Study on Equal Length Cellular Automata with Cost Efficient Pseudo-Random Number Generator Targeting Distributed Applications

Thesis submitted by

Arnab Mitra

Doctor of Philosophy (Engineering)

Department of Information Technology

Faculty Council of Engineering & Technology

Jadavpur University

Kolkata, India

2017

JADAVPUR UNIVERSITY

KOLKATA-700032, INDIA

INDEX NO. 46/14/E

1. **Title of the Thesis:**

   Study on Equal Length Cellular Automata with Cost Efficient Pseudo-Random Number Generator Targeting Distributed Applications

2. **Name, Designation & Institution of the Supervisors:**

- **Dr. Anirban Kundu**
  Associate Professor, Department of Information Technology
  Netaji Subhash Engineering College, Kolkata, PIN-700152, India

- **Dr. Matangini Chattopadhyay**
  Director and Professor, School of Education Technology
  Jadavpur University, Kolkata, PIN-700032, India

- **Dr. Samiran Chattopadhyay**
  Professor, Department of Information Technology
  Jadavpur University, Kolkata, PIN-700098, India

3. List of publications:

a) Papers published in journals

- Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S., (2018). On the exploration of equal length cellular automata rules targeting a MapReduce design in cloud, International Journal of Cloud Applications and Computing, 8(2), 1-26. IGI Global.
- Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2017). A cost-efficient one time password-based authentication in cloud environment using equal length cellular automata. Journal of Industrial Information Integration, 5, 17-25. Elsevier.
- Mitra, A., & Teodorescu, H. N. (2016). Detailed analysis of equal length cellular automata with fixed boundaries. Journal of Cellular Automata, 11(5-6), 425-448. Old City Publishing.
- Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2015). An analysis of equal length cellular automata (ELCA) generating linear rules for applications in distributed computing. Journal of Cellular Automata, 10(1-2), 95-117. Old City Publishing.
- Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2014). A novel design with cellular automata for system-under-test in distributed computing. Journal of Convergence Information Technology, 9(6), 55-68. AICIT.
- Mitra, A., Kundu, A., & Das, C. (2014). Random number generators: performance comparison of ELCA and MaxCA. CSI transactions on ICT, 2(2), 117-127. Springer.
- Mitra, A., & Kundu, A. (2012). Cost optimized design technique for pseudo-random numbers in cellular automata. International Journal of Advanced Information Technology, 2(3), 21-36. AIRCC Publishing.

b) Paper published as book chapter

- Mitra, A., & Kundu, A. (2014). Cost optimized random sampling in cellular automata for digital forensic investigations. Computational Intelligence in Digital Forensics: Forensic Investigation and Applications: Studies in Computational Intelligence, 555, 79-95. Springer.

c) Papers published in international conferences

- Mitra, A., Kundu, A., & Chattopadhyay, M. (2014, December). Energy efficient task-pull scheduling using equal length cellular automata in distributed computing. In Proceedings of the Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference on (pp. 40-45). IEEE.

- Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2014, August). Dynamics computation of equal length cellular automata (ELCA) rules. In Proceedings of the Electronics Engineering & Computer Science, (IEMCON), 2014 Fifth International Conference on (pp. 214-220). Elsevier.
- Mitra, A., Kundu, A., & Das, C. (2014, February). Cost effective PRNG using ELCA: a BIST application. In Proceedings of the Automation, Control, Energy and Systems (ACES), 2014 First International Conference on (pp. 1-6). IEEE.
- Mitra, A., & Kundu, A. (2013, September). Cost optimized set of primes generation with cellular automata for stress testing in distributed computing. In Proceedings of the Computational Intelligence: Modeling Techniques and Applications (CIMTA), 2013 First International Conference on (pp. 365-372). Elsevier.
- Mitra, A., & Kundu, A. (2012, September). CA based cost optimized PRNG for Monte-Carlo simulation of distributed computation. In Proceedings of the CUBE International Information Technology Conference on (pp. 332-337). ACM.
- Mitra, A., & Kundu, A. (2012, May). Cost optimized approach to random numbers in cellular automata. In Proceedings of the Computer Science, Engineering and Applications (ICCSEA), 2012 Second International Conference on (pp. 609-618). Springer.

4. List of Patents: Nil.

5. List of Presentations in National/International/Conferences/Workshops:

- Energy efficient task-pull scheduling using equal length cellular automata in distributed computing; Fourth International Conference on Emerging Applications of Information Technology (EAIT)-2014.

- Dynamics computation of equal length cellular automata (ELCA) rules; Fifth International Conference on Electronics Engineering & Computer Science-2014 (IEMCON-2014).

- Cost Effective PRNG using ELCA: a BIST Application; First International Conference on Automation, Control, Energy & Systems (ACES)-2014.

- Cost optimized set of primes generation with cellular automata for stress testing in distributed computing; First International Conference Computational Intelligence: Modeling, Techniques and Applications (CIMTA)-2013.

JADAVPUR UNIVERSITY

KOLKATA-700032, INDIA

CERTIFICATE FROM THE SUPERVISORS

This is to certify that the thesis entitled "**Study on Equal Length Cellular Automata with Cost Efficient Pseudo-Random Number Generator Targeting Distributed Applications**" submitted by Shri **Arnab Mitra,** who got his name registered on **24th November, 2014 (Index. No. 46/14/E)** for the award of Ph. D. (Engg.) degree of Jadavpur University is absolutely based upon his own work under the supervision of **Dr. Anirban Kundu, Dr. Matangini Chattopadhyay** and **Dr. Samiran Chattopadhyay** and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

1.

Signature of the Supervisor and date with Office Seal

Dr. Anirban Kundu

Associate Professor, Department of Information Technology

Netaji Subhash Engineering College, Kolkata 700152, India

2.

Signature of the Supervisor and date with Office Seal

Dr. Matangini Chattopadhyay

Director and Professor, School of Education Technology

Jadavpur University, Kolkata 700032, India

3.

Signature of the Supervisor and date with Office Seal

Dr. Samiran Chattopadhyay

Professor, Department of Information Technology

Jadavpur University, Kolkata 700098, India

Dedicated to

My Mother & Father

Acknowledgement

First and foremost, I wish to express my sincere gratitude to my supervisors Dr. Anirban Kundu, Dr. Matangini Chattopadhyay, and Dr. Samiran Chattopadhyay for their supervisions, encouragements and constant supports that enabled me to complete this thesis. It is their alluring personalities that have changed my approach towards academics during this age. They have also educated me to be regimented in research. I express my sincere appreciation to Dr. Chandra Das, Netaji Subhash Engineering College. I also acknowledge supervisions and guidance from Prof. Horia-Nicolai Teodorescu, Faculty of Electronics, Telecommunications and Information Technology, “Gheorghe Asachi” Technical University of Iasi, Romania where I have completed Erasmus Mundus sponsored ‘cLink’ Ph.D. mobility.

I would like to express my deepest gratitude to my mother, and my father. It is their encouragement and blessings that enabled me to cross the hurdle. My deepest gratitude goes to all my family members, and friends for their encouragements during the progress. I have received uncountable supports in all respects from my family members, and friends who are anxiously waiting for the successful completion of my Ph.D degree. I am also thankful to all those personalities whose names have not been mentioned here as it is impracticable to bring up all their names.

Dated: July, 2017

Kolkata, India. Arnab Mitra

Contents

List of Figures

List of Tables

Abstract

Major concern of engineering and scientific applications is to reduce the computing cost in distributed computing by introducing low cost physical design, reduced energy consumption and reusable model. Cheap and generalized Equal Length Cellular Automata (ELCA) based model is a good choice for modelling of several distributed computing applications. ELCA is of special interest for its characteristics. ELCA is a special classification of Cellular Automata (CA) where all generated CA subspaces (cycles) are of equal lengths. A detailed study on ELCA characteristics, synthesis, analysis and classification along with potential applications of ELCA in distributed computing have been described in this thesis. Inherent strengths of ELCA based models in random pattern generation have also been described. ELCA based simple solutions enhance cost efficiencies in terms of time complexity, space complexity, design complexity and searching of Prohibited Pattern Set (PPS). Three neighbourhood based CA structures in null boundary scenario have been considered for generation of ELCA cycles. A cost-efficient pseudo-random number generator in CA is presented in this thesis. A new approach towards exclusion of Prohibited Pattern Set (PPS) from randomness generating patterns is discussed. An effective approach for set of primes generation in null boundary CA scenario is presented. A complete set of CA rules for ELCA generation and synthesis of ELCA rules are described. Set of necessary and sufficient conditions of CA rules for generation of ELCA cycles are described. An analysis on ELCA linear rules is carried out to investigate the properties of the characteristics matrix and characteristics polynomials. Presence of primitivity or, recursive primitivity in ELCA characteristics polynomial is confirmed, which is a prerequisite criterion for generation of randomness. Mapping in ELCA generated patterns is explored using discrete mathematics representations. Detailed analysis on the dynamics of ELCA rules is presented. All the properties of CA and ELCA so developed have been used to design a cost-efficient solution for different distributed applications such as test pattern generation for reliability assessment, built-in self-test (BIST), system-under-test (SUT), set of primes generation, energy efficient job scheduling, and one time password (OTP) generation in cloud environment.

CHAPTER 1

INTRODUCTION

1.1. Introduction

Cellular Automata (CA) are used as a tool for modelling of any dynamic complex system [1, 2]. CA are discrete over space and time. Several researchers have carried out investigations on CA rules. S. Wolfram classified CA into four groups based on their behaviors [1] which is described in Table 1.1.1.

Table 1.1.1. CA classification by S. Wolfram

CA Class	Observations
I.	Homogeneous state spaces are evolved to limit points.
II.	Simple separated periodic structures are evolved to limit cycles.
III.	Chaotic aperiodic patterns are produced; chaotic behaviors of the kind associated with strange attractor are evolved.
IV.	Complex patterns of localized structures are evolved.

In this thesis, we have first explored the set of CA rules responsible for generation of Equal Length Cellular Automata (ELCA), their detailed characteristics and dynamics and some uses in selected Distributed Computing Applications.

Distributed Computing [3, 4] is a new computing technique that solves a problem through collaboration of geographically distant computing units by performing specific tasks in less time. In such computing, a single algorithm is parallelized in a sophisticated manner and then executed by several computing units. All the partial solutions of these tasks are combined to form a single solution. Several distributed applications are required to be processed in a cost-effective and reliable manner.

Studies with CA in recent years explore that they are also capable of modelling different applications in distributed computing as they are elegant mathematical structures which can have a low cost, and simple VLSI (very large-scale integration) implementation. In this thesis, we have focussed on CA based solutions for some chosen distributed applications.

1.2. Studies on ELCA

In the past, many researchers have worked on several aspects of group, and nongroup CA. S. Nandi et al. have introduced non-maximal length CA for applications in cryptosystems [5]. P. P. Chaudhuri et al. have discussed about maximum length CA, theory and applications of several additive CA rules [2]. N. Ganguly et al. have presented a detailed survey on CA, and its potential applications [6]. S. Ghosh et al. have presented invertible CA, and equal length cycle CA as a special case of invertible CA for potential application in protein synthesis [7, 8]. I. Aguiar et al. have presented CA dynamics for rule ‘26’, and ‘154’ at different boundary conditions [9]. We have observed that few detailed studies on equal length cycles generated by group CA and their detailed characteristics, dynamics at different boundary were available in the literature which may be beneficial for modeling a number of distributed applications.

We have studied Equal Length Cellular Automata (ELCA) and their properties. An n-cell CA produces all '2^n' states in the form of equal length cycles. Mathematically, generation of ELCA is described in following Equation 1.1 [10].

$$2^n = 2^m * 2^{n-m} \text{ for } n \geq 1 \text{ and } m = 1,2,3,\dots,(n-1) \quad (1.1)$$

where, '2^n' is total number of states of an n-cell CA, '2^m' is total number of equal length cycles and '2^{m-n}' is length of cycles.

We have found that ELCA structures are produced from balanced CA rules at null boundary scenario. We have explored Linear CA rules and Complemented linear rules for ELCA generation. We have also shown that fixed length cycles are produced with rule 51, and 204; variable length cycles are found with rule 153. It was further concluded that generation of equal length cycles is not dependent on boundary values of CA. We have developed a characteristics matrix for uniform ELCA, and characteristics matrices for a number of hybrid ELCA.

We have also shown that the following properties hold for ELCA [10, 11].

- Minimal polynomials for an n-cell uniform CA using rule '204' and '51' are the same.
- Characteristic polynomials for an n-cell uniform CA using rule '204' and '51' are same.
- Variable length ELCA can be generated from hybrid CA if the characteristic matrix has a form of tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal or off-diagonal positions are occupied by "1, 0" pattern sequence. Corresponding determinant of the characteristic matrix is equal to one.
- Variable length ELCA can be generated from hybrid CA if characteristic matrix has a form of tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal or off-diagonal positions are occupied by "1, 1, 0" pattern sequence. Corresponding determinant of characteristic matrix is equal to one.
- ELCA cannot be generated from hybrid CA if characteristic matrix has a form of tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal and off-diagonal positions are occupied by "1, 0" or "1, 1, 0" pattern sequence. Corresponding determinant of characteristic matrix is equal to zero.

Following pair of necessary and sufficient conditions are found for group CA to distinguish between the generation of Maximum-length CA (MaxCA) and ELCA [10].

- Necessary condition for ELCA and MaxCA generation: determinant of $[n \times n]$ characteristic matrix should be equal to one.
- Sufficient condition for ELCA generation: trace of $[n \times n]$ characteristic matrix should be equal to 'n'.
- Sufficient condition for MaxCA generation: trace of $[n \times n]$ characteristic matrix should be equal to '$\lceil \frac{n}{2} \rceil$'.

1.3. ELCA based models targeting Distributed Computing Applications

We have considered the following distributed applications in which group CA specifically ELCA can be effectively used.

- Pseudo-random pattern generation in Monte-Carlo simulation for assessment of reliability
- System-Under-Test (SUT)
- Equally populated task pull allocation for job scheduling
- Set of Primes generation for Stress testing
- OTP based authentication in cloud environment

In the past, several researchers have worked on CA based reliability assessment. E. Zio has focused on solutions of advanced network reliability problems by means of CA and Mont Carlo sampling [12]. E. Jio et al. have focused on combinational usage of CA and Monte-Carlo simulator for computing the availability of complex network systems [13]. B. Canizes et al. have focused on hybrid fuzzy monte-carlo method towards reliability assessment [14]. In all these above-mentioned works, we have observed that heterogeneous models have been considered. Moreover, cost effectiveness has not been focused in randomness generation. We have developed a CA based cost optimized PRNG for Monte-Carlo simulation in distributed computation [15].

Several researchers have worked on built-in self-test (BIST) pattern generation. I. Kokolakis, et al. have focused on competitive results for CA patterns with reference to linear feedback shift register (LFSR) generated patterns in built-in self-test (BIST) [16]. S. Das et al. focused on non-linear CA based PPS free BIST test pattern generation [17, 18]. S. Jamuna has focused on VHDL and LFSR uses in BIST architecture [19]. After going through these research papers, we have found out that LFSR based models or heterogeneous CA models have primarily been considered. Moreover, cost effectiveness has not been focused in randomness generation and it must be noted that

PPS exclusion process is crucial. We have developed a cost effective PRNG using ELCA and have applied that towards BIST generation [20-24].

In terms of past works in System-under-test (SUT), efficiencies for fault insertion (FI) usage in SUT have been examined by M. Cukier et al. [25]. C. Tr¨odhandl et al. have focused on conceptual FI framework based on hybrid hardware-software method of fault injection for distributed system [26]. I. Hsu. et al. have focused on combined usage of software-implemented FI and virtualization for an automated validation and analysis of distributed SUT [27]. A.W. Ulrich et al. have focused on system test architecture for distributed computing [28, 29]. We have observed that CA models have not been used. Besides, randomness for test cases in SUT architecture has not been assured, and PPS exclusion has not been focused for test case generation. We have presented CA based SUT for Distributed Computing in [30].

CA based Job Scheduling has been another area of active research. F. Seredynski et al. have proposed a CA based genetic algorithmic approach for multi-processor scheduling in distributed environment [31]. Co-evolutionary genetic algorithm has been introduced for the definition of a program graph neighbourhood for determining rules in distributed scheduling problems by M.S. Laghari et al. [32, 33]. P. Agrawal et al. have discussed periodic boundary CA based optimal scheduling in distributed computing [34]. We have observed that heterogeneous models, and scheduling with nonlinear CA rules have been introduced. Besides, randomness has not been focused in generated job pulls. Generation of equal length task pull has also not been focused. We have presented an energy efficient task-pull scheduling for Distributed Computing using ELCA [35].

Prime generation has been another interesting research topic. Generation of Primes by a one-dimensional real-time iterative array has been focused by P. C. Fischer [36]. Signals in one-dimensional cellular automata has been introduced by J. Mozoyer targeting prime generation [37]. Prime generation with CA has been described by H. Umeo et al [38-40]. We have observed that individual prime number generation has been attempted but a set of random prime generation has not been properly dealt with. Moreover, cost effectiveness of the generation algorithm has not been considered. We have designed a cost optimized algorithm for generating set of primes using Cellular Automata which can be applied in stress testing of distributed applications [41].

Lots of research works on CA based authentication and one time password (OTP) generation have gained importance. S. Nandi et al. have presented group CA based data security and authentication [5]. M. Mukherjee et al. have presented CA based authentication [42]. J. C. Jeon et al. have presented non-group CA based one time password (OTP) authentication scheme in wireless networks [43]. R. Yampolskiy et al. have presented CA rule 30 based data security and authentication [44]. We have observed that controllable generation of multiple OTP sets was not considered. Generation of equally populated OTP sets was also not discussed. We have explored

ELCA dynamics and designed a cost efficient One Time Password-based authentication in cloud environment using ELCA [45, 46].

1.4. Organisation of dissertation

Organisation of this dissertation is as follows. A brief introduction to CA, randomness and MaxCA are presented in Chapter 2. Design of a novel CA based PRNG and its potential application in reliability assessment, BIST, and SUT are explored in Chapter 3. Performance comparison among several random number generators (RNGs) is presented in Chapter 4. CA based set of primes generation targeting stress testing in distributed computing is also described in Chapter 4. A set of CA rules responsible for generation of ELCA are explored in Chapter 5 along with a detailed analysis of ELCA generating rules and its finite state machine (FSM) representation using discrete mathematics. A potential application (energy efficient task-pull allocation in distributed computing) using ELCA model is also presented in Chapter 5. The dynamics of ELCA rules and its potential application in OTP based authentication are presented in Chapter 6. Concluding remarks and future scope are presented in Chapter 7. Literature survey of the related topics have been reported in each of these chapters.

CHAPTER 2

PRELIMINARY CONCEPTS

2.1. Cellular Automata

2.1.1. Introduction

A Cellular Automaton (pl. Cellular Automata, in short CA) is a discrete mathematical model. CA is referred to as a collection of 'valued' cells on a grid of specified shape (dimension). CA evolves through several discrete time steps according to a set of rules; each of the cells is in one of a finite number of states, such as either value "1 (ON)" or value "0 (OFF)". A typical hardware implementation of CA cell using flip flop is presented in Fig. 2.1.1 [2].

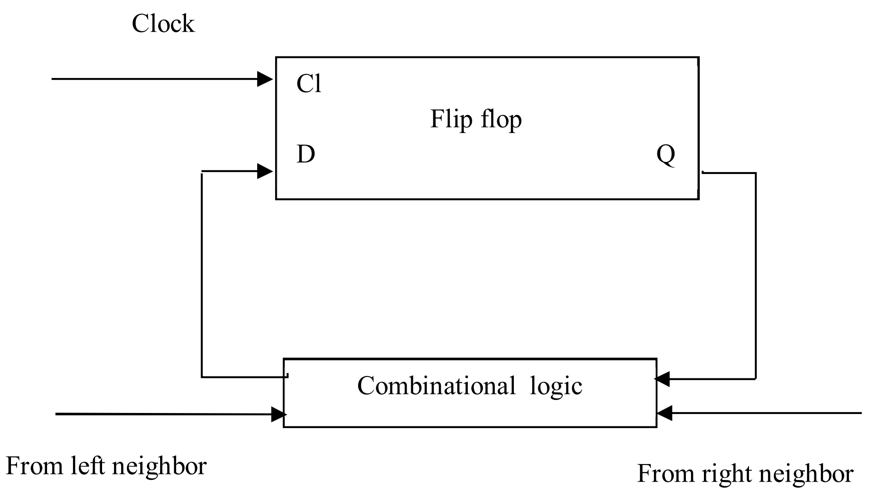


Fig. 2.1.1. A typical hardware implementation of CA [2]

A one-dimensional CA is defined as Equation 2.1.1 [37].

$$M_{CA} = (Q, \#, \mathrm{L}, \delta) \tag{2.1.1}$$

here "Q" is the finite set of all states; "#" is a special border state, but never contained in "Q";

"L" is another state such that δ (L, L, L) = δ (#, L, L) = L (the quiescent state);

and transition function "δ" is a mapping as defined in Equation 2.1.2 [37].

$$\delta : Q\,U\,\{\#\} * Q * Q \rightarrow Q \tag{2.1.2}$$

Null boundary and periodic boundary CA- *If leftmost and rightmost cell are grounded, then it is referred to as null boundary CA. In periodic boundary condition, the leftmost cell and rightmost cell of the CA are connected instead of being grounded.*

A typical N-cell null boundary CA is presented in Fig. 2.1.2 [2].

