.6* depicts the variations in magnitude of away movements for different values of n in terms of percentage of n.

It can be observed from *Fig.8.6* that the away movement is limited to 20% of n - that is, for $n = 30$, as many as 5 bits get flipped during the traversal from a seed to the nearest pivot. This fact demands - $GMACA$ rules should have an equal balance of $1s$ and $0s$.

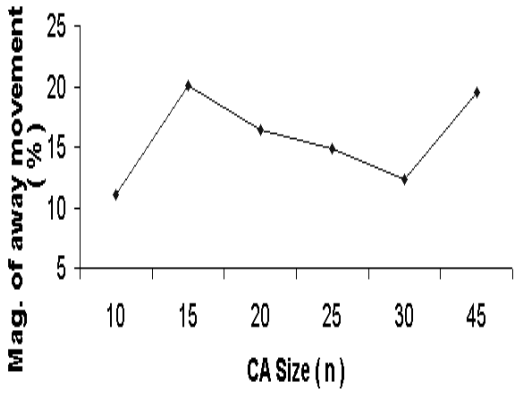


Figure 8.6: Maximum Away Movement of Seeds

Table 8.1: Mean and standard deviations of λ_{av} value of GMACA

CA size (n)	λ_{av} value less than 0.5		λ_{av} value greater than 0.5	
	Mean	Std. Devi^n	Mean	Std. Devi^n
10	0.464423	0.025872	0.530833	0.015723
12	0.454233	0.019912	0.528891	0.019137
15	0.457576	0.025365	0.524074	0.020580
17	0.471336	0.022839	0.529311	0.001763
20	0.463194	0.006875	0.560937	0.013532
22	0.464223	0.031927	0.542419	0.007345
25	0.486250	0.005995	0.517500	0.007500
27	0.461275	0.021345	0.524841	0.024519
30	0.465000	0.024840	0.509375	0.004541

Expt 3 This experimentation validates the selection of IP in the λ_{pr} region. To find λ_{pr}, the experiment has been conducted for $n = 10$ to 30. For each n, 15 different sets of patterns, to be learnt, are selected randomly. The number of patterns in a set is taken as $0{\cdot}15n$, - the maximum number of n-bit patterns that a Hopfield network can recognize. The GA starts with the IP of 50 chromosomes chosen at random and then undergoes evolution till 100% fit rules ($GMACA$) are obtained.

The λ_{av} values for each of $GMACA$ are computed for an n. *Table 8.1* depicts the mean and standard deviation of λ_{av} of the desired $GMACA$ rules. *Column 3* indicates the mean value below 0·5 while *Column 5* indicates the mean above 0·5 for different CA size. The result, as shown in *Table 8.1*, reports that the λ_{av} values of the desired $GMACA$ rules are clustered around in the areas that are roughly equidistant (0.04) from 0.5 and, therefore, λ_{pr} can be chosen as $0.46\pm$ or $0.54\pm$.

Scheme 2 : Graph Resolution Algorithm

This scheme is based on a reverse engineering technique. It constructs a set of $GMACA$ rules for the patterns $P_1, P_2, \cdots, P_{\mathcal{K}}$ to be learnt. The $GMACA$ rules thus are the candidates for IP. The basic concept of mismatch pair algorithm proposed by Myer [**?**] is employed to construct the 3-neighborhood $GMACA$ considering patterns to be learnt as the members

of different attractor cycles of a $GMACA$. The following steps illustrate the design.

- **Step 1** - Construct basin for each of the patterns to be learnt $\mathcal{P}_i$ assuming $\mathcal{P}_i$ as the attractor state (single cycle) and each reachable state having 'p' number of predecessors. The set of states $[\acute{\mathcal{P}}_i]$ which follow **R2** in *Section 8.3* are taken randomly as the predecessor nodes of $\mathcal{P}_i$.

  Fig. 8.7 represents k arbitrary basins for the k number of patterns $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_k$ to be learnt. For example design of *Fig.8.7*, 'p'=3. $\acute{\mathcal{P}}_{j(i)}$ are the j^{th} noisy patterns of $\mathcal{P}_i$, $i = 1, 2, \cdots, k$; $j = 1, 2, \cdots,$.

- **Step 2** - Generate state transition table from the basins [**?**] generated in *Step 1*. *Fig. 8.8* represents a set of representative patterns of a complete state transition table.

- **Step 3** - Generate rule vector of $GMACA$ from the state transition table. The identification of rule for the i^{th} cell of $GMACA$ is done on the basis of the $(i-1)^{th}$, i^{th} and $(i+1)^{th}$ columns of state transition table. Let us consider the identification of rule for the 2^{nd} cell of $GMACA$ from *Fig.8.8*. States for the 8 present state configurations of 1^{st}, 2^{nd} and 3^{rd} columns of *Fig.8.8* are

Neighborhood :	111	110	101	100	011	010	001	000
(i) NextState :	0	x	0	x	1	x	0	0

  x represents don't care. Randomly replacing don't cares by 0/1, we arrive at the rule. Therefore, the rule for the 2^{nd} cell is 01001000 - that is, 72.

  The collision in the state transition table - that is, both 0 and 1 appear as the next state for a present state configuration, is resolved heuristically. In *Fig.8.8*, we have shown collision for the 5^{th} cell. The occurance of '0' in the next state of 011 is n_0=1 while number of occurance of '1' is n_1=3. In resolving the collision (a) we randomly select either 0 or 1 if $n_0 \simeq n_1$. (b) If $n_0 >> n_1$ the next state is taken as 0 while next state is taken as 1 for $n_1 >> n_0$.

For the example design of *Fig.8.8*, the next state for 011 is 1. Different sets of CA rules are derived by constructing approximate state transition diagrams for the attractor set $P_1, P_2, \cdots, P_k$. These rules are the member of IP.

Scheme 3 : Mixed Rules

This scheme combines the *schemes* 1 & 2. The IP comprises of the (i) chromosomes from λ_{pr} region, (ii) chromosomes produced through graph resolution algorithm, (iii) randomly generated chromosomes, and (iv) chromosomes that are produced from the concatenation of fit CA of smaller sizes.

The next two subsections report the guiding principles employed for fast convergence of the solution.

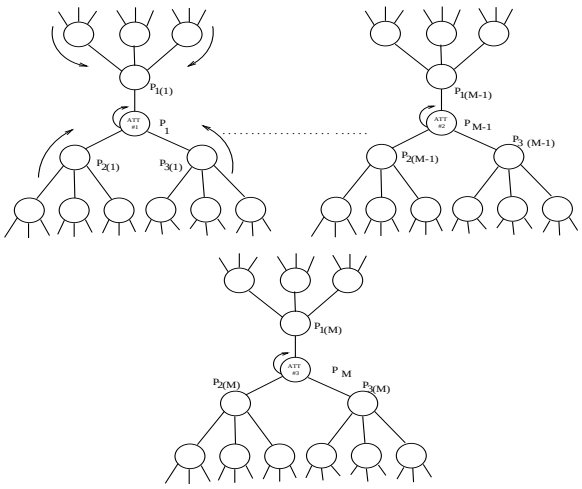


Figure 8.7: Empirical basins created for illustration of Graph Resolution Algorithm

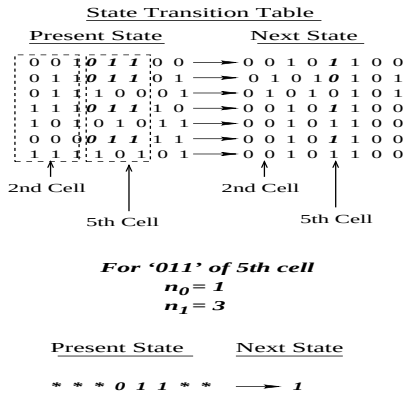


Figure 8.8: An example of State transition table

8.4.2 The Fitness Function

The fitness $\mathcal{F}(C_r)$ of a particular chromosome C_r (that is, CA rule) in a population is determined by the hamming distance between (i) a state $\acute{\mathcal{P}}_j$, evolved from the run (generation), and (ii) the desired attractor state $\mathcal{P}_j$ ($j = 1, 2, \cdots k$) - an element of the learnt pattern set. A chromosome (CA) C_r is run with 300 randomly chosen initial seeds and the fitness of C_r is determined by averaging the fitness for each individual seed.

Let us assume that, the chromosome C_r is run for a maximum iteration $\mathcal{L}_{max}$ with a seed and reaches to a state $\acute{\mathcal{P}}_j$. If $\acute{\mathcal{P}}_j$ is not the member of any attractor cycle, that is, it is still a *transient state*, then the fitness value of the chromosome C_r is considered as zero. On the other hand, if $\acute{\mathcal{P}}_j \in$ an attractor cycle containing $\mathcal{P}_j$, then the fitness of C_r is $\frac{n-|\mathcal{P}_j - \acute{\mathcal{P}}_j|}{n}$, where $\mid \mathcal{P}_j - \acute{\mathcal{P}}_j \mid$ is the hamming distance between $\mathcal{P}_j$ & $\acute{\mathcal{P}}_j$ and n is the maximum possible hamming distance between an input and the learnt pattern. Therefore, the fitness function $\mathcal{F}(C_r)$ can be defined as

$$\mathcal{F}(C_r) = \frac{1}{q} \sum_{j=1}^{q} \frac{n - \mid \mathcal{P}_j - \acute{\mathcal{P}}_j \mid}{n} \tag{8.1}$$

where q is the number of random seeds.