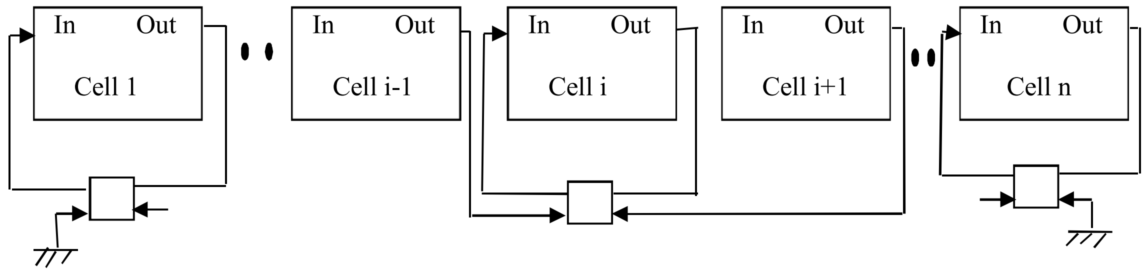


Fig. 2.1.2. Typical n-cell null boundary CA structure [2]

Schematic diagram of 3-cell null boundary CA scenario is presented in Fig. 2.1.3.

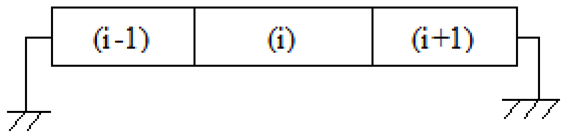


Fig. 2.1.3. Typical structure of 3-cell null boundary CA

Mathematically CA has defined by J. M. Baetens et al. as a function of sextuples as described in Equation 2.1.3 [47].

$$CA = < \Gamma, S, \mathrm{s}, \mathrm{s}_0, N, \Phi > \tag{2.1.3}$$

where, countably infinite tessellation of an n-dimensional Euclidean Space R^n is represented by Γ, consisting of cells c_i such that $i \in \mathbb{N}$;

finite set of k states referred to as S where often $S \subset \mathbb{N}$;

output mapping function $s: \Gamma \times \mathbb{N} \to S$ produces the state value of cell c_i at the t^{th} discrete time step denoted by $s(c_i, t)$;

initial condition for every cell c_i i.e., $s(c_i, t) = s_0(c_i)$ is assigned by function $s_0: \Gamma \to S$;

every cell c_i is mapped to a finite sequence $N(c_i) = (c_{ij})_{j=1}^{|N(c_i)|}$ by neighborhood function $N: \Gamma \to \bigcup_{P=1}^{\infty} \Gamma^P$ and $|N(c_i)|$ is the number of all distinct cells c_{ij} ;

$\Phi = (\phi_i)_{i \in \mathbb{N}}$ is a family of functions $\phi_i: S^{|N(c_i)|} \to S$ such that each ϕ_i is responsible for the dynamics of cell c_i, i.e., $s(c_i, t+1) = \phi_i(\tilde{s}(N(c_i), t))$, as $(\tilde{s}(N(c_i), t)) = (\tilde{s}(N(c_i), t))_{j=1}^{|N(c_i)|}$.

Following definitions as found in [2] are presented to understand the CA basics.

Uniform CA- *If same rule is applied in all the CA cells, it is titled Uniform CA, and otherwise it is titled as a hybrid CA.*

Group CA- *If produced subspaces in the transition diagram of a CA are form of circular queue only, it is entitled as Group CA; else it is mentioned as Non-group CA.*

Linear CA- *If a rule of a CA cell involves only XOR logic then it is a linear rule. A CA with all linear rules is a linear CA.*

Complemented CA rule- *All rules involving XNOR logic is said complemented rule.*

Additive CA- *A CA having a combination of XOR and XNOR rules is called an Additive CA.*

Non-additive CA rule- *Rules with AND-OR logic is called non-additive rules.*

Group CA and non-group CA- *If produced subspaces of the transition diagram of a CA are of a form of circular queue only, it is referred to as Group CA; else it is referred to as Non-group CA.*

Group CA and non-group CA have been described in Fig. 2.1.4.

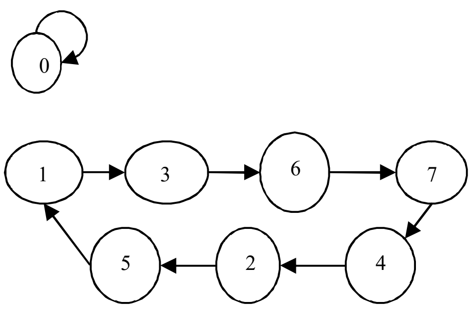


Fig. 2.1.4 (a).

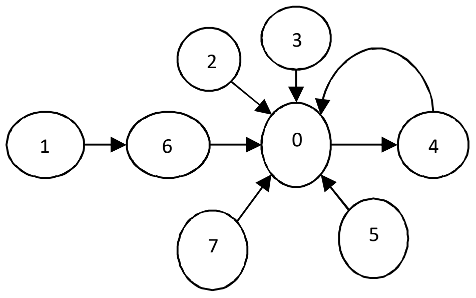


Fig. 2.1.4 (b).

Fig. 2.
Fig. 2.1.4 (a). Transition diagram of group CA for <90, 90, 150>
Fig. 2.1.4 (b). Transition diagram of non-group CA for <1, 2, 10>

Total 2^3 (8) possible patterns are possible for three cell three-neighborhood CA configuration as described in Fig. 2.1.3. Next state function of the ith cell of CA is expressed as a truth table (refer Table 2.1.1). Decimal equivalent of the eight outputs is called 'Rule' R_i [6, 13- 17]. In any three-neighborhood CA (refer Fig. 1), total [2^{2^3}] (256) rules are possible in space. Generally, these 256 rules are referred with their Wolfram code, which gives each rule a distinct number from 0 to 255. For example, rules ‘247’, ‘135’ and ‘100’ are mentioned in Table 2.1.1. The first row of the table enlists the possible 2^3 (8) combinations of the present states of ‘(i – 1)th’, ‘ith’ and ‘(i + 1)th’ cells at time ‘t’.

Table 2.1.1. Truth table for rule ‘150’, ‘170’ and ‘204’

Present State (RMT)	**111 (7)**	**110 (6)**	**101 (5)**	**100 (4)**	**011 (3)**	**010 (2)**	**001 (1)**	**000 (0)**	**Rule (Decimal Equivalent)**
(i) Next State	1	0	0	1	0	1	1	0	150
(i) Next State	1	0	1	0	1	0	1	0	170
(i) Next State	1	1	0	0	1	1	0	0	204

The set of rules $R = <R_1, R_2, \dots, R_n>$ in charge for the arrangement of the cells of a CA is called the rule vector. If identical rule is followed in all the CA cells, then the CA is supposed to be a uniform CA; otherwise it is a non-uniform/hybrid CA. If the next-state logic is employed with only XOR then it is called a linear rule and if it is employed with XNOR logic, then it is said complemented linear rule. CA with a combination of XOR and XNOR rules is called additive CA (ACA) and associated rules are defined as additive rules. CA with a combination of additive rule(s) and non-linear rule(s) are referred to as non-linear CA. If all the states of a CA in its transition diagram are in some cycles are referred to as group CA [2].

2.1.2. Application fields of CA

Potential applications of CA are found in:

- VLSI circuit testing,
- Pattern classification,
- Bio-informatics,
- Image processing,
- Mobile computing,
- Distributed computing,
- Cryptography and authentication,
- Search Engine Optimization (SEO),
- Ontology,
- Information processing and retrieval,

- Modelling of Artificial Life,
- Synthesis of Artificial Music, etc.

2.1.3. Summary

CA preliminary concepts are discussed. Potential applications of CA are reported.

2.2. Randomness

2.2.1. Random numbers

Random numbers are defined as homogeneously distributed values over a well specified interval. Prediction for the next values is unfeasible for a random sequence [48, 49]. Random patterns are achieved based on the following recursive PRNG Equation 2.2.1 [48].

$$X_{n+1}=P_1X_n+P_2 \text{ (Mod N)} \qquad (2.2.1)$$

where, 'P_1' and 'P_2' are two prime numbers;

'N' is range of random numbers;

'X_n' is calculated recursively using the value of 'X_0' as base value;

'X_0' is termed as seed and it is also a prime number;

2.2.1.1. Pseudo-random number

If 'X_0' (seed) (refer Equation 2.2.1) is same all time, then it produces pseudo-random number.

Pseudo-random numbers are produced using approaches such as, recursive algorithm based computer program, Monte-Carlo (M-C) number generator, Linear Feedback Shift register (LFSR) based random number generator or, CA based random number generator.

2.2.1.2. True-random number

Non-deterministic method is primarily required in 'seed' selection for generation of true-random numbers (TRNs). 'Seed' is fetched from physical procedures such as radioactive decay, photon emissions or atmospheric noise [48].

Rest of the section is organized as follows: comparison of PRNG and TRNG is described in Sub-section 2.2.2; finally, summary is reported in Sub-section 2.2.3.

2.2.2. Comparison of PRNGs and TRNGs

Comparative data among PRNG and TRNG as discussed in [48] is reported in Table 2.2.1.

Table 2.2.1. RNG Performance

Characteristics	**PRNGs**	**TRNGs**
Efficiency	Excellent	Poor
Determinism	Deterministic	Nondeterministic
Periodicity	Periodic	Aperiodic

2.2.3. Summary

PRNGs are better choice over TRNGs for real life applications. PRNG generated random sequences are efficient, reproducible, periodic and thus easy to implement in physical systems.

2.3. MaxCA - discussion

2.3.1. Mathematical foundation

An n-cell MaxCA is characterized by presence of a cycle of length (2^n-1). Randomness is found in generated maximum length pattern of MaxCA. Characteristics polynomial for MaxCA is primitive [6, 49].

Consider, CA size of ‘n’;

$$\text{Then, } 2^n = (2^n - 1) + 2^0 \qquad (2.3.1)$$

where, ‘(2^n -1)’ is referred to as ‘MaxCA’ cycle and ‘2^0’ is referred to as single length cycle;

MaxCA pattern is shown in Fig. 2.3.1.

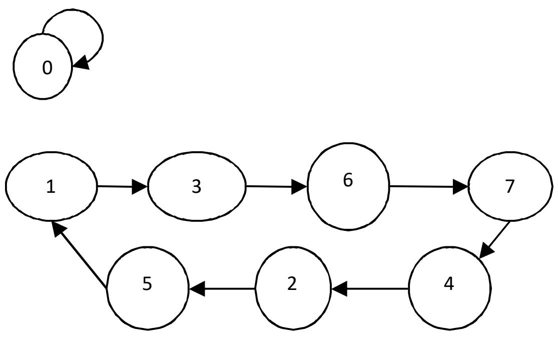


Fig. 2.3.1. MaxCA pattern for <90, 90, 150>

Rest of the section is organized as follows: CA rule combinations for MaxCA generation are presented in Sub-section 2.3.2; degree of randomness for MaxCA pattern is reported in Sub-section 2.3.3; finally, summary is in Sub-section 2.3.4.

2.3.2. CA rule combinations for MaxCA

Rule “90” and “150” in a unique combination are responsible for generation of MaxCA [2, 6, 50]. MaxCA length over different combinations of rule “90” and “150” are presented in Table 2.3.1. Rule “90” is referred to as 0 and rule “150” is referred to as 1 in Table 2.3.1 [2].

Table 2.3.1. Typical MaxCA configurations [2]

Number of cells	MaxCA Rule Vector	MaxCA Cycle Length
3	001	7
4	0101	15
5	11001	31
6	010101	63
7	1101010	127
8	11010101	255
9	110010101	511
10	0101010101	1,023
11	11010101010	2,047
12	010101010101	4,095
13	1100101010100	8,191
14	01111101111110	16,383
15	100100010100001	32,767
16	1101010101010101	65,535
17	01111101111110011	131,071
18	010101010101010101	262,143
19	0110100110110001001	524,287
20	11110011101101111111	1,048,575
21	011110011000001111011	2,097,151
22	0101010101010101010101	4,194,303
23	11010111001110100011010	8,388,607
24	111111010010110101010110	16,777,215
25	1011110101010100111100100	33,554,432
26	01011010110100010111011000	67,108,863
27	000011111000001100100001101	134,217,727
28	0101010101010101010101010101	268,435,455
29	10101001010111001010001000011	2^{29}-1
30	111010001001101100101000111101	2^{30}-1
31	0100110010101101111101110011000	2^{31}-1

A unique rule combination is available for MaxCA generation for each fixed length CA size.

2.3.3. Degree of randomness found in MaxCA generated patterns

High degree of randomness is reported for MaxCA generated patterns. Maximum degree of randomness is found in the generated maximum length cycle containing non-zero states only. Diehard results for MaxCA are followed in Table 2.3.2 [2, 6, 17, 18, 51].

Table 2.3.2. Degree of randomness for MaxCA patterns in Diehard tests

Serial number	CA cell size	Number of passes in Diehard tests
1.	23	10
2.	63	13
3.	64	14

2.3.4. Summary

Rule vector for generation of MaxCA for different cell sizes are reported in Table 2.3.1 and results in DieHard Tests for MaxCA generated patterns are reported in Table 2.3.2. No generalized method has been found to determine rule vector for generation of MaxCA patterns.

CHAPTER 3

COST OPTIMIZED CA PRNG

3.1. Cost optimized design technique for pseudo-random number generator with ELCA

3.1.1. Introduction

Random numbers play a fundamental role in the field of research work varying from computer science, mathematics or statistics to cutting edge VLSI Circuit testing. Random numbers [49, 52-54] are also used in cryptographic key generation and game playing. Mathematicians describe that this random numbers happen in a sequence where the values are homogeneously distributed over a well-defined interval, and it is unfeasible to predict the next values based on its past or present ones. The most common way to generate pseudo-random number (using structured programming language) is to use a combination of"randomize" and "rand" functions. Random patterns are achieved based on the following recursive PRNG Equation 3.1.1.1 [58].

$$X_{n+1} = P_1 X_n + P_2 (modN) \qquad (3.1.1.1)$$

where 'P_1', 'P_2' are prime numbers; 'N' is the range for random numbers; 'X_n' is calculated recursively using the base value 'X_0'; 'X_0' is a prime number and referred to as 'seed'; if X_0 (seed) is same all time or, selected in a deterministic way, then pseudo-random number is produced [58]. Random numbers over a specified boundary are essentially normalized with some distributions. Power-law distribution for random number is described in Equation 3.1.1.2 [58].

$$P(x) = CX^n \text{ for } X \in [x_0, x_1] \qquad (3.1.1.2)$$

where 'P(x)' is power-law distribution and 'C' is a constant;

The quality of randomness generated by a random number generator is needed to be verified. The diehard tests are a battery of statistical tests for measuring the quality of a random number generator. This statistical test suit was developed by George Marsaglia over several years and first published in 1995 on a CD-ROM [55].

CA based random pattern generation have been in focus of researchers' due to its low cost physical implementation capability, and high degree of randomness [1, 2, 17, 18, 51, 56, 57, 69, 70]. It is important to remeber that PPS refers to the noncomputability state for a digital circuit caused by a particular input pattern. Thus, PPS exclusion is important towards random test pattern generation. PPS exclusion from randomness generating patterns were described in [51, 56, 57]. In our studies, we have not found a simple, cost effective, and easy PPS exclusion feature in CA based PRNG. Hence a simple and cost-effective ELCA based PRNG has been introduced, which may be advantageous towards several cost-effective daily life applications.

Rest of the section is organized as follows: proposed work is described in sub-section 3.1.2; experimental results and analysis are discussed in Sub-section 3.1.3; finally, summary is in Sub-section 3.1.4.

3.1.2. Proposed approach

A cost effective and simple method targeting flexible exclusion of prohibited pattern set (PPS) is proposed in CA for generation of random numbers. Prohibited pattern is referred to as a bit configuration found in a digital circuit for noncomputability state. The cost efficiency refers to the space, time, searching of PPS and design complexity of an algorithm. A one dimensional Equal Length Cellular Automata is proposed over the existing Maximum Length Cellular Automata for random pattern generation. The decomposition of the larger cycle into more relevant sub-cycles is proposed where the concerning complexities cost can be reduced. The proposed PRNG system is described in Fig. 3.1.1.

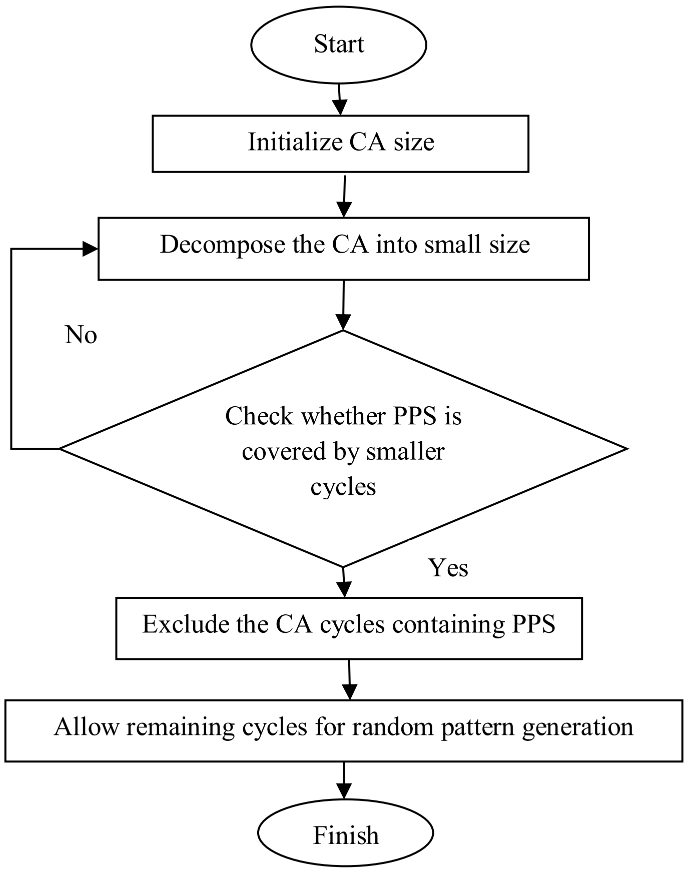

Start
Initialize CA size
Decompose the CA into small size
No
Check whether PPS is covered by smaller cycles
Yes
Exclude the CA cycles containing PPS
Allow remaining cycles for random pattern generation
Finish

Fig. 3.1.1. Flowchart of proposed CA PRNG system

A new mathematical approach is proposed to obtain randomization in low cost with respect to various complexities and hardware implementation.

Several equal length sub-cycles are considered in proposed approach instead of one cycle of maximum length cycle. The sub-cycles together are capable to generate all the states of the n-cell CA. The following Algorithm 3.1.1 is used for the generation of randomness in proposed approach.

Algorithm 3.1.1. Cycle_Decomposition

Input: *CA size (n), PPS Set*

Output: *m-length cycles excluding PPS*

```
Step 1: Start
Step 2: Initialize the number of n-cell CA to generate random patterns using n-cell CA
Step 3: Decompose the cell number (n) into two equal numbers (m) such that n=2*m
Step 4: Check each PPS whether it belongs to a single smaller cycle CA
Step 5: Repeat Step 3 and Step 4 until each PPS belongs to separate smaller cycles
Step 6: Allow m-length cycles of n-cell CA after excluding all the PPS containing cycles
Step 7: Stop
```

The primary concern in proposed approach is to exclude the PPS. The occurrence of every prohibited pattern is ensured in some of the smaller sub-cycles. Remaining prohibited patterns free cycles may be allowed to generate random patterns.

Proposed methodology implies a better cost effective approach. The proposed methodology simplifies the design complexity and empowers the searching complexity. The terminology design complexity refers to the implementation procedure for generation of random pattern and empowering searching complexity means the zero overhead for keeping track for PPS for random pattern generation. In comparison with an n-cell maximum length CA, more number of smaller cycles instead of one MaxCA are available.

Assume, for an n-length CA, the total number of states is '2^n'. By Equation 1.1, we have,

$2^n = 2^{n-1}+2^{n-1}$

= $2^1*(2^{n-1})$ (two number of equal length cycles)

= $2^2*(2^{n-2})$ (four number of equal length cycles)

= $2^m * (2^{n-m})$ (2^m number of equal length cycles) for $n \geq 1$ and m=1, 2, 3… (n-1).

Thus ‘m’ is always less than ‘n’.

PPS containing part is excluded from the MaxCA cycle [51, 56, 57]. PPS is completely removed in proposed approach. Random pattern generation in this scenario is followed in Fig. 3.1.2. Fig. 3.1.2(a) shows one maximum length cycle with prohibited patterns and Fig. 3.1.2(b) shows several equal length smaller cycles where some of the cycles contain prohibited set. The PPS in Fig. 3.1.2 is denoted as {PS_0, PS_1……PS_9}.