8.4.3 Selection, Crossover and Mutation Probability

From the in-depth study of GA evolutions, we have set the associated parameters to derive NP (Next Population) out of PP (Present Population). The population size at each generation is set to 50. Out of which 35 chromosomes of NP are formed from single point crossover of PP. The 5 chromosomes of NP are formed from single-point mutation of the best 10 chromosomes of PP. We follow the elitist model and carry forward 10 best solutions to the next generation. The detailed experimental results of next subsection validate the GA framework we have set to arrive at the desired solution with feasible computation.

8.4.4 Experimental Results

This section details (i) the convergence rate of the algorithm with different initial selection schemes; and (ii) the memorizing capability of the $GMACA$. While the convergence rate of GA evolution gives the measure of computation cost, the quality of desired solution can be gauged from the memorizing capacity of $GMACA$ finally evolved.

Convergence Rate

The performance evaluation of the four IP selection schemes, with respect to the GA convergence rate and the mean fitness with standard deviation is provided in *Table 8.2*. The experiment has been done on Compaq server in Linux environment. The execution time, reported in the last column of *Table 8.2*, clearly establishes that the execution time increases linearly with n. The entries '$*$' in *Column 2* of *Table 8.2* signify that the GA with random IP does not converge within 1000 generations for $n \geq 35$. The graph showing the number of generations required to converge the GA for different number of CA cells with random (Scheme I) and preselected initial populations (Schemes II, III, IV) are presented in

Table 8.2: Comparison of mean fitness and number of generation required to converge with random and preselected rules

CA size	Random(I)			λ_{pr} region (II)			graph resolution(III)			Mixed rule (IV)			Execution time (sec)
	No of $gene^n$	Mean Fitness	Std. $Devi^n$	No of $gene^n$	Mean Fitness	Std. $Devi^n$	No of $gene^n$	Mean Fitness	Std. $Devi^n$	No of $gene^n$	Mean Fitness	Std. $Devi^n$	
10	166	65.35	7.395	141	67.15	5.249	118	72.86	3.863	128	73.12	7.819	18.183
12	160	64.25	5.889	198	66.19	5.116	132	72.71	3.983	96	73.09	7.516	19.438
15	156	62.28	5.611	253	65.05	5.463	148	72.67	5.611	66	74.81	9.114	21.378
17	328	61.35	6.113	312	63.22	5.714	268	71.89	3.274	94	73.12	8.102	23.061
20	512	61.16	5.828	413	61.78	5.926	364	70.66	5.493	172	72.84	7.436	24.867
22	634	61.81	4.119	521	61.08	4.761	368	71.88	4.106	198	73.21	8.662	27.823
25	721	60.44	4.045	690	61.15	3.746	376	73.46	4.128	210	74.38	8.037	30.344
27	785	60.19	6.719	688	61.84	4.992	412	74.29	3.982	235	73.89	7.159	34.621
30	852	59.33	5.712	718	62.10	5.139	468	75.03	4.106	280	76.88	8.196	37.268
32	*	59.68	6.119	715	61.89	4.329	486	73.67	4.942	312	74.63	9.613	39.916
35	*	59.75	4.981	785	60.62	4.107	525	67.96	4.359	340	70.44	7.824	42.792
37	*	59.67	5.723	809	60.12	5.006	537	67.46	3.746	362	70.87	7.114	45.217
40	*	59.71	5.182	826	59.81	5.628	575	67.21	4.109	410	71.28	8.129	48.882
42	*	58.81	6.091	872	59.81	4.984	599	67.82	3.917	428	70.98	7.197	53.617
45	*	58.48	5.661	935	58.50	5.647	612	66.78	4.008	485	71.17	8.004	57.792

Fig.8.9. It is observed that the convergence rate of GA progressively increases for *Scheme I, II, III & IV* respectively.

The convergence rate of the schemes vary for two reasons. (i) The initial fitness of the population differ and are progressively better in *Schemes I, II, III* & *IV* respectively. (ii) The more subtle reason is that the preselected versions *IV, III* & *II* identify the right schema instances as explained below.

A schema is a set of bit strings that can be described by a template made of ones, zeros and asterisks, the asterisks representing wild cards. An example schema is $< * * * * 010 * * >$. The simultaneous implicit evaluation of a large number of schemas provide *implicit parallelism* to genetic algorithm. The effect of selection is to gradually bias the schemas whose fitness is above average. The preselected rules enhances the convergence because it finds many schemas from the beginning which help the genetic algorithm climb through the correct path. Two important experiments have been further conducted to establish progressive improvement of GA performance of the schemes.

Expt 4 In order to nullify the effect of difference in initial fitness, GA has been evolved with each IP selection scheme to the same level of fitness (say F). (In *Fig.8.10* it is marked with dotted line F.) Subsequently, the number of generations needed to converge the solutions is noted. *Fig.8.10* depicts that the better performance has been observed progressively for the *Schemes I, II, III* & *IV*. For the representative example of *Fig.8.10*, the number of generations (G) required to converge from fitness level F are 686, 531, 369 & 260 for the *schemes I, II, III* & *IV* respectively. The G is computed as $G = b - a$, where b is the total number of generations and a is the number of generations required by the population to reach the fitness level F.