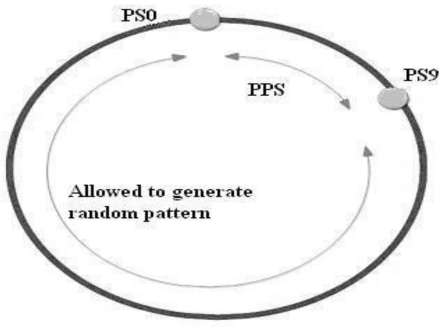


Fig. 3.1.2(a).

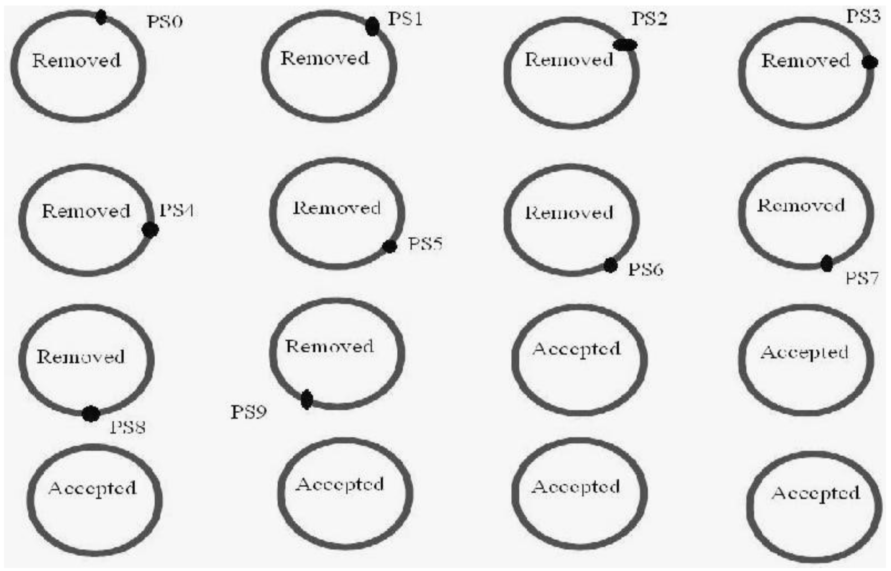


Fig. 3.1.2(b).
Fig. 3.1.2.
Fig. 3.1.2(a). Typical PPS exclusion in maximum length CA cycle
Fig. 3.1.2(b). Typical PPS exclusion in proposed equal length CA of smaller cycle size

Assume that n-numbers of prohibited patterns are present in the maximum length cycle. Let the prohibited patterns are $\{PS_0,\ldots,PS_n\}$. The minimum length of arc (Arc_{min}) between the prohibited pattern PS_1 and PS_n should be measured in the scenario of MaxCA so that remaining cycle arc i.e. effective arc (E_{arc}) may be utilized for random number generation. E_{arc} is typically free from PPS. This scenario is shown in Fig. 3.1.3. Fig. 3.1.2(a) and Fig. 3.1.3 are inspired from [57].

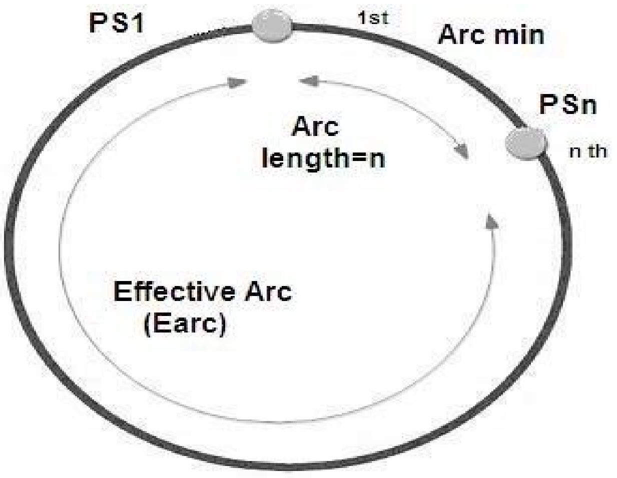


Fig. 3.1.3. Typical cycle structure of an n-cell maximum length CA

Arc_{min}- *The minimum length of arc (Arc_{min}) is the minimum distance between the first and last prohibited pattern in an n-cell maximum length CA cycle.*

E_{arc}- *The effective arc (E_{arc}) is the remaining arc length of an n-cell maximum length CA cycle which excludes Arc_{min} from the corresponding CA cycle of states and it is responsible for generating pseudo-random patterns of integers.*

3.1.3. Experimental observations & result analysis

PPS containing arc is excluded from the maximum length cycle. On the other hand, the PPS containing cycles are totally removed for generation of random sequences in proposed approach. Concerns to calculate E_{arc} and Arc_{min} are totally absent in proposed methodology. The following Table 3.1.1 compares the procedures of maximum length cycle and proposed equal length cycle based approaches.

Table 3.1.1. Comparison of fault coverage procedures

	Maximum length CA	**Equal length CA**
^{E}arc	Computation policy required	N/A
Arc_{min}	Computation policy required	N/A

The p-value analysis in Diehard test helps to decide whether the test data set passes or fails the diehard test. Diehard test returns the p-value, which should be uniform over [0, 1) if the input file contains truly independent random bits. The p-values are obtained by p=F(x), where F is the assumed distribution of the sample random variable 'x', which is often normal. The value p<0.025 or p>0.975 means the RNG has "failed the test at the 0.05 level" [55]. Comparison results for proposed CA-PRNG with reference to MaxCA PRNG [51, 56, 57] are presented in Table 3.1.2.

Table 3.1.2. Performance result through Diehard for different CA random number generators

Serial Number	**Name of the test**	**MaxCA**		**Proposed approach**	
		n=23	*n=64*	*n=23*	*n=64*
1	Birthday Spacings	Pass	Pass	Pass	Pass
2	Overlapping Permutations	Pass	Pass	Pass	Pass
3	Ranks of 31x31 and 32x32 matrices	Pass	Pass	Pass	Pass
4	Ranks of 6x8 Matrices	Pass	Pass	Pass	Pass
5	The Bitstream Test	Fail	Fail	Fail	Fail
6	Monkey Tests OPSO,OQSO,DNA	Fail	Pass	Fail	Pass
7	Count the 1`s in a Stream of Bytes	Pass	Pass	Pass	Pass
8	Count the 1`s in Speci_c Bytes	Fail	Pass	Fail	Pass
9	Parking Lot Test	Pass	Pass	Pass	Pass
10	Minimum Distance Test	Pass	Pass	Pass	Pass
11	The 3DSpheres Test	Pass	Pass	Pass	Pass
12	The Sqeeze Test	Fail	Pass	Fail	Pass
13	Overlapping Sums Test	Fail	Pass	Fail	Pass
14	Runs Test	Pass	Pass	Pass	Pass
15	The Craps Test	Pass	Pass	Pass	Pass
Total Number of Diehard Test Passes		10	14	10	14

Competitive degree of randomness is found in Table 3.1.2. Several complexities for the two CA based methodologies are in Table 3.1.3.

Table 3.1.3. Complexity comparison between MaxCA and proposed methodology

Name of the Complexity	**Comparison Result**
Space	Same
Time	Slightly Improved
Design	Improved
Searching for PPS	Improved

Space complexity for both approaches is same as total length of an n-cell CA are same. Some changes in other complexities are found in Table 3.1.3. Other complexities are improved in case for proposed methodology. The proposed methodology is allowed only to generate random patterns from smaller cycles that exclude PPS. The PPS exclusion feature from the main cycle improves the design complexity and search of PPS process.

3.1.4. SUMMARY

Results based on Table 3.1.1, Table 3.1.2 and Table 3.1.3 conclude that proposed methodology is capable to produce pseudo-random sequences at low cost.

3.2. CA PRNG in Monte-Carlo simulation

3.2.1. Introduction

Distributed computing is a reliable solution in modern days computing requirement. Monte-Carlo simulator is highly effective in assessing the reliability of a distributed system. Monte-Carlo (M-C) random number generator (RNG) found with Monte-Carlo simulator is alternatively used for generation of pseudo-random numbers [58, 59]. The term 'Monte-Carlo' was introduced by Von Neumann and Ulam during World War II. M-C method was applied to problems related to the atomic bomb. The mean values of stochastic variables are expressed as integral of variables in M-C method as described in Equation 3.2.1 [58, 59].

$$I = \int_D h(x) f(x) dx \qquad (3.2.1)$$

where 'D' is high dimensional domain with coordinates 'x' and 'f(x)' is a non-negative function.

Equation 3.2.2 [58, 59] is satisfied by 'f(x)'.

$$\int_D f(x) dx = 1 \qquad (3.2.2)$$

Past research explores that CA based PRNG may be used in Monte-Carlo simulation [12-14]. Thus, CA based PRNG may be used in reliability assessment.

Rest of the section is organized as follows: proposed work is described in Sub-section 3.2.2; experimental results and analysis are in Sub-section 3.2.3; finally, summary is in Sub-section 3.2.4.

3.2.2. Proposed approach

Monte-Carlo simulation is used to evaluate reliability of a complex and distributed system. Monte-Carlo test cases are produced using pseudo-random numbers. A default recursive PRNG is available along with the Monte- Carlo simulator. CA-PRNGs are considered to generate pseudo-random numbers in Monte Carlo simulator. Flowchart of the proposed system is in Fig. 3.2.1.

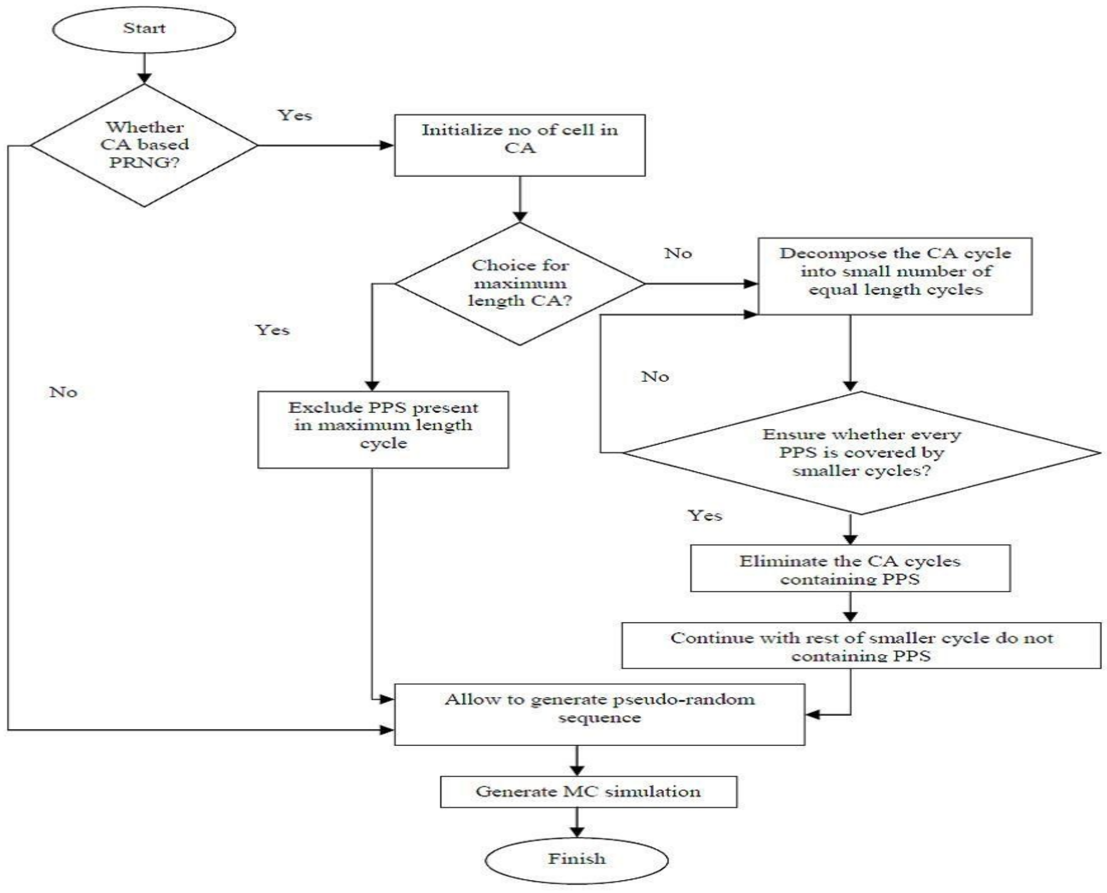

Start
Whether CA based PRNG?
Yes
No
Initialize no of cell in CA
Choice for maximum length CA?
No
Yes
Decompose the CA cycle into small number of equal length cycles
Exclude PPS present in maximum length cycle
Ensure whether every PPS is covered by smaller cycles?
No
Yes
Eliminate the CA cycles containing PPS
Continue with rest of smaller cycle do not containing PPS
Allow to generate pseudo-random sequence
Generate MC simulation
Finish

Fig. 3.2.1. Flowchart of the proposed reliability assessment system

A novel low cost approach is introduced in the flowchart of the proposed system to achieve randomization in the resulting Monte-Carlo simulation. Algorithm 3.2.1 is used in proposed approach. Algorithm 3.2.1 is based on Algorithm 3.1.1.

Algorithm 3.2.1. Monte-Carlo_Simulation_Generation

Input: *Choice for PRNG, CA size (n), PPS Set*

Output: *Pseudo-random number based Monte-Carlo Simulation*

Step 1: Start

Step 2: If selected PRNG is based on CA then follow Step 3 else follow Step 10

Step 3: Initialize the number of n-cell CA for generating random patterns using n-cell CA

Step 4: If PRNG is based on maximum length CA then follow Step 9 else follow Step 5

*Step 5: Decompose the cell number (n) into two equal numbers m) such that n=2*m*

Step 6: Verify for each PPS whether the PPS belongs to a single smaller cycle CA or not

Step 7: Repeat Step 3 and Step 4 until each of the PPS belongs to separate smaller cycles

Step 8: Permit m-length cycles of n-cell CA after excluding all the PPS containing cycles

Step 9: Exclude PPS present in maximum length cycle and follow Step 11

Step 10: Select Monte-Carlo default random number generator

Step 11: Generate random number sequence

Step 12: Generate pseudo-random number based Monte-Carlo Simulation

Step 13: Stop

3.2.3. Experimental observations & result analysis

Degree of randomness for the two CA-RNGs and default Monte-Carlo simulator RNG are shown in Fig. 3.2.2.

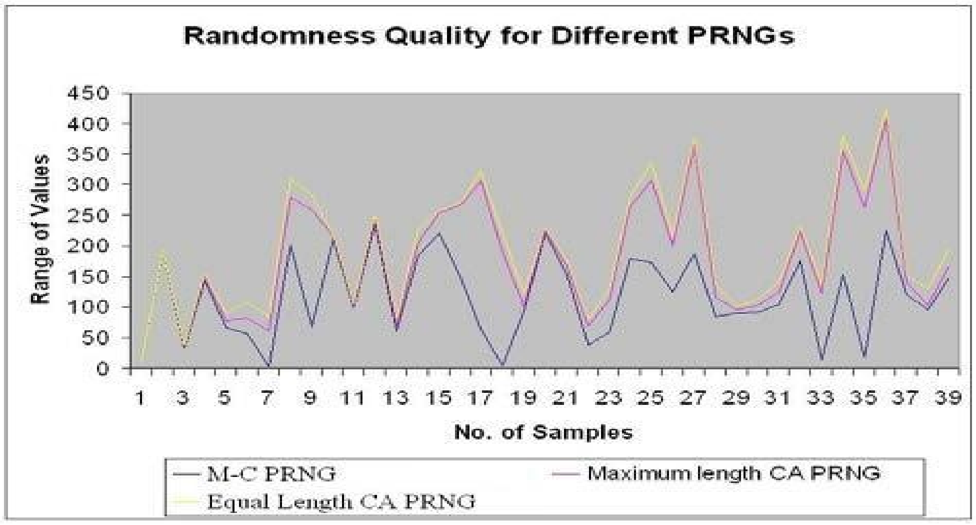


Fig. 3.2.2(a) Randomness quality for different PRNGs

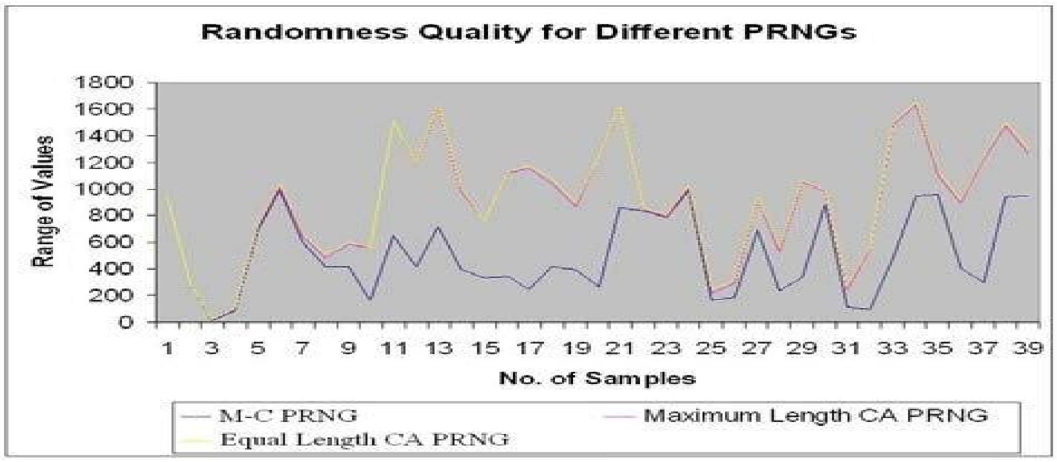


Fig. 3.2.2(b) Randomness quality for different PRNGs

Fig. 3.2 2. Monte-carlo simulation result using different PRNGs

Randomness for a part of pattern from the PRNGs used in Monte-Carlo simulation is shown in Fig. 3.2.2. CA length 8 is applied for Fig. 3.2.2(a), and in CA length 10 is applied in Fig. 3.2.2(b). A better quality of randomness is found for CA-PRNGs compared to Monte-Carlo in Fig. 3.2.2(a) and Fig. 3.2.2(b). Competitive randomness is found for both CA-PRNGs in Fig. 3.2.2.

3.2.4. Summary

Competitive randomness is found in Fig. 3.2.2 for both CA-PRNGs. Associated costs for proposed equal length CA PRNG are much lower compared to MaxCA (refer Sub-section 3.1). Hence proposed CA-PRNG is capable of low cost test case generation in Monte-Carlo simulation.

3.3. Cost effective PRNG in BIST application

3.3.1. Introduction

The advancements of Information Technology are realized in the modern age chip fabrication techniques. The size of a complete electronic system is now reduced to a scale that is harder to view with bare eyes. The impact of minimization initiated a strong necessity for testing of minimized systems, circuit boards and chip components.

Built-In Self-Test (BIST) [60-63] is typically used for testing purpose in design-for-testability (DFT) technique. A section of the circuit under test (CUT) is verified and tested to recognize the behavior of the total circuit. A typical BIST architecture is presented in Fig. 3.3.1 [60].

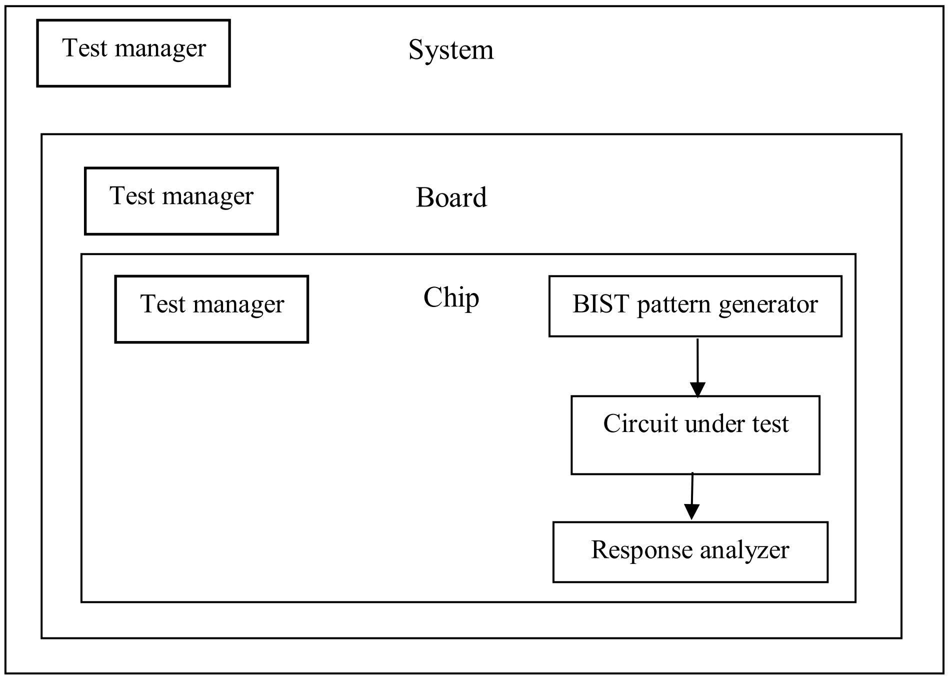


Fig. 3.3.1. Typical BIST architecture [60]

Pattern generator, response analyzer and test controller are essential to continue BIST in digital circuit. Patterns are stored in Read Only Memory (ROM). LFSR is used as a pattern generator. LFSR is also used as a response analyzer. Pseudo-random sequences are needed for pattern generation in BIST applications [60-63].

Linear Feedback Shift Register (LFSR) is a popular choice as pseudo-random pattern generation for Built-In Self-Test (BIST) application [11-13]. LFSR has some advantages as it possesses fewer XOR gates and a good internal feedback policy. Better quality of randomness and cost effective physical implementation was described for the CA based PRNG compared to the LFSR based PRNG [16]. Several CA based PRNG in BIST applications were described in [17, 18, 51, 56, 57]. In our studies, we have not found a simple, cost effective and easy PPS exclusion feature in CA based PRNG. Hence a simple and cost-effective ELCA based PRNG has been introduced, which may be advantageous towards cost-effective BIST application.

Rest of the section is organized as follows: proposed work is described in Sub-section 3.3.2; experimental results and analysis are in Sub-section 3.3.3; summary is in Sub-section 3.3.4.

3.3.2. Proposed approach

ELCA based PRNG is proposed for BIST test pattern generation. Flowchart of our proposed ELCA based BIST pattern generator is followed in Fig. 3.3.2.

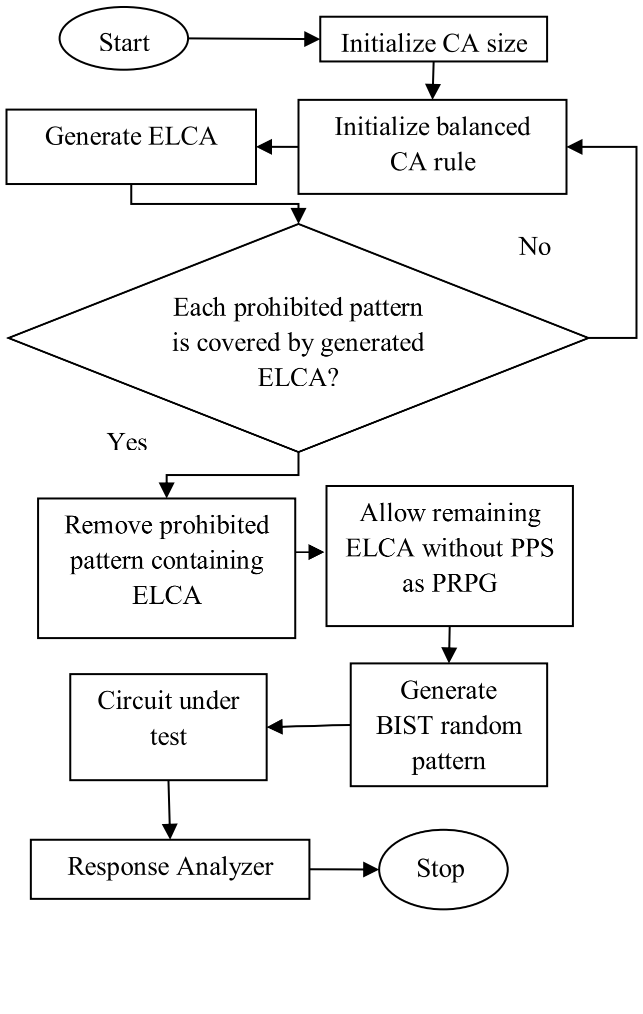

Start
Initialize CA size
Generate ELCA
Initialize balanced CA rule
No
Each prohibited pattern is covered by generated ELCA?
Yes
Remove prohibited pattern containing ELCA
Allow remaining ELCA without PPS as PRPG
Circuit under test
Generate BIST random pattern
Response Analyzer
Stop

Fig. 3.3.2. Flowchart of proposed BIST pattern generation system

Flowchart of Fig. 3.3.2 is implemented for ELCA based BIST pattern generation using Algorithm 3.1.1 of Sub-section 3.1.2. Cost effective PRNG is obtained from Algorithm 3.1.1. PPS exclusion policy in proposed method is same as described in Sub-section 3.1.2.

3.3.3. Experimental observations & result analysis

Fault coverages for several benchmark circuits with ELCA, MaxCA, and LFSR BIST pattern generator are presented in Table 3.3.1. Fault coverages for several "ISCAS 85" and "ISCAS 89" benchmark circuits in "BISTAD" [64] are presented in Table 3.3.1.

Table 3.3.1. Fault coverage for different pattern generators

Circuit name	Feed back	Seed	Coverage by LFSR	Coverage by MaxCA	Coverage by ELCA
s386	2	15	93.17 %	96.67%	100.00%
s298	2	15	93.30%	100.00%	100.00%
s1488	2	15	98.92%	97.72%	98.68%
s1494	2	15	98.11%	96.87%	97.87%
s208_1	2	15	95.18%	96.93%	99.56%
s27	2	15	100.00%	100.00%	100.00%
c17	2	15	100.00%	100.00%	100.00%

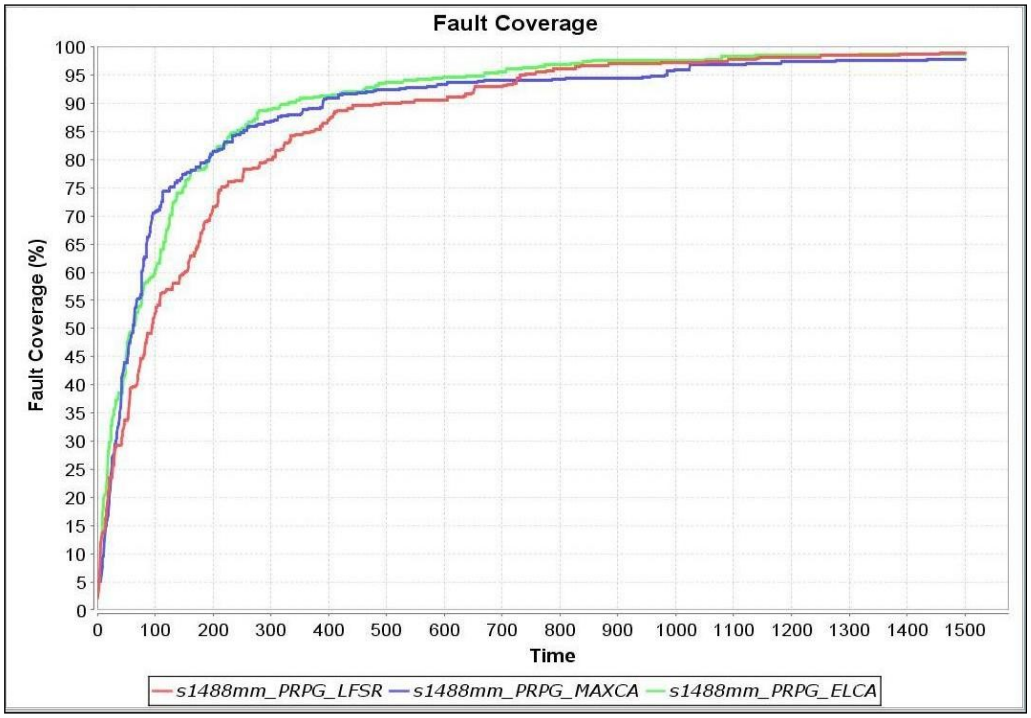


Fig. 3.3.3. Screenshot for fault coverage in "ISCAS89 s1488" benchmark circuit

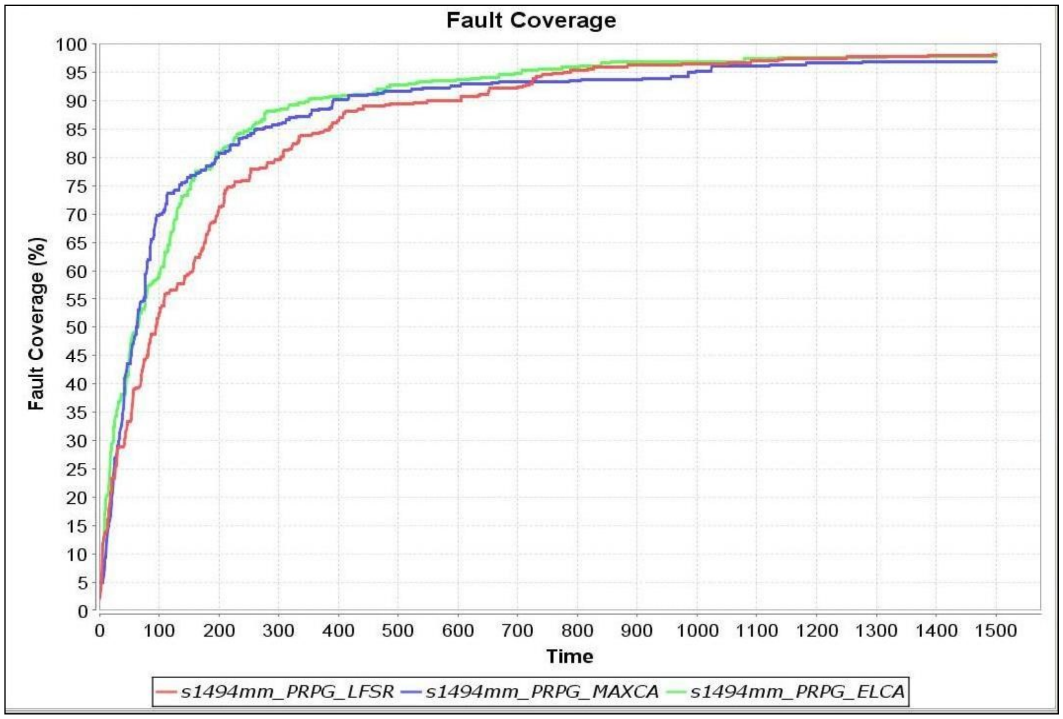


Fig. 3.3.4. Screenshot for fault coverage in "ISCAS89 s1494" benchmark circuit

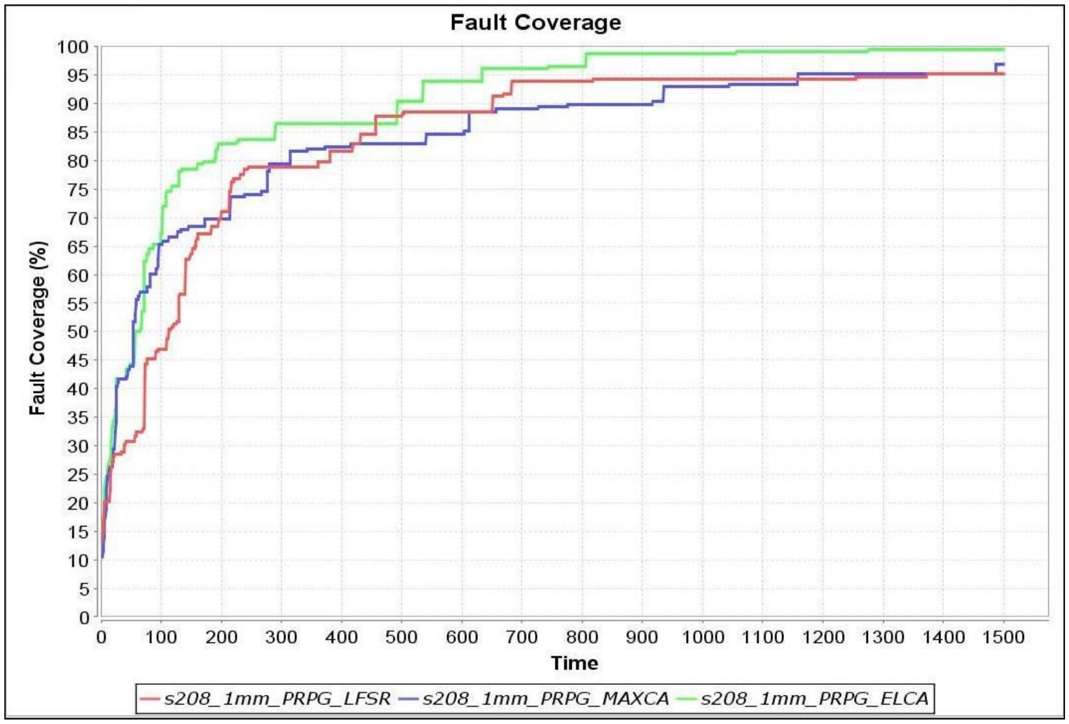


Fig. 3.3.5. Screenshot for fault coverage in "ISCAS89 s208" benchmark circuit

Competitive results in fault coverages of several "ISCAS 85" and "ISCAS 89" circuits are shown in Fig. 3.3.3, Fig. 3.3.4 and Fig. 3.3.5.

3.3.4. Summary

Competitive results for ELCA-PRNG in several "ISCAS 85" and "ISCAS 89" benchmark circuits are found in Table 3.3.1. Cost-effectiveness of ELCA over MaxCA is described in Section 3.1. Hence ELCA is a cost-effective choice for BIST application.

3.4. Design of a CA based System-Under-Test

3.4.1. Introduction

Distributed computing is the present trend for efficient resource utilization and processing in networked architecture. An inherent requirement to enhance the distributed computing is satisfied with the design of a fault tolerance in distributed system. Several aspects are focused for the enhancements of distributed computing. Researches were carried out on virtualization to enhance fault tolerance in distributed systems [25-27, 65-67]. Testing algorithms for distributed systems often require fault injection (FI) to assess reliability in distributed system. Faults are considered as elements of the applicable input data for fault-tolerant distributed. An infrastructure was presented by I. Hsu et al. [27] to support the analysis of the behavior of distributed system-under-test (SUT). Software-implemented fault injector (FI) and virtualization were combined in [25] for an automated validation and analysis of distributed SUT. Hybrid fault injection for distributed SUT was introduced by C. Trödhandl et al. [26]. Validation of fault tolerant distributed system requires a flexible infrastructure for the execution of injected faults. Three key requirements for the infrastructures as identified in [27] are as follows.

i) Boundary conditions across multiple system components (nodes) should be tested at randomized test values;

ii) initial system state for each test must be at "error free" state;

iii) a log file should be available for off-line analysis of fault injections, system responses, and resource uses.

FIs are categorized into Simulation based FI [65], hardware FI [66] and software FI [67]. Timing behavior of the complete system-under-test (SUT) is adversely affected by injection of faults. Hardware FI is preferred over software FI. Hardware FI is primarily used on the chip-level [65-68]. BIST may be used as a choice for testing purpose in SUT. Thus, a simple ELCA based BIST architecture is proposed towards SUT in distributed computing, which may be advantageous as a cost-effective solution in distributed computing.

ELCA based infrastructure in SUT is projected in this section. Cost optimized performance of ELCA in BIST applications in Sub-section 3.3.1 initiates the potential application of ELCA in SUT.

Rest of the section is organized as follows: proposed work has been described in Sub-section 3.4.2; experimental results and analysis have been reported in Sub-section 3.4.3; finally, summary has been reported in Sub-section 3.4.4.

3.4.2. Proposed approach

ELCA based design for testing of hardware components in system-under-test (SUT) is proposed. Faults injected by FI in a target component of a SUT are tested with ELCA based design. Proposed ELCA based design in SUT is presented in Fig. 3.4.1.

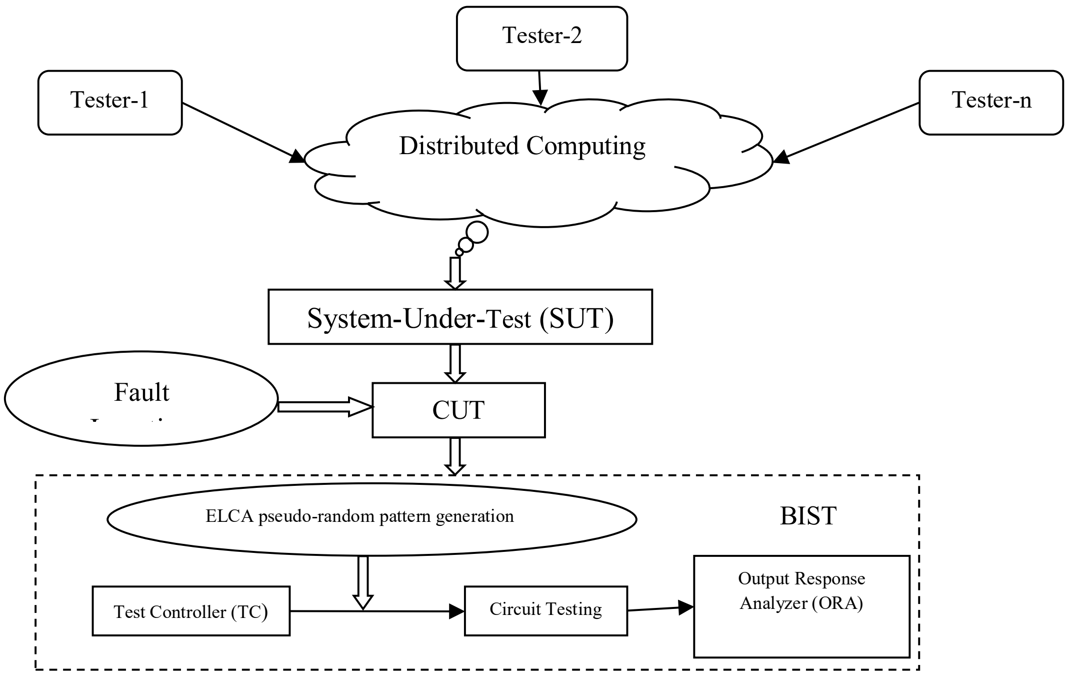


Fig. 3.4.1. Proposed SUT architecture in distributed computing environment

Proposed SUT architecture may be incorporated over existing Client-Server based FI hardware as shown in [26]. Client-Server based FI hardware is presented in Fig. 3.4.2 [26].

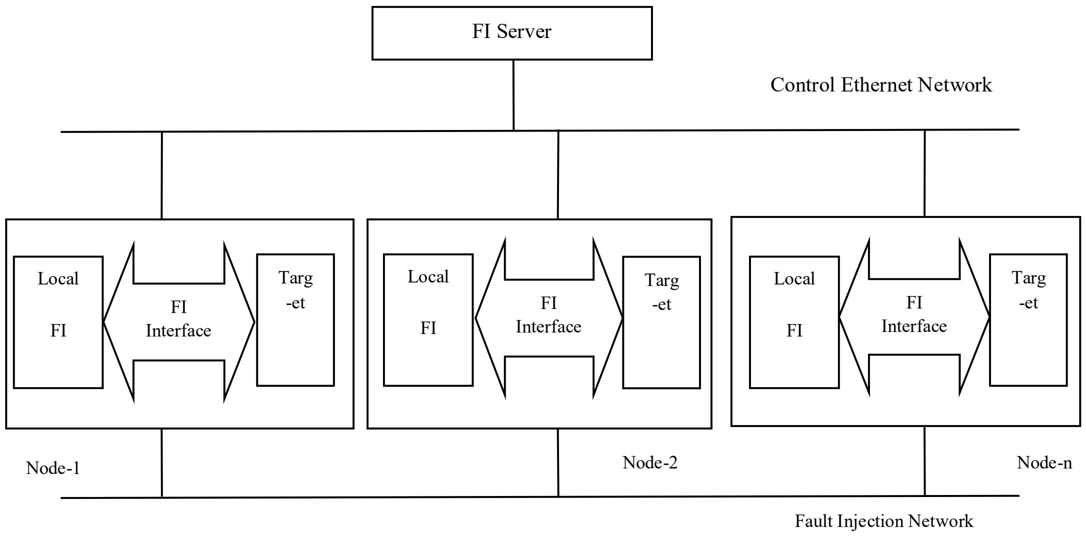


Fig. 3.4.2. Fault injection client-server architecture [26]

Flowchart of proposed SUT design is presented in Fig. 3.4.3.

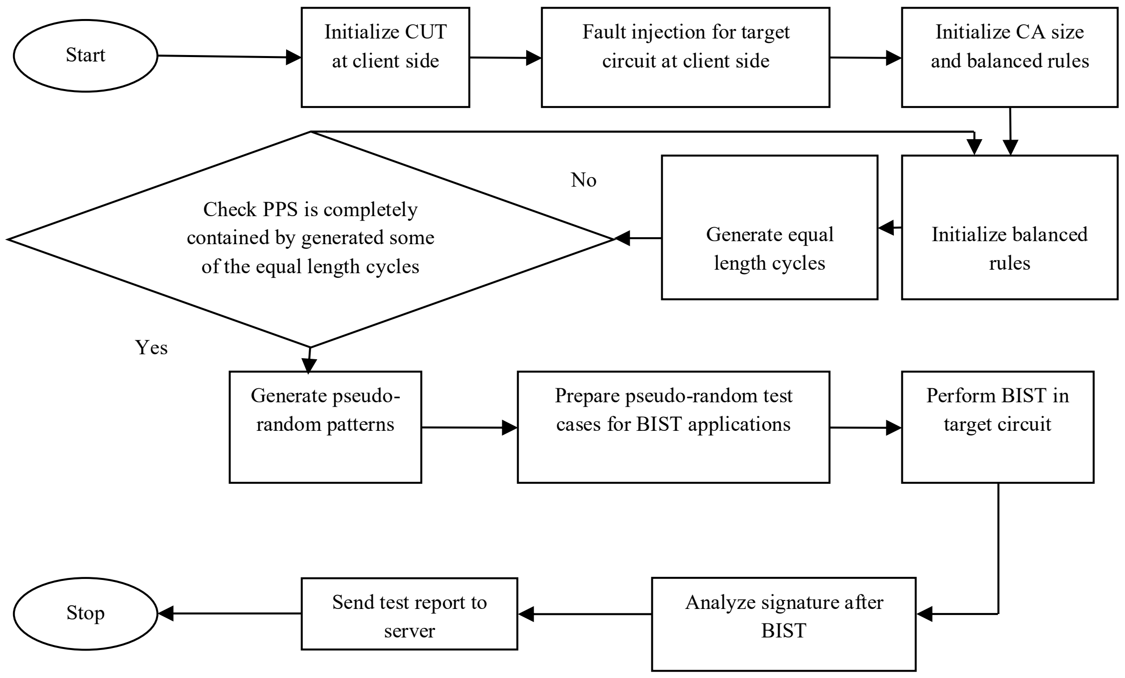

Start
Initialize CUT at client side
Fault injection for target circuit at client side
Initialize CA size and balanced rules
Initialize balanced rules
Generate equal length cycles
Check PPS is completely contained by generated some of the equal length cycles
No
Yes
Generate pseudo-random patterns
Prepare pseudo-random test cases for BIST applications
Perform BIST in target circuit
Analyze signature after BIST
Send test report to server
Stop

Fig. 3.4.3. Proposed flowchart of SUT using ELCA BIST

PPS free pseudo-random pattern generation in SUT is same as described in Sub-section 3.1.1 and Sub-section 4.1.1. Algorithm 3.4.1 is used to perform CUT in SUT.