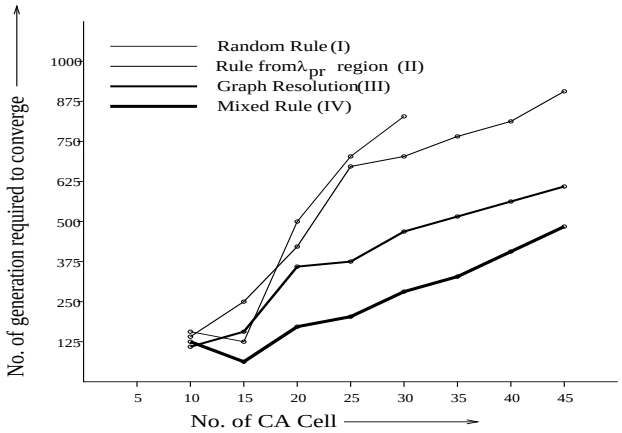


Figure 8.9: Graph showing the number of generations required in different initial populations for different number of CA cells

Expt 5 50 chromosomes (CR_F) of the population that reached the fitness value (say F) are taken. Each such chromosome is compared bit by bit with 10 best chromosomes (CR_b) of final population. We compute *Fraction* of match, represented as the number of bits found same between a pair of chromosomes of size n divided by n, for each pair of (CR_F, CR_b).

For *Scheme IV* the match is as high as 0.7 and the fraction of match on average settles at a value > 0.6 (*Fig.8.10*). This validates higher convergence rate of GA.

Memorizing Capacity of $GMACA$

The experiments to evolve pattern recognizable n-cell $GMACA$ for different values of n are carried out. For each n, 15 different sets of patterns to be trained are selected randomly. The number of patterns to be learnt by the CA is progressively increased. *Table 8.3*

Table 8.3: Performance of the CA based pattern recognizer

CA size (n)	CA based Pattern Recognizer	Conventional Hopfield Net
10	4	2
12	4	2
15	4	2
17	4	3
20	5	3
22	5	3
25	6	4
27	6	4
30	7	5
32	7	5
35	8	5
37	8	6
40	9	6
42	9	6
45	10	7

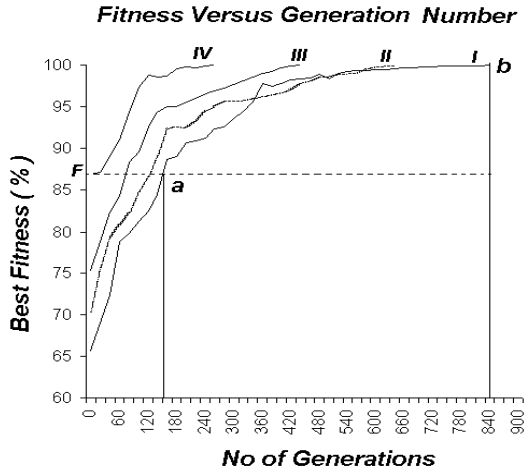


Figure 8.10: Graph showing the number of generations required in different initial populations for a particular CA cell (n=25)

demonstrates the pattern recognition capability of CA based design. *Column 2* of *Table 8.3* depicts the maximum number of patterns that an n-cell CA can memorize - that is, the number of patterns for which the GA has obtained 100% fit rules. The results of conventional Hopfield Net on the same data set are provided in *Column 3* for the sake of comparison. Hopfield net, as reported in [**?**], memorizes 0·15n where size of the pattern is of n-bit.

The experimental results clearly establish the fact that the $GMACA$ have much higher capacity to learn patterns in comparison to Hopfield Net and provides an elegant solution for the pattern recognition problem.

The rule space covered by the $GMACA$ capable of performing pattern recognition, has gone through extensive study. The diverse parameters proposed over the years to characterize the CA behavior are used to identify the class in which the $GMACA$ belongs. The significance of the parameters and the results of the studies on $GMACA$ are reported next.

8.5 $GMACA$ Characterization

The $GMACA$ has been characterized in respect of the parameters proposed by earlier authors [**?**, 36, 55, 58, 57, **?**] to study CA behavior. Discussions on each of the parameters follow.

8.5.1 Space Temporal Study

Dynamical behaviors of space-time patterns generated by the CA provide a guideline to characterize the CA rule space [**?**, 36, 55]. The macroscopic measurements of CA dynamics

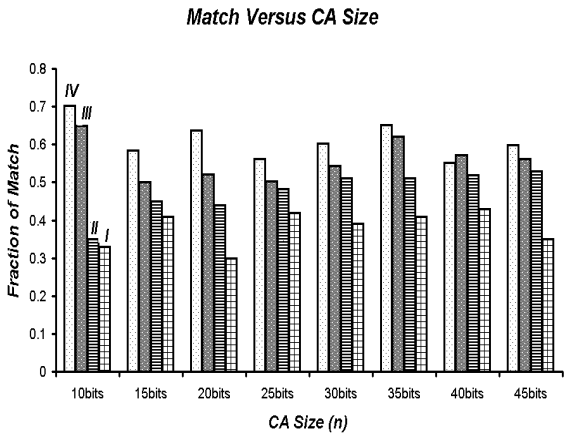


Figure 8.11: Matching in different IP selection schemes

like *entropy*, *mutual information* are studied to classify the $GMACA$ rules.