Algorithm 3.4.1. ELCA_based_Built-In Self-Testing_ for_System-Under-Testing

Input: *SUT components*

Output: *Checked error free system components for reliable computation*

Step 1: Start

Step 2: Initialize the circuit-under-test (CUT) area to be tested by server side

Step3: Inject faults by Fault Injector (FI) in target circuit of the client side

Step 4: Initialize the Built-in self-test (BIST) at client side

Step 5: Generate PPS free pseudo-random test patterns using Algorithm 3.1.1

Step 6: Perform testing on target circuit

Step 7: Analyze the signature generated in BIST

Step 8: Report circuit status for fault inspection at server side

Step 9: Report SUT tester about component status for processing of reliable computation

Step 10: Stop

3.4.3. Experimental observations & result analysis

Several functional fault models for BIST [68] are as follows.

i) **Stuck_at_faults (SAF)**: *A cell/line is stuck to logical zero/one state. SAF is thus categorized into Stuck_at_1 and Stuck_at_0 fault.*

ii) **Transition_faults (TF)**: *It is not possible to retain previous state value since the memory value is changed once. It is similar to SAF.*

iii) **Coupling_faults (CF)**: *Coupling between two adjacent cells is focused with CF during a transition from "zero to one" or, "one to zero". Neighbouring cell is forced to change its value during transition of target cell.*

iv) **Neighborhood_pattern_sensitive_faults (NPSF)**: *Center cell of a nine-neighbourhood configuration is bound to change its value influenced by its neighbourhood.*

v) **Data_retention_faults (DF)**: *It is not possible by memory cell to retain its value over time after a memory write/read operation.*

vi) **Address_decoder_faults (AF)**: *No cell or, multiple cells are accessed simultaneously with an address or, a single cell is accessed by multiple addresses.*

Coverage of SAF is considered in proposed SUT design. “BISTAD” is used for testing the fault coverages of “t3.agm” benchmark circuits [64].

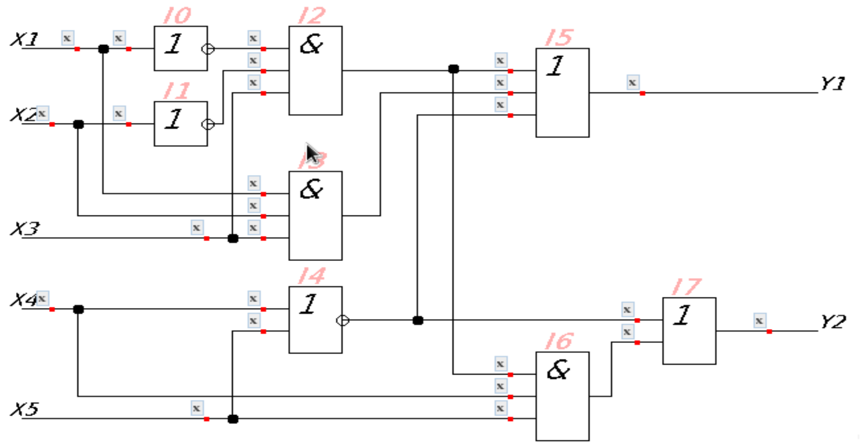
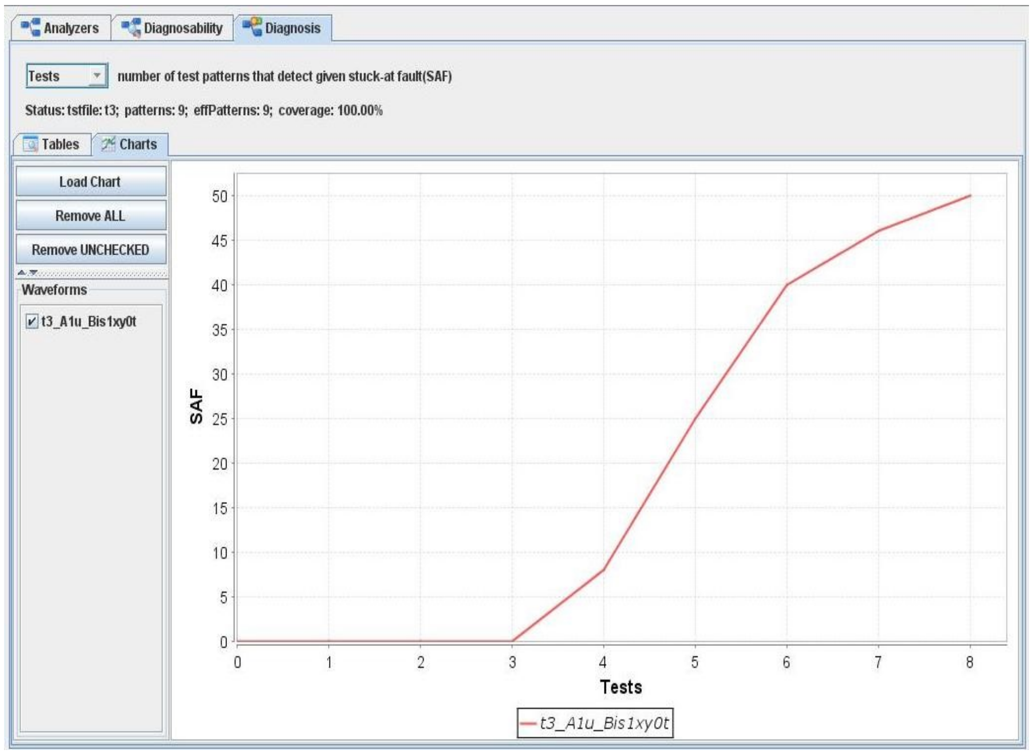


Fig. 3.4.4. Screen shot for “t3.agm” benchmark circuit

Simulation results for “t3.agm” benchmark circuit are presented in Table 3.4.1, Fig. 3.4.5 and Fig. 3.4.6. Fault Table used for “t3.agm” circuit testing in “BISTAD” [62, 63] is followed in Table 3.4.2.

Table 3.4.1. Fault table for “t3.agm”

Vectors	% (Total)	%	F A U L T T A B L E																								
10101	28.0	28.0	0	1	X	X	0	0	X	1	X	X	X	1	X	X	0	1	1	1	X	X	X	1	1	1	1
01101	38.0	28.0	1	0	X	X	0	X	0	X	1	X	1	X	X	X	0	1	1	1	X	X	X	1	1	1	1
00011	44.0	20.0	X	X	1	X	X	X	X	X	X	1	X	X	X	X	X	1	1	1	1	X	X	1	1	1	1
11111	56.0	24.0	0	0	0	X	X	X	X	X	X	X	0	0	0	X	X	X	0	X	1	X	X	1	1	1	0
11011	58.0	20.0	X	X	1	X	X	X	X	X	X	X	X	X	1	X	X	1	1	1	1	X	X	1	1	1	1
00110	78.0	34.0	1	1	0	0	1	1	1	0	0	0	X	X	X	0	X	0	X	X	X	X	1	1	1	1	0
10000	90.0	16.0	X	X	X	1	1	X	X	X	X	X	X	X	X	1	1	X	X	0	X	X	X	0	X	0	0
00111	98.0	34.0	1	1	0	0	0	1	1	0	0	0	X	X	X	X	X	0	X	X	0	0	0	X	0	0	0
00101	100.0	34.0	1	1	0	1	0	1	1	0	0	0	X	X	X	X	0	0	X	X	X	1	X	1	1	1	0
		&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&



number of test patterns that detect given stuck-at fault(SAF)

Status: tstfile: t3; patterns: 9; effPatterns: 9; coverage: 100.00%

Fig. 3.4.5. Screen shot of SAF test coverage graph for “t3.agm”

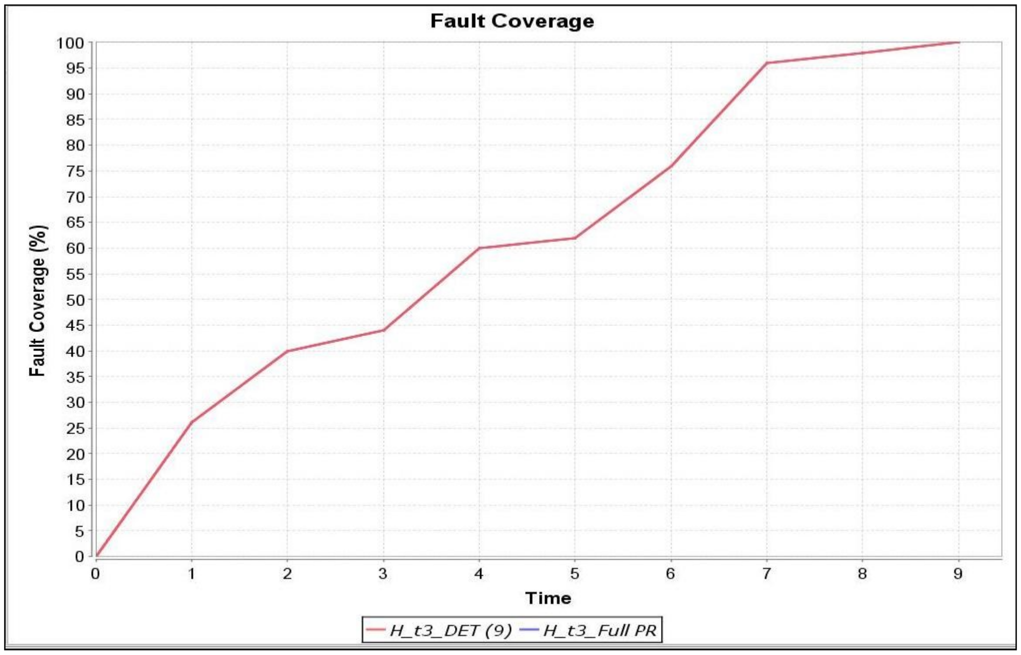


Fig. 3.4.6. Screen shot of fault coverage for “t3.agm”

Stuck-at-faults (SAF) coverage for “t3.agm” is shown in Fig. 3.4.5 and fault coverage of “t3.agm” is shown in Fig. 3.4.6.

Fault coverages for several combinational circuits obtained in "BISTAD" are presented in Table 3.4.2. Each "t.agm" combinational circuit contains 5 inputs and 2 outputs, and behaves as an independent system [64]. Fault coverage is described in Equation 3.4.1 [68].

$$fault\_coverage = \frac{no\_of\_faults\_noticed}{total\_no\_of\_faults} \tag{3.4.1}$$

Table 3.4.2. Fault coverage table for 't' benchmark circuits

Serial No.	**Model No.**	**Fault coverage**
1	t1.agm	100%
2	t2.agm	100%
3	t3.agm	100%
4	t4.agm	100%
5	t5.agm	100%
6	t6.agm	100%
7	t7.agm	100%
8	t8.agm	100%
9	t9.agm	100%
10	t10.agm	100%

High fault coverage for different "t.agm" benchmark circuits is found in Table 3.4.2.

3.4.4. Summary

High fault coverage for ELCA based PRNG is obtained for "ISCAS 85", "ISCAS 89" and "t.agm" circuits. Therefore, ELCA PRNG is an efficient test pattern generator for BIST, SUT applications in distributed computing.

CHAPTER 4

PERFORMANCE COMPARISON AMONG RNGs

4.1. Random number generators: performance comparison

4.1.1. Introduction

Random numbers [48, 49, 58] are considered in research works varying from Computer Science to Mathematics. Random numbers are homogeneously distributed over a well-defined interval and it is unfeasible to predict the next values for a random pattern. 'Seed' is used as a specification of an initial number for generation of a random pattern. RNGs are classified into several groups based on the difference in generation procedures of random numbers. Pseudo-random number (PRN) and true-random number (TRN) are most commonly used in scientific works.

Random numbers or, random-patterns [48, 49, 58] obtained with a computer program is based on recursive algorithm. Deterministic way for selection of the 'seed' makes the pattern generation procedure as 'Pseudo-random' [48]. Several efforts are found in literature to produce quality random numbers [17, 18, 48, 51, 56, 57, 69, 70]. Few popular techniques for generation of pseudo-random sequences are described briefly in the following sub-sections.

Recursive algorithm based computer program which is most frequently used as a source of pseudo-random sequence is considered in this section. Generation of pseudo-random numbers by recursive algorithm is shown in Fig. 4.1.1.

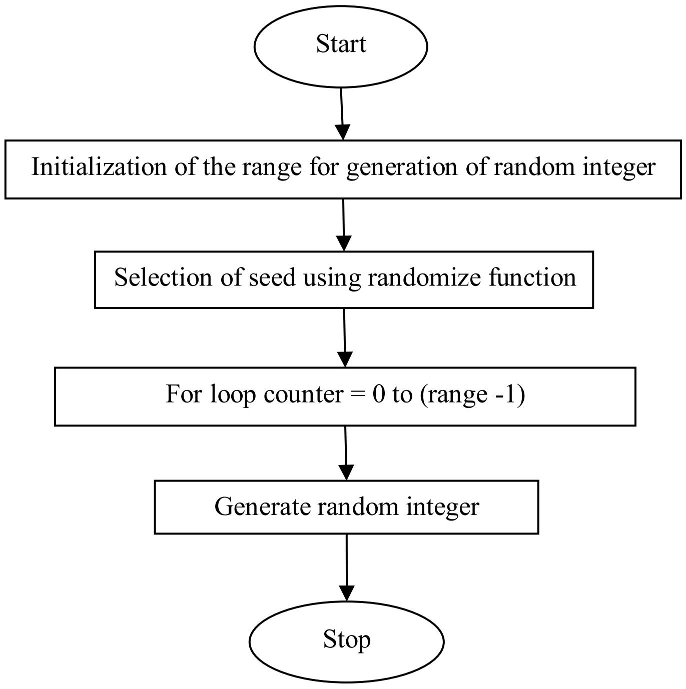

Start
Initialization of the range for generation of random integer
Selection of seed using randomize function
For loop counter = 0 to (range -1)
Generate random integer
Stop

Fig. 4.1.1. Flowchart for pseudo-random number generation by recursion

Algorithm 4.1.1 is used to prepare pseudo-random number.

Algorithm 4.1.1. Recursive_Pseudo-Random_Pattern_Generation

Input: *Upper limit for random numbers to be generated (n)*

Output: *Random pattern of integers*

Step 1: Start

Step 2: Initialize the range for which random integers to be generated

Step 3: Setting up of randomize seed

Step 4: Repeat Step 5 until required number of iteration has been achieved

Step 5: Generate random number using the random seed

Stop 6: Stop

M-C PRNG (found with Monte-Carlo Simulator) is another option to produce pseudo-random numbers within a specific boundary [13]. Pseudo-random sequence generation using CA specifically MaxCA is briefly described in Section 4.1.2.

Rest of the section is organized as follows: proposed work is described in Sub-section 4.1.2; experimental results and analysis are described in Sub-section 4.1.3; finally, summary is in Sub-section 4.1.4.

4.1.2. Proposed approach

Few popular RNGs are considered in search of a cost-efficient RNG along with flexible prohibited pattern set (PPS) exclusion process. Prohibited pattern is referred to as a bit configuration found in a digital circuit for noncomputability. Recursive Random Number Generator (RRNG), True Random Number Generator (TRNG) [48], Monte-Carlo Random Number Generator (M-C RNG), MaxCA PRNG, and ELCA PRNG are considered for this comparison.

All states of an n-cell CA may be generated as a collection of several equal length cycles. Flowchart (Fig. 3.1.1) and algorithm (Algorithm 3.1.1) presented in Section 3.1 are used in the following sub-section. An n-cell CA is divided into two or more equal sub-cycles instead of taking the full cycle. Generated sub-cycles together produce all the states of the n-cell CA.

Example 4.1.1.

Consider, CA size 5. 5-cell CA might be decomposed into some equal length smaller cycles instead of one maximum length cycle. CA size 5 in Example 4.1.1 is decomposed into 4 cycles of length 8 (Refer 4.1.2(b)); or, it is decomposed into 8 smaller cycles of length 4 (Refer Fig. 4.1.2(c)). Maximum length cycle is shown in Fig. 4.1.2(a); Fig. 4.1.2(a) is based on Null Boundary 5 cell CA having rules in specified sequence < 90, 90, 90, 90, 150 >. The synthesis of this example in Fig. 4.1.2(b) and Fig. 4.1.2(c) Equal Length CA (ELCA) are achieved for a combination of balanced CA rules [2] such as “60”, “102”, “153”, “195” for CA size n=5.

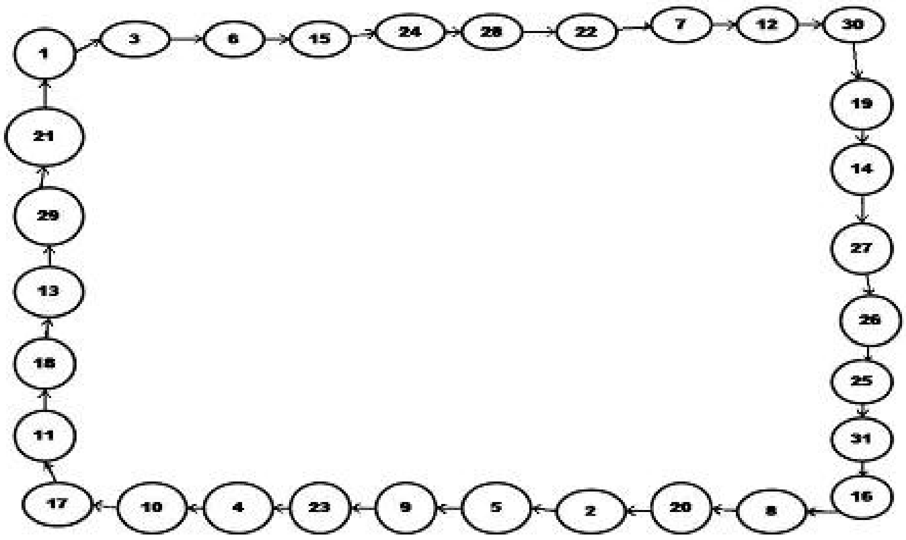


Fig. 4.1.2(a)

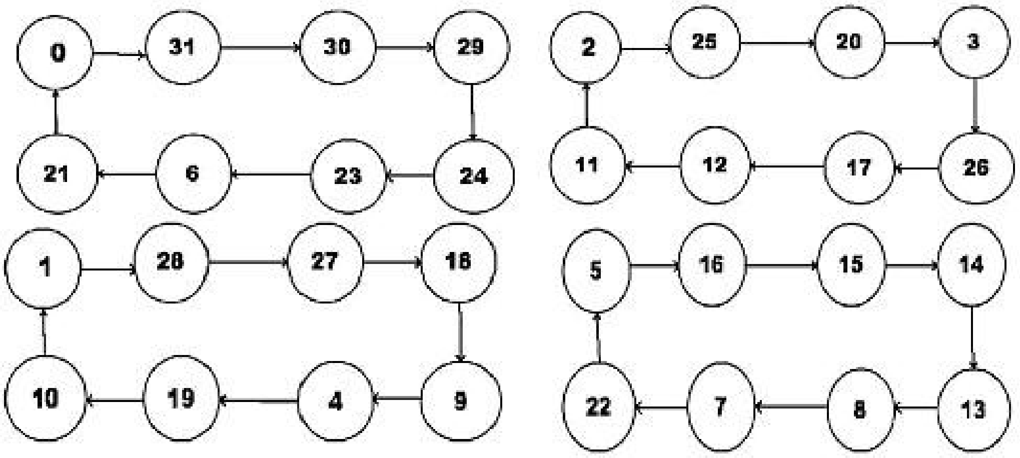


Fig. 4.1.2(b)