- **Entropy** is the measure of randomness of a system [**?**]. The maximum *entropy* (close to 1) [**?**] of a system signifies *chaotic behavior*, whereas low *entropy* indicates *ordered behavior*. In case of *complex* system, mean *entropy* is close to the critical value 0·84 [**?**] with high variance.

  To measure the **entropy**, we select a moving window of 10 time steps (w=10). The system has been run for 10000 time steps from a random initial state. The mean *entropy* and the standard deviation from the mean have been computed [57]. For each $GMACA$ the procedure is repeated for 15 times with different random initial states. The values shown in the *Columns 2* and *3* of *Table 8.4* are the mean and standard deviation of *entropy*, computed for the evolved $GMACA$ patterns for different size. It is observed that the evolved $GMACA$ patterns have high standard deviation of mean *entropy* and the mean value clusters around the critical value of 0.84.

- **Mutual information** measures the correlation between patterns generated at a fixed time interval. If a pattern $\mathcal{P}_1$ is the copy of $\mathcal{P}_2$, then *mutual information* between $\mathcal{P}_1$ and $\mathcal{P}_2$ is 1. The *mutual information* between two statistically independent patterns is 0. Both the ordered and chaotic CA rules do not create spatial structures and in effect generate pattern set with low *mutual information*. On the other hand, the complex CA rules create highly correlated structures producing maximum *mutual information* [**?**].

  To measure the *mutual information* of the patterns generated by $GMACA$, we follow the method proposed in [**?**]. Here, we select patterns separated by a particular window of size 6. The *mutual informations* corresponding to the evolved $GMACA$ rules, noted in *Column 4* of *Table 8.4*, are found to be very high (close to 1).

Table 8.4: Space temporal study to categorize CA rule space that display 100% pattern recognizing capability

CA size (n)	Entropy		Mutual Information
	Mean	Std. Devi^n	
10	0.840966	0.072076	0.969
12	0.832549	0.062185	0.957
15	0.820147	0.064446	0.940
17	0.832854	0.071821	0.942
20	0.845909	0.057592	0.978
22	0.841167	0.042171	0.907
25	0.834503	0.065129	0.929
27	0.829847	0.038596	0.917
30	0.839001	0.070335	0.928
32	0.832786	0.052841	0.943
35	0.838102	0.071002	0.964
37	0.841829	0.030996	0.941
40	0.848010	0.061029	0.953
42	0.832871	0.054291	0.948
45	0.839116	0.0510321	0.933

8.5.2 Z-parameter

The CA rule space can be categorized by evaluating $\mathcal{Z}$ [57] parameter. The $\mathcal{Z}$ parameter, proposed by Wuensche, provides an alternative to track the CA behavior very closely. It not only counts the fraction of 1's in the rule table, also takes into account the allocation of rule-table values for the sub-categorization of related neighborhoods. The detailed is reported in [**?**]. The value of $\mathcal{Z}$ varies from 0 to 1. The $\mathcal{Z}$ value close to 1 indicates chaotic behavior of the CA, while $\mathcal{Z} = 0$ indicates order. The intermediate value of $\mathcal{Z}$ identifies the complex CA rules [**?**].

The $\mathcal{Z}$ parameters for the evolved $GMACA$ rules of different size (n) are reported in *Table 8.5*. *Column 2* depicts the mean value of $\mathcal{Z}$ while *Column 3* reports the standard deviation in $\mathcal{Z}$ parameter to arrive at the desired $GMACA$ rules. The values, shown in the table, are in between 0 and 1.

The attractor basins of a CA show the categorization of the CA state-space. The concept of complexity, chaos and phase transitions (edge of chaos) in local dynamics can be related to the convergence in the attractor basin topology of the system. Study on the characterization of attractor basin topology of $GMACA$ is next reported.

8.5.3 Characterization of Attractor Basin

The global parameters such as *G-density*, *Maximum in-degree*, *In-degree frequency histogram*, *Transient length* associated with the attractor basin topology, can be used to characterize $GMACA$ rule space. These parameters point to the degree of convergence of dynamical flow of attractor basin.