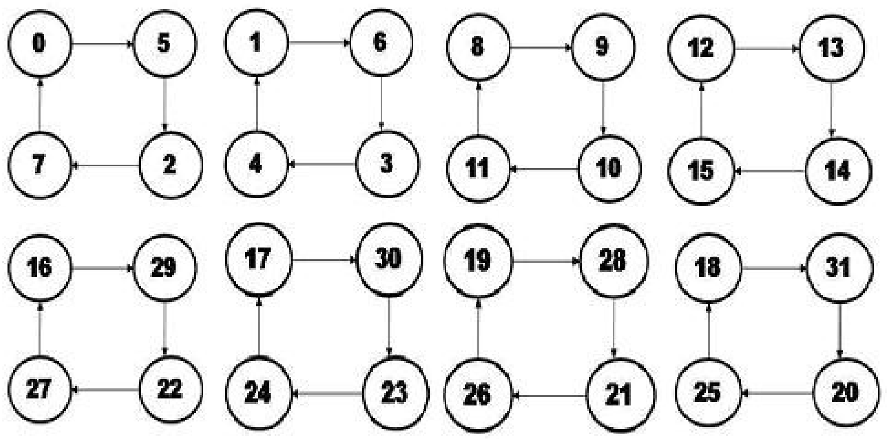


Fig. 4.1.2(c)

Fig. 4.1.2

Fig. 4.1.2(a). MaxCA Cycle for n=5 for <90, 90, 90, 90,150>

Fig. 4.1.2(b). Proposed 4 ELCA of cycle size 8 for < 153,153,153,153,153 >

Fig. 4.1.2(c). Proposed 8 ELCA of cycle size 4 for < 60, 60,195,102,153 >

Set of balanced CA rules for generation of ELCA as presented in Fig. 4.1.2 are: “51”, “60”, “102”, “153”, “195” and “204”. Few details of ELCA generating rules are presented in Table 4.1.1

Table 4.1.1. ELCA rule information

Serial no.	**CA rule**	**Binary equivalent of CA rule**	**Combinatorial binary logic for next state =i(t+1)**
1	51	00110011	NOT i(t)
2	204	11001100	i (t)
3	60	00111100	i-1(t) XOR i(t)
4	195	11000011	i-1(t) XNOR i(t)
5	102	01100110	i(t) XOR i+1(t)
6	153	10011001	i(t) XNOR i+1(t)

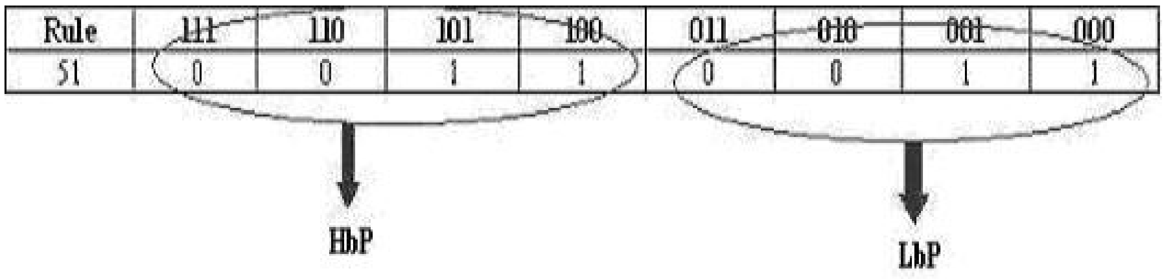
Binary representations of ELCA rules explore that all CA rules are balanced in nature. A pair of necessary and sufficient conditions are found for ELCA rules.

Necessary Condition for ELCA: Higher bits *partition (HbP) and lower bits partition (LbP) in binary representation for the CA rule, is balanced, i.e., HbP and LbP both individually contains two numbers of 0’s and two numbers of 1’s.*

Sufficient Condition for ELCA: *8 bit binary representation for the CA rule, is balanced, i.e., the binary representation contains four numbers of 0's and four numbers of 1's.*

Example 4.1.2.

Consider an ELCA generating balanced rule "51". Rule "51" are represented in 8 bit binary format as presented in Fig. 4.1.3.



Rule	111	110	101	100	011	010	001	000
51	0	0	1	1	0	0	1	1

Fig. 4.1.3. Binary representation of rule "51"

Binary representation for rule "51" is shown in Fig. 4.1.3. HbP (Higher bits partition) and LbP (lower bits partition) section show that they are individually balanced and rule "51" itself is balanced CA rule [2].

Corollary 4.1.1. ***All the balanced CA rules, whose HbP and LbP positions are not balanced, are not responsible for generating ELCA.***

Proof:

All the rules at their binary equivalent are having equal numbers of 0's and 1's in its HbP and LbP positions (Table 4.1.1). Total two numbers of 0's and two numbers of 1's are present at HbP and LbP positions for each ELCA rule. It indicates that all the ELCA rules are balanced at their HbP and LbP positions.

An unbalanced situation at HbP and LbP position never satisfy the necessity and sufficiency conditions. Hence no rule with unbalanced condition at HbP and LbP position is bound to produce ELCA cycles.

(End of Proof.)

Theorem 4.1.1. ***Space complexity is at least same for proposed methodology with respect to MaxCA.***

Proof:
Space complexity increases with the increased number of cells in a CA. Increased numbers of CA cells require more hardware component for implementation. Memory space required to process a fixed length of an n-cell CA is always same.

(End of Proof.)

Theorem 4.1.2. ***Design cost of proposed ELCA system is less than MaxCA.***

Proof:

Proposed methodology is allowed only to generate random patterns from smaller cycles that do not contain PPS. The PPS exclusion feature from the main cycle, which is responsible for generating random patterns, improves the design complexities. The logic behind this simplicity is that the proposed methodology simply discards the equal length cycles containing prohibited patterns. So there is no need to keep track of Arc_{min} length in cycles. Concept of Arc_{min} and E_{arc} is only applicable for MaxCA based design only.

Let the time taken for calculating Arc_{min} and E_{arc} are T_{arc} and T_E respectively in MaxCA. There is no concept of calculating T_{arc} and T_E in ELCA. All the PPS containing smaller cycles are discarded from pattern generation process. Time consumed for pattern generation T_{ELCA}. T_{ELCA} is free from the overhead of calculation of T_{arc} and T_E. Hence the design complexity of ELCA is simpler compared to MaxCA.

(End of Proof.)

Theorem 4.1.3. ***Time complexity is less in proposed methodology with respect to MaxCA.***

Proof:

Random pattern is only allowed to generate in proposed approach from PPS free equal length CA cycles which are smaller in size. So, execution time for smaller cycles are much less with respect to MaxCA. The time complexity of an n-length MaxCA is O (n). Time complexity for m-length ELCA is O (m). We have "m" is smaller than "n" by Equation 3.1.1.

Hence the time complexity of equal length is less than the time complexity of MaxCA.

(End of Proof.)

4.1.3. Experimental observations & result analysis

Data sets generated by different RNGs (i.e., Recursive PRNG, M-C PRNG, TRNG, MaxCA PRNG and ELCA PRNG) have been reported in Fig. 4.1.4 visualizing randomness of corresponding RNGs. The graph describes that the Recursive PRNG is having the least degree of randomness whereas CA PRNGs are better compared to all other RNGs.

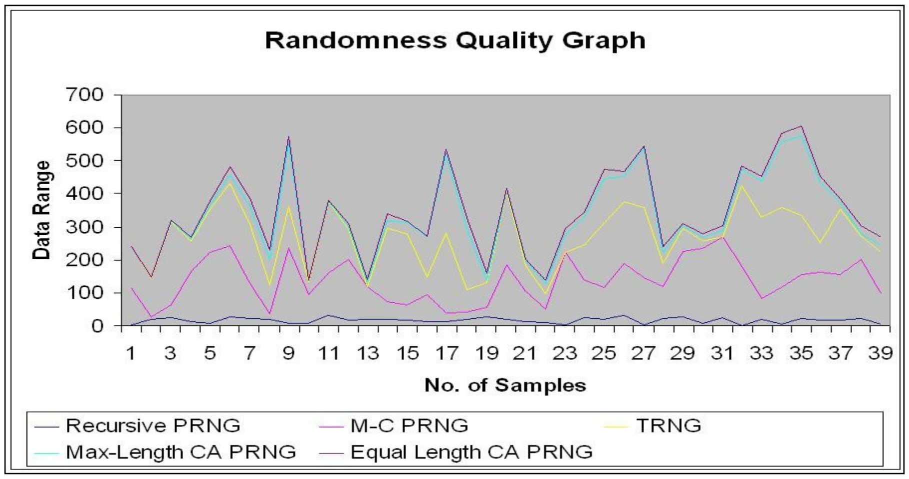


Fig. 4.1.4. Randomness quality graph for five different RNGs

The degree of randomness achieved by different random number generators in Diehard tests is presented in Table 4.1.2. Detailed Diehard result for TRNG has been described in [48].

Table 4.1.2. Performance results using Diehard for different RNGs

Serial	Name of the test	Recursive PRNG	M-C PRNG	TRNG
1	Birthday Spacings	Fail	Fail	Pass
2	Overlapping Permutations	Fail	Fail	Fail
3	Ranks of 31x31 and 32x32 matrices	Fail	Fail	Fail
4	Ranks of 6x8 Matrices	Fail	Fail	Pass
5	The Bitstream Test	Fail	Fail	Fail
6	Monkey Tests OPSO, OQSO, DNA	Fail	Fail	Pass
7	Count the 1`s in a Stream of Bytes	Fail	Fail	Pass
8	Count the 1`s in Specific Bytes	Fail	Fail	Pass
9	Parking Lot Test	Fail	Fail	Pass
10	Minimum Distance Test	Fail	Fail	Pass
11	The 3DSpheres Test	Fail	Fail	Pass
12	The Sqeeze Test	Fail	Fail	Fail
13	Overlapping Sums Test	Fail	Fail	Pass
14	Runs Test	Fail	Fail	Pass
15	The Craps Test	Fail	Fail	Pass
	Total No. of Passes	0	0	11

Results obtained for different RNGs (Table 3.1.2 and Table 4.1.2) are illustrated graphically in Fig. 4.1.5.

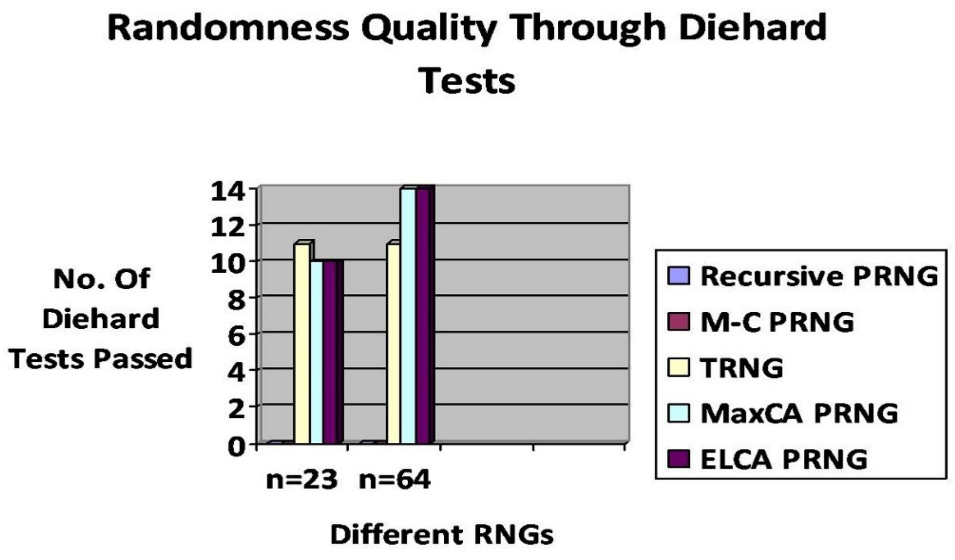


Fig. 4.1.5. Diehard test performance graph

The results obtained in Fig. 4.1.5 have ensured maximum degree of randomization in CA based design.

PPS exclusion policy in CA based PRNGs are briefly described as follows. PPS containing arc in random pattern generating cycle is excluded from the cycle in MaxCA (Refer Fig. 4.1.6). On the other hand, the PPS containing cycles are totally removed to generate the random sequences. There is no overhead to calculate E_{arc} and Arc_{min} in proposed ELCA methodology. Table 4.1.3 compares the procedures of MaxCA and ELCA based random pattern generators (Refer Fig. 4.1.6).

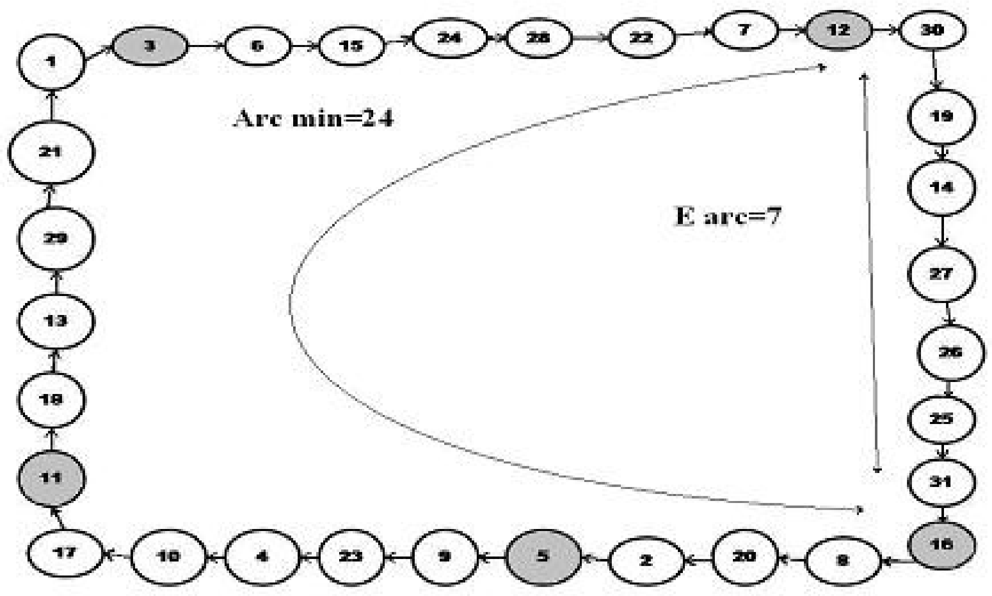


Fig. 4.1.6(a)

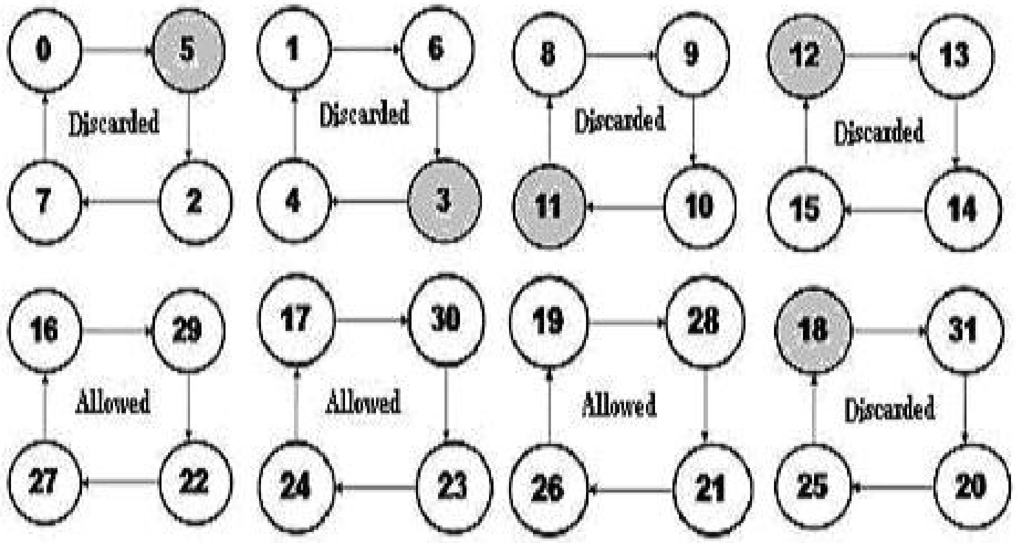


Fig. 4.1.6(b)
Fig. 4.1.6
Fig. 4.1.6(a). MaxCA cycle structure with PPS
Fig. 4.1.6(b). ELCA cycle structure with PPS

Fig. 4.1.6 is based on an arbitrarily drawn scenario where total number of prohibited patterns is 5 and let an arbitrary PPS is {5, 3, 11, 12, 16}. In worst case scenario, every single prohibited pattern is found in five independent cycle as illustrated in Fig. 4.1.6(b).

Comparison result between MaxCA and ELCA on this given set of PPS have been enlisted in Table 4.1.3.

Table 4.1.3. Comparison of fault coverage procedures

	MaxCA	**ELCA**
CA size(n)	5	5
E_{arc}	7	N/A
Arc_{min}	24	N/A

Advantages of the proposed methodology over MaxCA are presented in Table 4.1.2. No overhead for E_{arc} and Arc_{min} are found for ELCA. PPS exclusion policies for Recursive PRNG, M-C PRNG, TRNG are not available.

Complete data from MaxCA and ELCA PRNGs are shown in Fig. 4.1.7. Fig. 4.1.7 is based on MaxCA for <90, 90, 90, 90, 90, 150> and ELCA for <153,153,153,153,153,153>.

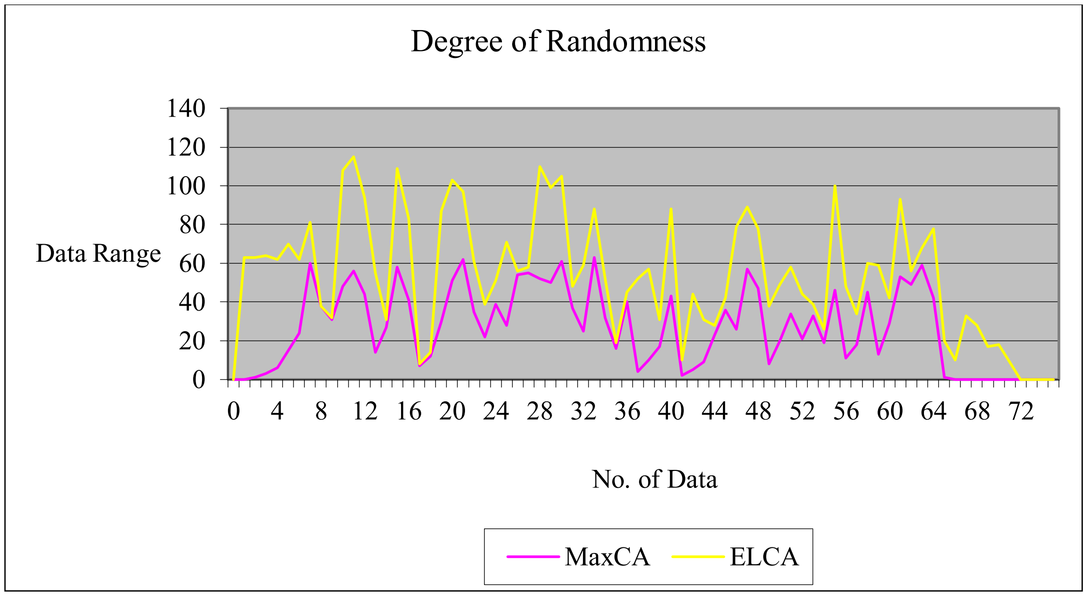


Fig. 4.1.7. Randomness quality graph for different CA-PRNGs

Hardware, time, design, and searching complexities for the said PRNGs are presented in Table 4.1.4. Required number of flip-flops for physical implementation of concerned PRNG system is referred to as hardware complexity. Time required for generation of a pseudo-random pattern by RNG is referred to as time complexity. The inherent design methodology dealing with problems of PPS is considered as design complexity. The complexity associated with searching PPS free pseudo-random patterns is referred to as searching complexity.

Table 4.1.4. Complexity comparison among different pattern generators

Name of the Complexity	**Recursive**	**M-C**	**TRNG**	**MaxCA**	**ELCA**	**Remarks**
Hardware	Not available	Not available	Not available	O(n)	O(n)	CA PRNGs are feasible for implementation using flip flops
Time	O (n) Here 'n' denotes number of iteration required in the concerned program.	O (n) Here 'n' denotes number of iteration required in the concerned program.	O (n) Here 'n' denotes number of iteration required in the concerned method.	O(n) Here 'n' denotes length of cycle.	$\sum O(m_i)$ Here 'm' denotes length of cycle and 'i' denotes number of ELCA.	Single cycle in ELCA has less time complexity.
Design	Require randomize () for random seed selection and no such particular method to deal with PPS.	Require specific mechanism for random seed selection and no such particular method to deal with PPS.	Require natural source for random seed selection and no such particular method to deal with PPS.	Requirements for Calculation of Arc_{min} to deal with PPS.	Does not require to calculate any Arc_{min} to deal with PPS.	ELCA is simpler design to deal with PPS.
Searching for PPS	No such particular method to deal with PPS.	No such particular method to deal with PPS.	No such particular method to deal with PPS.	Requirements for calculation of E_{arc}.	Does not require to calculate E_{arc}..	ELCA has simpler searching to deal with PPS.

Advantages of ELCA based PRNG over other RNGs are described in Table 4.1.4.

4.1.4. Summary

Quality of randomness achieved from the various samples of random data sets are verified. Good quality and cheap implementation are advantages for ELCA PRNG.

4.2. Set of Primes generation with CA

4.2.1. Introduction

Prime numbers [36-40, 71, 72] are used in cryptography [5, 73, 74] and stress testing. S. Wolfram focused on generation of primes in CA. CA rule "110" is used as the basis for some of the smallest universal Turing machines [75]. Several researches were carried out towards CA based primes generation [36-40, 76]. P. C. Fischer focused on the generation of primes in real-time by a single-dimensional iterative array [36]. J. Mazoyer presented prime generation in one-dimensional CA [37]. CA based prime number generation in several scenarios were described in [38-40]. In our studies, we have not found a cost effective and easy generation of set of primes in null boundary ECA scenario. Hence a simple and cost-effective set of primes generation may be advantageous towards cost-effective application.

Stress testing (torture testing) is an intense or thorough testing used to determine the stability of a distributed computing entity. Beyond normal operational capacity is often involved in stress testing, often to its breaking point to monitor the results [77].

The unique primality property of a prime number ensures that the prime number has exactly two divisors, one and the number itself. A number is decided to be a prime or composite number depending upon the result of successful pass or failure in primality test. Probability theory based Fermat primality testing [78] is used in primality testing. High degree of confidence in declaration of prime or composite number, low error ratios and faster execution are found in Fermat's Hypothesis for primes. Fermat Theory for primality is illustrated in Equation 4.2.1 and Equation 4.2.2 [78].

$$A^P \equiv A(\bmod P) \qquad (4.2.1)$$

where "P" is a prime number and "A" is a natural number.

Furthermore, if P*A ("A" is not divisible by "P"), then there is some minimum exponent "P" where

$$A^{P-1} \equiv 1(\bmod P) \qquad (4.2.2)$$

and "$A^{P-1} - 1$" is divisible by "P".

Example 4.2.1.

An example is followed for illustration of Fermat Primality Hypothesis.

Let a number “220” should be checked whether it is prime or composite.

For convention, let us consider a randomly choice value of “a” where, $1 \leq a< 220$, say $a = 37$. Now by definition,

$a^{n-1}= 37^{219}\equiv 1 \pmod{220}$.

Now, either 220 is prime, or 37 is a Fermat liar, so another value of ‘a’, say 29 is considered.

$a^{n-1}= 29^{219}\equiv 1 \pmod{220}$.

Therefore, 220 is composite and 37 is indeed a Fermat liar.

Rest of the section is organized as follows: proposed work is described in Sub-section 4.2.2; experimental results and analysis are in Sub-section 4.2.3; finally, summary is in Sub-section 4.2.4.

4.2.2. Proposed approach

ELCA generating rule along with rule “110” are proposed for cost effective generation procedure of set of maximum numbers of primes. Primality for produced number is verified using Fermat Hypothesis as described in Equation 4.2.2. Proposed system flowchart is described in Fig. 4.2.1.

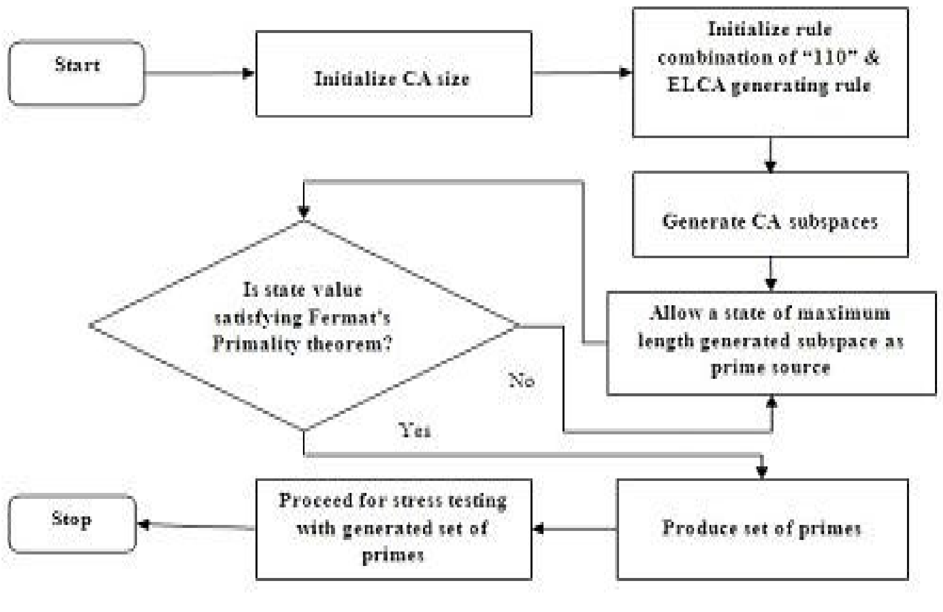

Start
Initialize CA size
Initialize rule combination of “110” & ELCA generating rule
Generate CA subspaces
Allow a state of maximum length generated subspace as prime source
Is state value satisfying Fermat's Primality theorem?
No
Yes
Produce set of primes
Proceed for stress testing with generated set of primes
Stop

Fig.4.2.1. Proposed flowchart of the system for stress testing

Proposed set of maximum number of primes generation is performed using Algorithm 4.2.1.

Algorithm 4.2.1. Stress_testing_for_distributed_computing_using_CA

```
Input: Cell size (n), rule "110", balanced ELCA rule(s )
Output: Stress test passed distributed system
Step1: Start
Step2: Initialize CA size and a combination of rule "110" and balanced ELCA rule(s)
Step3: Consider maximum length sub space for possible set of primes
Step4: Generate possible primes using Algorithm 2
Step5: Perform stress testing
Step6: Follow Step 2 for new prime set, else follow Step7
Step7: Stop
```

Algorithm 4.2.2. Possible_primes_set_generation

```
Input: Decimal state values for maximum length sub space
Output: A maximum length set of prime numbers (S)
Step1: Start
Step2: For every decimal values of state follow Step 3
Step3: Perform Fermat Primality testing as reported in Equation 2 and follow Step 4
Step4: If state value satisfies Fermat Primality then follow Step 5 else follow Step 6
Step5: update set of primes (S) with this state value and follow Step 6
Step6: Stop
```

Example 4.2.1.

Set of primes generation for <110,110,204> in null-boundary CA is presented in Fig. 4.2.2. The maximum length subspace is of length four and contains larger numbers of primes as compared to other subspaces. The maximum length subspace is considered in proposed approach as source of primes.

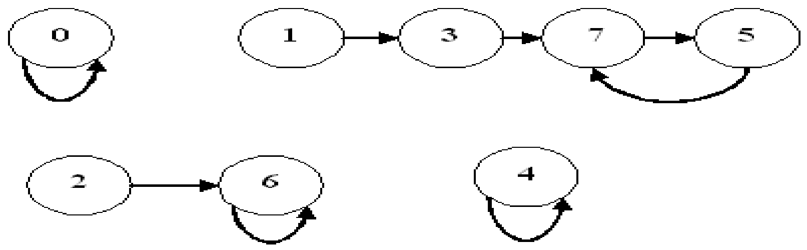


Fig. 4.2.2. Transition diagram for <110,110,204>

4.2.3. Experimental observations & result analysis

A combination of rule “110” and ELCA generating rule “204” are considered for generation of prime numbers. Detailed studies on these rules are presented in Table 4.2.1.

Table 4.2.1. Rule details for prime number computation

Serial no.	Rule	Binary equivalent	Next state computing function
1.	110	01101110	NOT i-1(t) AND i(t) AND i+1(t) XOR i(t) XOR i+1(t)
2.	204	11001100	i(t)

ELCA rule “204” along with “110” are responsible for length reduction in maximum length state space. ELCA rule “51” and “204” is not much efficient in length reduction in maximum length state space, but are capable to generate set of primes. Experimental results obtained for a data set generated using Algorithm 4.2.1 and Algorithm 4.2.2 are illustrated in Fig. 4.2.3. Data shown in Fig. 4.2.3(a) is based on <110, 110, 110, 110> and Fig. 4.2.3(b) is based on <110, 110, 110, 204 > in null boundary condition.

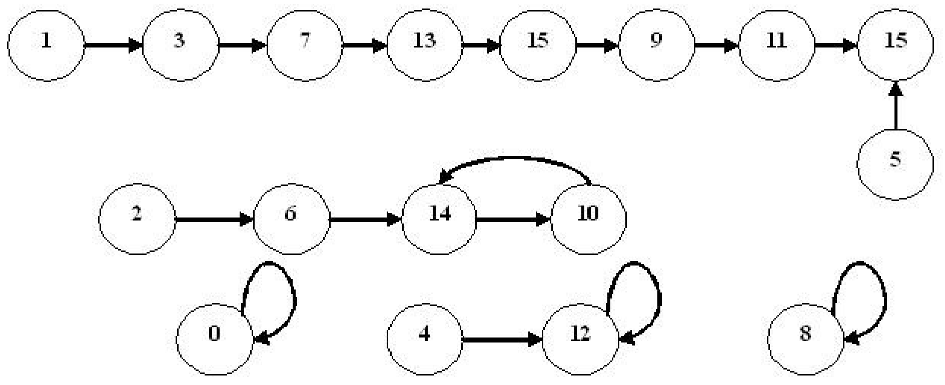


Fig. 4.2.3(a).

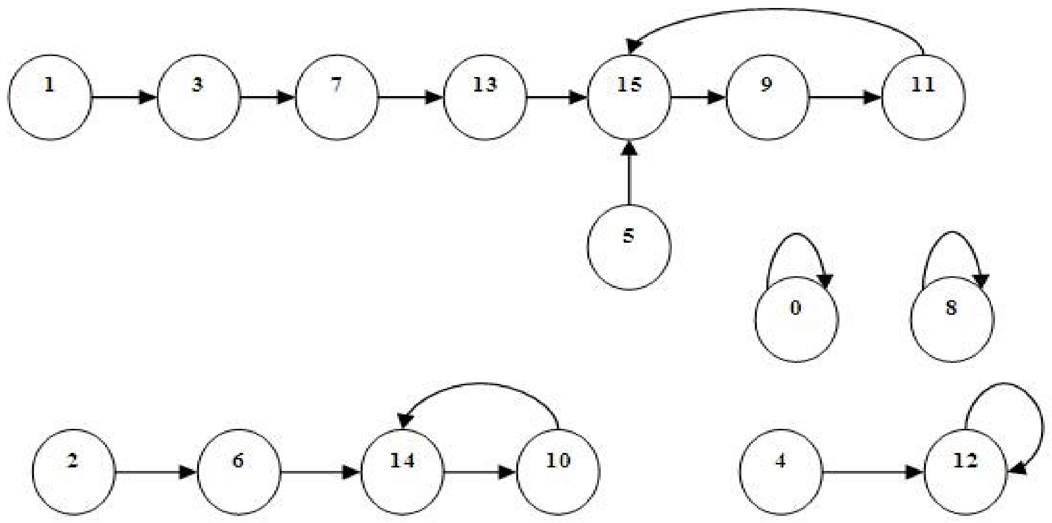


Fig. 4.2.3(b).
Fig. 4.2.3
Fig. 4.2.3(a). Transition diagram for <110,110,110,110>
Fig. 4.2.3(b). Transition diagram for <110,110,110,204>

State transitions are shown in Fig. 4.2.3. Algorithm 4.2.2 is applied on the maximum length state space as shown in Fig. 4.2.3. Performance of finding maximum number of primes in proposed methodology is reported in Table 4.2.2.

Table 4.2.2. Prime finding performance

Serial No.	**No. of Cells in CA (n)**	**No. of States in generated maximum subspace (s1)**	**No. of Primes found (p1)**	**Performance = (p1/s1)*100%**	**No. of States in generated maximum subspace (s2) in proposed method**	**No. of Primes found (p2) in proposed method**	**Performance = (p2/s2)*100% in proposed method**
1.	6	15	5	33.33%	14	5	35.71%
2.	7	17	7	41.18%	16	7	43.75%
3.	8	22	7	31.82%	21	7	33.33%

Proposed design methodology for obtaining maximum number of primes is found better in Table 4.2.2 as compared to existing method. Experimental results on different CA sizes are presented in Table 4.2.3.

Table 4.2.3. Generation of different sets of primes

Serial no.	**No. of cells in CA (n)**	**No. of states in generated maximum length subspace (s2)**	**No. of primes found (p2)**
1.	3	3	3
2.	4	8	4
3.	5	8	4
4.	6	14	5
5.	7	16	7
6.	8	21	7
7.	9	23	9
8.	10	29	10

Observation 4.2.1. Number of primes in generated maximum length state space is often equal to the CA size.

Observation 4.2.2. Number of primes in produced set almost increases with increased number of CA size.

Observation 4.2.3. Distribution of primes in generated state space is of random pattern.

Randomness in generated set of primes is reported in Fig. 4.2.4.

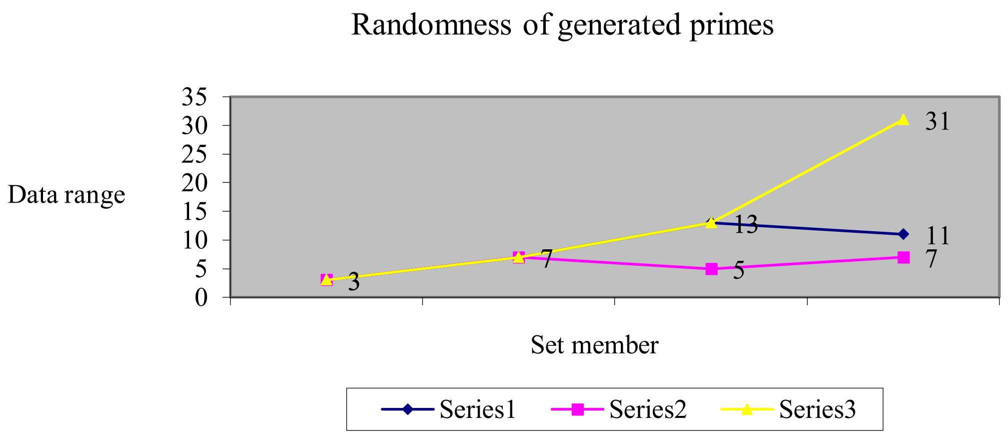


Fig. 4.2.4. Randomness of generated pattern of primes

Degree of randomness for set of primes as reported Series1, Series2 and Series3 in Fig. 4.2.4 are achieved for CA size 3, 4 and 5 respectively. Here first four members of the set are presented. Result obtained in Table 4.2.3 is illustrated in Fig. 4.2.5.

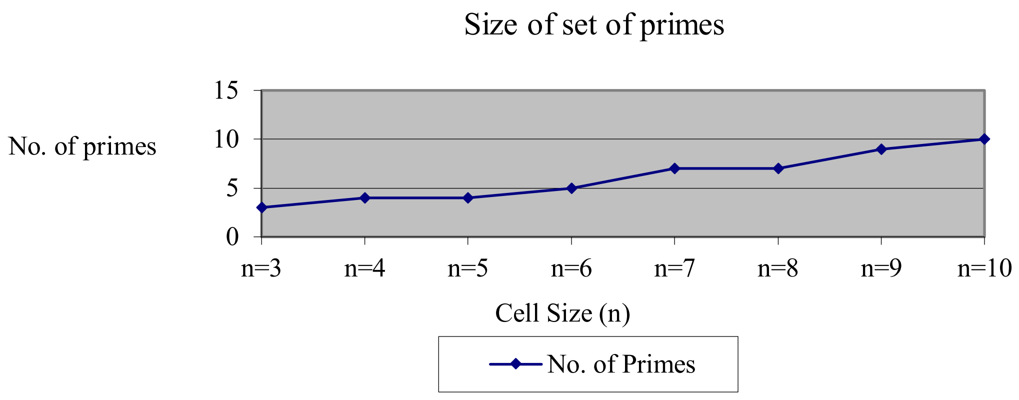


Fig. 4.2.5. Different sets of primes obtained with varying number of cell size

Observation 4.2.4. Same degree of randomness is found for both methodology. Graphical representation of equivalent randomness for CA size 4 is described in Fig. 4.2.6. Members of prime set are already shown in Fig. 4.2.3.

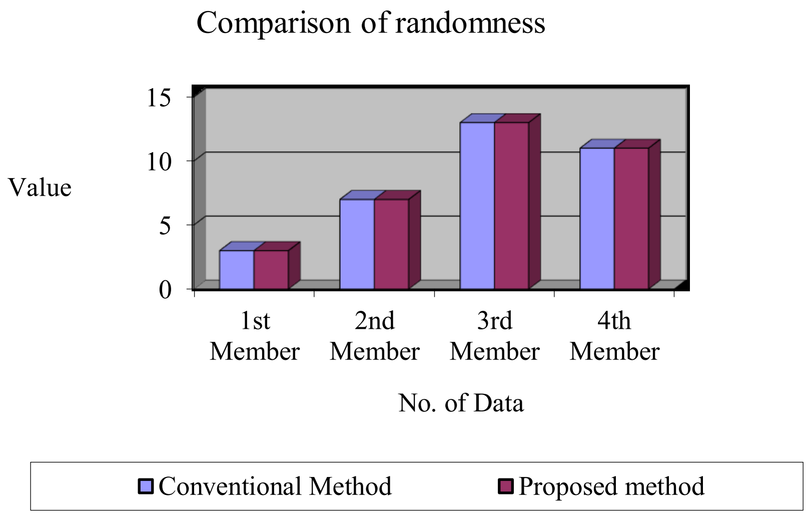


Fig. 4.2.6. Equivalency of degree of randomness in different applied methods

Observation 4.2.5. Performance for finding primes is increased with proposed approach.

Prime finding performance based on data reported in Table 4.2.2 is shown in Fig. 4.2.7

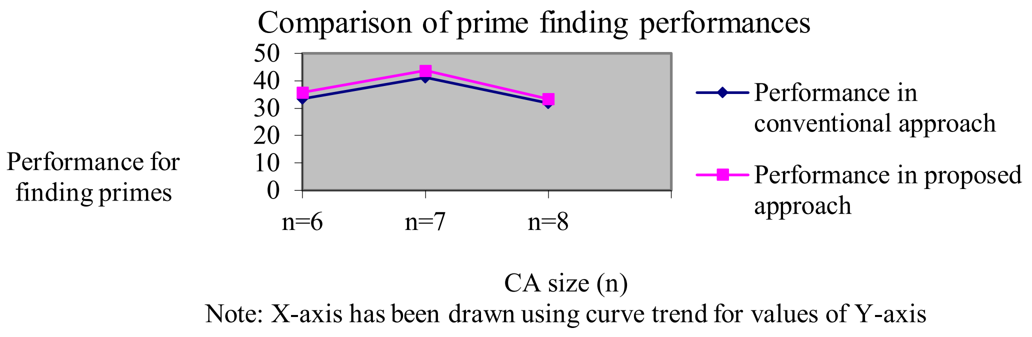


Fig. 4.2.7. Graphical representation of prime finding performance

Selection of rule for set of primes generation in null boundary CA is in Table 4.2.4.

Table 4.2.4. Class rule selection for primes

First rule	**Intermediate rule**	**Last rule**
110	110	204
110	110	51
110	110	153

4.2.4. Summary

Larger set of primes is produced from generated maximum length subspace. Length reduction for maximum length CA subspace without decreasing the number of primes present in that string is only possible with proposed methodology. Random distribution and cost-effectiveness in set of primes generation are found in proposed approach. Hence proposed method has a potential of uses in stress testing in distributed computing.

CHAPTER 5

ELCA SYNTHESIS AND ANALYSIS

5.1. CA rules exploration for ELCA generation

5.1.1. Introduction

Detailed studies with group CA in ECA scenario targeting several engineering applications were presented by researchers [2, 5-8]. Studies of group CA include maximum length [2], nonmaximal length [5], invertible CA [7, 8], and also equal length cycle CA (a special case of invertible CA) [8]. Nonmaximal length group CA is available in uniform CA scenario only [5] and ELCCA, as a special case of invertible CA with length 8 and 16 only is available in [8]. CA together with GA [79] based approach are considered towards modelling of complex and large systems including cryptosystems [2, 5, 73, 80, 81]. GA based CA rule synthesis [80, 81], pattern classification [73] were found in literature. CA based crypto system involving CA rule "51", "195" and "153" was described in [5, 74]. Unfortunately, we have not found complete set of CA rules for generation of equal length cycles both in uniform, and hybrid CA scenario, and easy synthesis of ELCA rules.

Motivations for this work are i) to explore the complete set of CA rules for ELCA generation, ii) to categorize the set of ELCA rules into linear and non-linear rules for detailed analysis of ELCA, and iii) to provide an easy synthesis of ELCA rules for generation of equal length cycles. Complete knowledge of ELCA rules is an essential criterion to explore the potential applications of ELCA in different scientific and engineering trends.

Rest of the section is organized as follows: proposed work is followed in Sub-section 5.1.2; analytical studies are in Sub-section 5.1.4; experimental results are presented in Sub-section 5.1.4; finally, summary is presented in Sub-section 5.1.5.

5.1.2. Proposed approach

Set of necessary and sufficiency conditions for ELCA generating rules are explored in Sub-section 4.1. Studies on ELCA generating rules in Sub-section 4.1 explored that balanced 'HbP' and 'LbP' configurations of balanced CA rule are responsible for generation of ELCA rules.