Table 8.5: The Values of $\mathcal{Z}$ Parameters of the CA Displaying 100% pattern recognizing capability

CA size (n)	$\mathcal{Z}$ parameter	
	Mean	Std. Devi^n
10	0.618	0.033
12	0.636	0.021
15	0.642	0.037
17	0.617	0.031
20	0.621	0.012
22	0.641	0.007
25	0.622	0.009
27	0.639	0.012
30	0.610	0.029
32	0.634	0.015
35	0.627	0.022
37	0.642	0.016
40	0.633	0.018
42	0.627	0.004
45	0.639	0.033

- **G-density** defines the density of garden-of-eden states, the states without pre-image (predecessor). For example, in *Fig.8.3*, state 'a' is the garden-of-eden state and pre-image of state 'b'.

- **Maximum in-degree** - the maximum number of immediate pre-images of a state is referred to as *Maximum in-degree* of the state. In *Fig.8.3*, *in-degree* of the state 'd' is 3.

  Very high *G-density* (close to 2^n) and significant frequency of high *in-degree* (close to 2^n) imply short and dense trees, which correspond to ordered rules. Whereas, low *G-density* (close to 1) implies low convergence, long sparse trees with branching points having a low *in-degree* (close to 1), implies chaos. The complex rules fall in between these two extremes [**?**].

  The *columns 2* and *3* of *Table 8.6* report the *G-density* and the *Maximum in-degrees* of the evolved $GMACA$. The results show that the proper balance between *G-density* and *Maximum in-degree* is maintained by the $GMACA$.

- **In-degree frequency histogram** - The *In-degree frequency distribution* of a basin of attraction can be plotted as a histogram with horizontal axis representing the value of in-degree and vertical axis as the frequency of in-degree. The *in-degree frequency histogram* gives accurate measurement of attractor basin topology and its convergence. The shapes of histogram indicate different CA dynamics. For complex rule, the histogram follows power law distribution [57].

  Fig.8.12 displays a typical *In-degree* frequency histogram for 15-cell $GMACA$ that recognizes 4 patterns. The histogram exhibits power law distribution. The frequency distribution of all evolved $GMACA$ exhibit similar distribution.

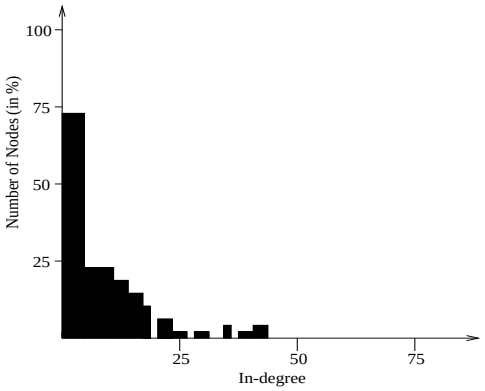


Figure 8.12: In-degree frequency histogram for 15-cell CA

- **Transient length** - The *Transient length* is defined as the time required to reach at the attractor cycle. In *Fig.8.3*, *transient length* for 'a' is 4. At *chaos*, the transient lengths of the patterns are very long, close to 2^n where n is the number of CA cells, whereas the transient lengths are short if order is maintained [**?**].

  The figures in *Column 4* of *Table 8.6* depict the *transient lengths* of the evolved $GMACA$ rules. The transient lengths of spatio-temporal patterns of the $GMACA$ are quite long but much shorter than the length of the transient length (close to 2^n) exhibited by chaotic rules.

Table 8.6: Parameters of Attractor Basin Topology

CA size (n)	G Density	Maximum In-degree	Transient Length
10	70.425	16	213
12	73.687	22	728
15	78.223	39	916
17	78.873	102	1274
20	80.864	174	1559
22	81.642	263	2036
25	83.281	483	3669
27	83.011	512	4128
30	83.342	724	5138
32	84.112	898	5662
35	85.127	1012	7210
37	85.343	2136	7865
40	86.261	4152	9120
42	87.369	6211	9532
45	88.418	7122	11020

The above observations confirm beyond doubt that $GMACA$ rules are complex and lies at the edge of chaos. For the $GMACA$ based associative memory model, this has resulted in : (i) better storage capacity than that of a neural network; and (ii) efficient recognition of patterns without/with noise.

8.6 Conclusion

This paper presents a comprehensive overview of the potential of *Cellular Automata* (CA) to act as an associative memory model. The potential has been explored by using *Genetic Algorithm* to evolve the desired model of CA termed as $GMACA$. Prudent selection of initial population ensures fast convergence of GA. The $GMACA$ has been found to bear the properties of class IV CA that can perform complex computations. The memorizing capacity of $GMACA$ is found to be higher than that of Hopfield Network.

Bibliography

[1] http://stat.fsu.edu/~ geo/diehard.html.

[2] J. Buhmann, R. Divko, and K. Schulter. Associative Memory with high information content. *Phys. rev. A*, 39:2689–92, 1989.

[3] G. A. Carpenter. Neural Network models for Pattern Recognition and Associative Memory. *Neural Networks*, 2(4):243–257, 1989.

[4] K. Cattel and J. C. Muzio. Synthesis of One Dimensional Linear Hybrid Cellular Automata. *IEEE Trans. on CAD*, 15:325–335, 1996.

[5] M. Chady and R. Poli. Evolution of Cellular-automaton-based Associative Memories. *Technical Report no. CSRP-97-15*, May 1997.

[6] S. Chakraborty, D. Roy Chowdhury, and P. Pal Chaudhuri. Theory and Application of Non-Group Cellular Automata for Synthesis of Easily Testable Finite State Machines. *IEEE Trans. on Computers*, 45(7):769–781, July 1996.

[7] S. Chattopadhyay. *Some Studies on Theory and Applications of Additive Cellular Automata*. PhD thesis, I.I.T. Kharagpur, India.

[8] S. Chattopadhyay, S. Adhikari, S. Sengupta, and M. Pal. Highly Regular, Modular, and Cascadable Design of Cellular Automata-Based Pattern Classifier. *IEEE Transaction on VLSI Systems*, 8(6):724–735, December 2000.

[9] P Pal Chaudhuri, D Roy Chowdhury, S Nandi, and S Chatterjee. *Additive Cellular Automata – Theory and Applications*, volume 1. IEEE Computer Society Press, California, USA, ISBN 0-8186-7717-1, 1997.

[10] P. Pal Chaudhuri, D Roy Chowdhury, S. Nandi, and S. Chatterjee. Additive Cellular Automata, Theory and Applications, VOL. 1. *IEEE Computer Society Press, Los Alamitos, California*, (ISBN-0-8186-7717-1), 1997.

[11] P. Cheeseman and J. Stutz. Bayesian Classification (AutoClass): Theory and Results. *In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press*, pages 153–180, 1996.

[12] D. Roy Chowdhury. *Theory and Applications of Additive Cellular Automata for Reliable and Testable VLSI Circuit Design.* PhD thesis, I.I.T. Kharagpur, India, 1992.

[13] D. Roy Chowdhury, S. Basu, I. Sen Gupta, and P. Pal Chaudhuri. A novel scheme for designing error correcting codes using cellular automata. In *Proc. TENCON'91.* New Delhi, India, August 1991.

[14] D. Roy Chowdhury, S. Basu, I. Sen Gupta, and P. Pal Chaudhuri. Design of CAECC — Cellular automata based error correcting code. *IEEE Trans. on Computers*, 43(6):759–764, June 1994.

[15] D. Roy Chowdhury, I. Sen Gupta, and P. Pal Chaudhuri. A class of two-dimensional cellular automata and applications in random pattern testing. *Journal of Electronic Testing : Theory & Applications*, 5:65–80, 1994.

[16] D. Roy Chowdhury, I. Sen Gupta, and P. Pal Chaudhuri. Cellular Automata Based Byte Error Correcting Code. *IEEE Trans. on Computers*, pages 371–382, March 1995.

[17] A. K. Das. *Additive Cellular Automata : Theory and Application as a Built-in Self-test Structure.* PhD thesis, I.I.T. Kharagpur, India, 1990.

[18] A. K. Das and P. Pal Chaudhuri. Efficient characterization of cellular automata. *Proc. IEE (Part E)*, 137(1):81–87, January 1990.

[19] A. K. Das, M. Pandey, A. Gupta, and P. Pal Chaudhuri. Built-in Self Test Structures Around Cellular Automata and Counters. *IEE Proceedings part (E)*, 137(1):81–87, January 1990.

[20] A. K. Das, D. Saha, A. Roy Chowdhury, S. Misra, and P. Pal Chaudhuri. Signature analyzer based on additive cellular automata. In *Proc. 20th Fault Tolerant Computing Systems*, pages 265–272. U.K., June 1990.

[21] A. K. Das, A. Sanyal, and P. Pal Chaudhuri. On the characterization of cellular automata. *Information Science*, 1991.

[22] K. B . Datta. *Matrix and Linear Algebra.* BPB Publications, 1993.

[23] D H Dummit and R M Foote. *Abstract Algebra.* Prentice Hall, $2^n d$ Edition, 1999.

[24] J. C. Muzio et al. Analysis of one-Dimensional Linear Hybrid Cellular Automata over GF(q). *IEEE Trans. on Computers*, 45(7):782–792, July 1996.

[25] N. Ganguly, A. Das, P. Maji, B. K. Sikdar, and P. Pal Chaudhuri. Evolving Cellular Automata Based Associative Memory For Pattern Recognition. *Proceedings of International Conference on High Performance Computing, Hyderabad, India*, 2001.

[26] N. Ganguly, P. Maji, A. Das, B. K. Sikdar, and P. Pal Chaudhuri. Characterization of Non-Linear Cellular Automata Model for Pattern Recognition. *Proceedings of 2002 AFSS International Conference on Fuzzy Systems, Calcutta, India*, pages 214–220, 2002.

[27] F. R. Gantamatcher. The Theory of Matrices. *Chelsa Publishing Co., New York*, I & II, 1959.

[28] Howard Gutowitz. A massively parallel cryptosystem based on cellular automata. In B. Schneier, editor, *Applied Cryptography*. K. Reidel, 1993.

[29] I. N. Herstein. *Topics in Algebra*. Vikas Publishing House Pvt. Ltd., New Delhi, 1976.

[30] J. Hertz, A. Krogh, and R. G. Palmer. Introduction to the Theory of Neural Computation. *Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley*, 1991.

[31] P. D. Hortensius, R. D. McLeod, and H. C. Card. Cellular automata based signature analysis for built-in self-test. *IEEE Trans. on Computers*, C-39(10):1273–1283, October 1990.

[32] J. Kari. Cryptosystems based on reversible cellular automata. *preprint*, April 1992.

[33] P. Maji, N. Ganguly, A. Das, B. K. Sikdar, and P. Pal Chaudhuri. Study of Non-Linear Cellular Automata for Pattern Recognition. *Proceedings of Cellular Automata Conference, Japan*, pages 187–192, 2001.

[34] O. Martin, A. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. *Commun. Math. Phys.*, 93:219, 1984.

[35] S. Misra. *Theory and Application of Additive Cellular Automata for easily testable VLSI circuit design*. PhD thesis, I.I.T. Kharagpur, India, 1992.

[36] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations. *Complex Systems*, 7:89–130, 1993.

[37] S. Nandi. *Additive Cellular Automata : Theory and Application for Testable Circuit Design and Data Encryption*. PhD thesis, I.I.T. Kharagpur, India, 1994.

[38] S. Nandi, B. K. Kar, and P. Pal Chaudhuri. Theory and Application of Cellular Automata in Cryptography. *IEEE Trans. on Computers*, 43(12), December 1994.

[39] J. Von. Neumann. The Theory of Self-Reproducing Automata. *A. W. Burks, Ed. University of Illinois Press, Urbana and London*, 1966.

[40] John Von Neumann. *Probabilistic logics and the synthesis of reliable organisms from unreliable components, J. Von Neumann's Collected Works*. A. Taub, Ed, 1963.

[41] K. Paul, S. Pal Chaudhuri, R. Ghosal, B. Sikdar, and D. Roy Chowdhury. GF(2^p) CA Based Vector Quantization for Fast Encoding of Still Images. In *Proc. of VLSI'00 INDIA*, January 2000.

[42] K. Paul, D. Roy Chowdhury, and P Pal Chowdhury. Theory of Extended Linear Machines. *IEEE Transactions on Computers*, 2002.

[43] K. Paul, A. Roy, P. K. Nandi, B. N Roy, M. Deb Purkhayastha, S. Chattopadhyay, and P. Pal Chaudhuri. Theory and Application of Multiple Attractor Cellular Automata for Fault Diagnosis. In *Proc. of ATS'98 Singapore*, December 1998.

[44] Kolin Paul. Phd thesis theory and application of gf(2^p) cellular automata. *B. E. College (Deemed University), Howrah, India*, 2001.

[45] W. Pries, A. Thanailakis, and H. C. Card. Group properties of cellular automata and VLSI applications. *IEEE Trans. on Computers*, C-35(12):1013–1024, December 1986.

[46] Palas Sarkar. A brief history of cellular automata. In *ACM Computing Systems*, volume 32, pages 80–107, March 2000.

[47] M. Serra and T. Slatear. A Lanczos Algorithm in a Finite Field and its Application. *Journal of Combinatorial Mathematics and Combinatorial Computing*, pages 11–32, April 1990.

[48] B. K. Sikdar, N. Ganguly, P. Majumder, and P. P. Chaudhuri. Design of Multiple Attractor $GF(2^p)$ Cellular Automata for Diagnosis of VLSI Circuits. *Proceedings of* 14^{th} *International Conference on VLSI Design, India*, pages 454–459, January 2001.

[49] Murray R Spiegel. *Theory and Problems of Probability and Statistics*. Schaum's Outline Series, 1980.

[50] H. Stone. *Linear Machines*. Princeton Univ. Press, 1965.

[51] H. S. Stone. *Discrete Mathematical Structures and Their Applications*. Science Research Associates Inc., 1973.

[52] J. Thatcher. Universality in Von Neumann cellular model. In *Tech. Report 03105-30-T, ORA, University of Michigan*, 1964.

[53] Trinh Xuan Thuan. Chaos and harmony. *OXFORD* University Press, 2001.

[54] S. Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55(3):601–644, July 1983.

[55] S. Wolfram. Theory and Application of Cellular Automata. *World Scientific*, 1986.

[56] Stephan Wolfram. Cryptography with cellular automata. *Proceedings of Crypto '85*, pages 429–432, 1985.

[57] A. Wuensche. Classifying Cellular Automata Automatically. *Santa Fe Institute Working Paper*.

[58] A. Wuensche. Complexity in One-D Cellular Automata. *Santa Fe Institute Working Paper*, (94-04-025), 1994.