Cartesian product using all identical combinations [92] of balanced HbP and balanced LbP structures of linear ELCA rule, is proposed for exploring all ELCA generating balanced CA rules. Proposed Cartesian product is described in Fig. 5.1.1.

$$\begin{array}{ccccc}\textbf{HbP} & & \textbf{LbP} & & \textbf{Constructed CA rule}\\ \begin{pmatrix}0110\\1001\\0011\\1100\\0101\\1010\end{pmatrix} & \mathrm{X} & \begin{pmatrix}0110\\1001\\0011\\1100\\0101\\1010\end{pmatrix} & = & \begin{pmatrix}0110\ 0110\\ .\\ .\\ .\\ .\\ 1010\ 1010\end{pmatrix}\end{array}$$

Fig. 5.1.1. Proposed method for all possible ELCA rule construction

Total thirty-six number of balanced CA rules is achieved with proposed method as described in Fig. 5.1.1. All the thirty-six numbers of balanced CA rules satisfy both sufficiency and necessity conditions is described in Fig. 4.1.3 (in Sub-section 4.1). A few CA rules are found to produce ELCA cycles at uniform CA scenario (referred as primary rules for equal length cycle generation). Other rules are found to produce of equal length cycles in hybrid CA scenario. Primary rules for equal length cycle generation are presented in Table 5.1.1.

Table 5.1.1. Primary CA rules for ELCA generation

Serial	**CA rule**	**Binary equivalent of CA rule**
1	204	11001100
2	51	00110011
3	153	10011001

Observation 5.1.1. *Table 5.1.1 consists of three balanced rules among them there is one pair of rules which is complement to each other (rule "204" and "51").*

Observation 5.1.2. *HbP and LbP for each primary ELCA generating rules are identical with reference to their image property.*

Observation 5.1.3. *All the primary rules are self-reproductive in nature in terms of their HbP and LbP format.*

Equal length cycles are shown in Fig. 5.1.2, Fig. 5.1.3 and Fig. 5.1.4. State transition diagrams of Fig. 5.1.2, Fig. 5.1.3 and Fig. 5.1.4 are obtained for a 4-cell null boundary uniform CA using primary rules of Table 5.1.1.

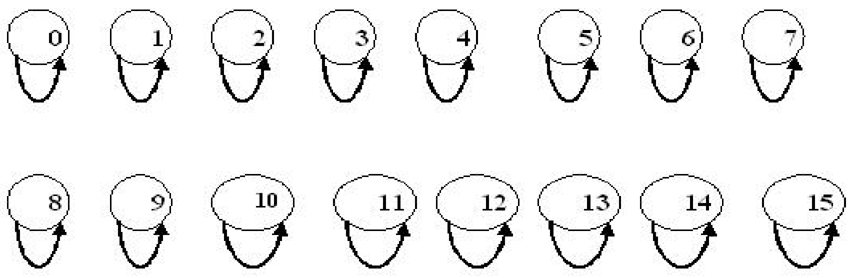


Fig. 5.1.2. State transition diagram for <204,204,204,204>

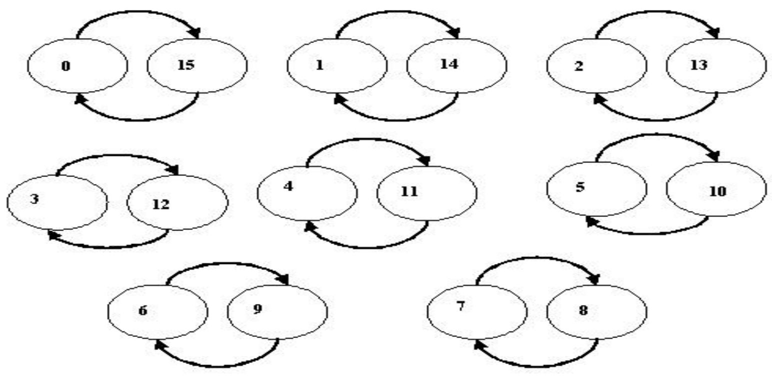


Fig. 5.1.3. State transition diagram for <51, 51, 51, 51>

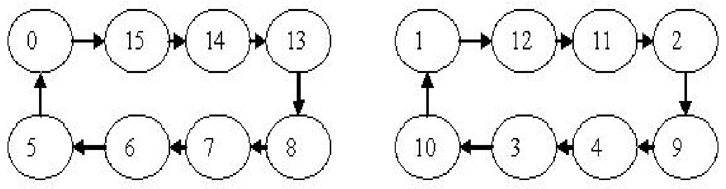


Fig. 5.1.4. State transition diagram for <153,153,153,153>

All ELCA generating rules are classified into some classes. Rule "204" and its complemented rule "51" can generate ELCA cycles by their own. Besides, this rule set is capable of synthesizing rule "195" and rule "60". Henceforth these two rules are kept together in a separate class. Remaining ELCA rules are categorized into different classes. Overviews for all ELCA generating rules along with their appropriate classes are presented in Table 5.1.2.

Table 5.1.2. Categorization of ELCA generating rules

Class	ELCA generating rules	Remarks
A	204, 51	Both of rules are capable of generating ELCA by their own, hence considered as primary rule set.
B	153, 102	Only Rule 153 is capable of generating ELCA by its own. Hence it is also a primary rule. Rule 102 is capable of generating ELCA only with a combination of primary rules.
C	195, 60	Rule 195 is achieved from Class A by a methodology discussed in Method 1. Rule 60 is complemented rule of Rule 195. This class is capable of generating ELCA with a combination of primary rules and also with a combination among themselves.
D	53, 54, 57, 58, 83, 85, 86, 89, 90, 92, 99, 101, 105, 106, 108, 147, 149, 150, 154, 156, 163, 165, 166, 169, 170, 172, 197, 198, 201, 202	All the rules are achieved from Class A, Class B, and Class C by a methodology as described in Method 1 and Method 2. All the rules are capable of generating ELCA with a combination of CA rules under Class A and Class B.

Rule "204" is the unique rule which is generating all single length CA cycle and is capable of synthesis of other balanced CA rules responsible for ELCA generation. Two different algorithms are followed to regenerate all other ELCA generating rules from rule "204". An intra-crossover based design is reported in Sub-section 5.1.2.1. and an inter-crossover based design is reported in Sub-section 5.1.2.2.

5.1.2.1. Intra-Crossover Design (bit wise crossing over at HbP / LbP)

An identical structure with reference to the HbP and LbP is found for ELCA generating primary rules. Identical HbP and LbP is observed for rule "204". Therefore, it is easy to implement same operation at both partitions of the rule "204". HbP / LbP structure for rule "204" is presented in Fig. 5.1.5.

111	110	101	100
1	1	0	0

Fig. 5.1.5. HbP/LbP structure of rule "204"

Lexicographical order [82] is used to synthesize other rules from the HbP / LbP structure of rule "204". Proposed approach is presented in Fig. 5.1.6.

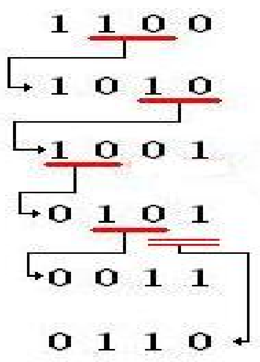


Fig. 5.1.6. All lexicographical order

Six different HbP structures are generated in Fig. 5.1.6. All six different HbP structures have successfully generated thirty-six balanced ELCA rules (Fig. 5.1.1). Described approach of Fig. 5.1.6 is used in following Sub-section 5.1.2.2.

5.1.2.1. Intra-Crossover based Algorithm

Algorithm 5.1.2.1. Intra_crossing_over_for_ELCA_rule_synthesis

Input: *HbP or, LbP of primary rule*

Output: *All balanced CA rules responsible for ELCA generation*

```
Step1: Start
Step2: Initialize with binary value of HbP or, LbP and store it as parent
Step3: If consecutive 0 and 1 are found in parent then goto Step4 else goto Step2
Step4: Perform complementation operation for both of the position
Step5: If generated child is balanced then follow Step6 else follow Step 8
Step6: If generated child is not in existing parent database then follow Step7 else follow
        Step9
Step7: Store it into class 'A' as future parent and goto Step3
Step8: Discard the child and goto Step3
Step9: Make one clone class 'B' for all generated valid parents of class 'A' and follow
        Step10
```

Step10: Perform Cartesian production operation on class 'A' and class 'B' and follow Step11

Step11: Store all results in class 'C'

Step12: Stop

5.1.2.2. Inter-crossover Design (bit wise crossing over between HbP and LbP)

All HbPs along with their complemented structures of the primary rules are crossed over with all LbPs along with their complemented structures of the primary rules. All the resulting child rules are ELCA generating rules satisfying both necessity and sufficiency conditions. Described approach is developed in Sub-section 5.1.2.2.

5.1.2.2 Inter-crossover based Algorithm

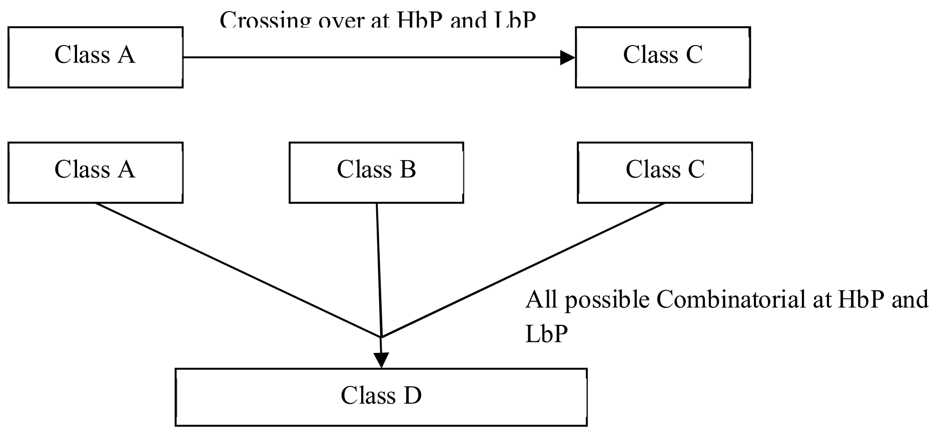
Algorithm 5.1.2. Inter_crossing_over_for_ELCA_rule_synthesis

Input: *Binary form of Class 'A' and Class 'B' rules*

Output: *All balanced CA rules responsible for ELCA generation*

```
Step1: Start
Step2: Initialize class 'A' with binary value of Class 'A' and Class 'B' rules
Step3: Decompose each rule into HbP and LbP
Step4: Perform all possible HbP and LbP positional crossing over for data stored in
       class 'A' and update class 'A' if it is not already contained in class 'A'
Step5: Perform bit-wise complementation operation for all of the HbP and LbP positions
       stored in class 'A'
Step6: Perform all possible HbP and LbP positional crossing over for data stored in
       class 'A'
Step7: If the rule is not existing into class 'A' then update class 'A' with it and follow Step8
Step8: Stop
```

Proposed synthesis methodology for ELCA rules as described in Algorithm 5.1.2.2 is presented diagrammatically in Fig. 5.1.7.



Crossing over at HbP and LbP
Class A
Class C
Class A
Class B
Class C
All possible Combinatorial at HbP and LbP
Class D

Fig. 5.1.7. All ELCA generating rule synthesis methodology

Motivation of Theorem 5.1.1 is to find out the mathematics behind exploration of the complete set of ELCA rules.

Theorem 5.1.1. Maximum numbers of ELCA generating balanced rules are thirty-six only.

Proof:

ELCA generating CA rules are balanced and at the same time, HbP and LbP structures of the balanced CA rule are also balanced by the set of necessity and sufficiency condition. Hence, the maximum number of combinations satisfying both the necessity and sufficiency conditions is ${}^4C_2$ (=6). So, the maximum number of balanced rules with ${}^4C_2$ number of balanced configurations in HbP or, LbP position is, ${}^4C_2* {}^4C_2=36$.

Therefore, maximum number of ELCA generating rules is ${}^4C_2* {}^4C_2=36$.

(End of proof.)

Corollary 5.1.1. All equal length cycle generating CA rules are balanced CA rules. Reverse is not true.

Proof:

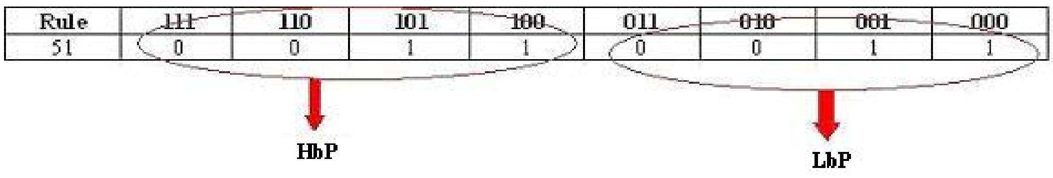
All ELCA generating rules are balanced as well as they are balanced at their HbP and LbP positions (the necessity and sufficiency conditions).

On the contrary it may happen that certain balanced CA rule contains equal number of 0's and 1's in its binary representation but it's HbP and LbP individually contains unequal distribution of 0’s and 1’s. Therefore, this type of CA rule violates the necessity and sufficiency conditions. Henceforth, ELCA generation is not confirmed.

(End of proof.)

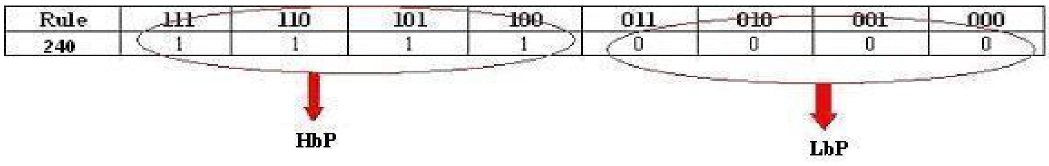
Example 5.1.1.

Let rule “51” and rule “240” for explanation of Corollary 5.1.1. Both of the rules are balanced. But only rule “51” is responsible for generating ELCA. Rule “240” does not satisfy necessity and sufficiency conditions. Binary representation structure for rule “51” is shown in Fig. 5.1.8(a) and Binary representation of rule “240” is described in Fig. 5.1.8(b).



Rule	111	110	101	100	011	010	001	000
51	0	0	1	1	0	0	1	1

5.1.8(a). Rule “51”



Rule	111	110	101	100	011	010	001	000
240	1	1	1	1	0	0	0	0

5.1.8(b). Rule “240”

Fig. 5.1.8. Binary representation of Rules

Motivations of Theorem 5.1.2 and Corollary 5.1.2 are to categorize the ELCA rules appropriately into linear and non-linear CA rules using basic binary operation.

Theorem 5.1.2. If the bit-wise OR/AND operation among balanced HbP and balanced LbP of a balanced rule results an even number of 0’s in result, then it is a linear rule or complemented rule.

Proof:

Binary representations of balanced ELCA rules explore that a bit wise OR/AND operation among HbP and LbP produce a result containing even number of 0’s. The next state calculating binary functions for these rules employ XOR/XNOR logic only. Hence the statement is true.

(End of Proof.)

Corollary 5.1.2. If the bit-wise OR/AND operation among balanced HbP and balanced LbP of a balanced rule results an odd number of 0's in result, then it is a non-linear rule.

Proof:

Binary representations of balanced ELCA rules explore that if the bit wise OR/AND operation among HbP and LbP of a concerned balanced rule does not produce a result containing even number of 0's then the next state calculating binary functions for these rules employ AND-OR logic only. Hence the statement is true.

(End of Proof.)

Let ELCA rule "102", "153", "201", and "54" for explanation of Theorem 5.1.2 and Corollary 5.1.2. Bit wise AND/OR operation among HbP and LbP of the concerned rule has been described in Sub-section 5.1.3.

5.1.3. Analytical studies

5.1.3.1. Binary representation for rule "102"

Table 5.1.3. Rule "102" in binary representation

Rule	HbP				LbP			
102	0	1	1	0	0	1	1	0

Bit-wise OR operation for rule "102"

```
0    1    1    0

0    1    1    0

-----------------------------------------

0    1    1    0
```

Bit-wise AND operation for rule "102"

```
0    1    1    0

0    1    1    0

-----------------------------------------

0    1    1    0
```

Next state calculating binary function for rule “102”

i(t+1)=i(t) XOR i+1(t) (5.1.1)

Both bit-wise OR/AND operation among HbP and LbP of rule “102” results two number of 0’s in the outcome and the next state calculating function employs XOR logic. Hence rule “102” is a linear rule.

5.1.3.2. Binary representation for rule “153”

Table 5.1.4. Rule “153” in binary representation

Rule	HbP				LbP			
153	1	0	0	1	1	0	0	1

Bit-wise OR operation for rule “153”

```
1    0    0    1

1    0    0    1

---------------------------------------

1    0    0    1
```

Bit-wise AND operation for rule “153”

```
1    0    0    1

1    0    0    1

---------------------------------------

1    0    0    1
```

Next state calculating binary function for rule “153”

i(t+1)=i(t) XNOR i+1(t) (5.1.2)

Both bit-wise OR/AND operation among HbP and LbP of rule “153” results even number of 0’s in the outcome and the next state calculating function employs XNOR logic. Hence rule “153” is a complemented rule.

5.1.3.3. Binary representation for rule "201"

Table 5.1.5. Rule "201" in binary representation

Rule	HbP				LbP			
201	1	1	0	0	1	0	0	1

Bit-wise OR operation for rule "201"

```
1      1      0      0

1      0      0      1

----------------------------------------

1      1      0      1
```

Bit-wise AND operation for rule "201"

```
1      1      0      0

1      0      0      1

----------------------------------------

1      0      0      0
```

Next state calculating binary function for rule "201"

$$i(t+1)=(\text{NOT } (i-1(t) \text{ OR } i+1(t))) \text{ OR } i(t) \qquad (5.1.3)$$

Both bit-wise OR/AND operation among HbP and LbP of rule "201" results odd number of 0's in the outcome and the next state calculating function does not employ any XOR/XNOR logic. Combination of OR logic have been considered. Hence rule "201" is a non-linear rule.

5.1.3.4. Binary representation for rule "54"

Table 5.1.6. Rule "54" in binary representation

Rule	HbP				LbP			
54	0	0	1	1	0	1	1	0

Bit-wise OR operation for rule "54"

```
0       0       1       1

0       1       1       0

----------------------------------------

0       1       1       1
```

Bit-wise AND operation for rule "54"

```
0       0       1       1

0       1       1       0

----------------------------------------

0       0       1       0
```

Next state calculating binary function for rule "54"

i(t+1)= (i-1(t) OR i+1(t)) OR i(t) (5.1.4)

Both bit-wise OR/AND operation among HbP and LbP of rule "54" results odd number of 0's in resultant. Furthermore, next state calculating function of rule "54" does not employ any XOR/XNOR logic; only combination of OR logic is found. Hence rule "54" is a non-linear rule.

Motivations of Theorem 5.1.3 and Corollary 5.1.3 are to establish class-relationships among the specified classes as referred in Table 5.1.3 and Table 5.1.4.

Theorem 5.1.3. In ELCA rule synthesis, an evolutionary approach with intra-combination to class 'A' always produces another additive CA class rules.

Proof:

Intra-combination among balanced HbP and balanced LbP of a "class A" rule produces a balanced class rule which is also balanced at its HbP and LbP. Class A rules are additive in nature and they also satisfy another property that their HbP and LbP is image to each other.

The pair of rules in class A is complemented rule to each other. Therefore, the intra-combination among the HbP and LbP generates a set of balanced rules with balanced HbP and LbP where the HbP and LbP is mirror image to each other. The next state function for these rules simply imply the implementation of XOR/XNOR logic.

Hence the new derived class is additive CA class rules.

(End of Proof.)

Corollary 5.1.3. An evolutionary approach with inter combination between two additive classes mostly produces non-additive CA class rules.

Proof:

Inter combination among balanced HbP and balanced LbP of different class rules produce a balanced class rule which is balanced at its HbP and LbP, but their HbP and LbP is mostly different i.e., they do not satisfy any image or mirror image property.

The inter combination among the HbP and LbP generates a set of balanced rules with balanced HbP and LbP where the HbP and LbP is neither image or mirror image to each other. The next state function for these rules simply imply the implementation of AND-OR logic instead of XOR/XNOR logic.

Hence the new derived class is non-additive CA class rules.

(End of Proof.)

5.1.4. Experimental results

Detailed experimental results obtained in computer simulation are reported in Sub-section 5.1.4.1, and Sub-section 5.1.4.2.

5.1.4.1. Real time activities

ELCA pattern generation is presented in Table 5.1.7.

Table 5.1.7. Different equal length cycles generated by primary rules

Serial Number	**Rule**	**Cell Size (n)**	**Cycle Length (m)**
1	204	3→31	1
2	51	3→31	2
3	153	3	4
		4→7	8
		8→15	16
		16→31	32

Different uniform and hybrid ELCA scenarios and corresponding rule selection have been reported in Sub-section 5.1.4.2.

5.1.4.2. Efficient Usage

Class rule selection for uniform CA is presented in Table 5.1.8 and class rule selection for hybrid CA is in Table 5.1.9, Table 5.1.10 and Table 5.1.11.

Table 5.1.8. Class rule selection for uniform ELCA

Class of R_i	**R_i**	**Class of R_{i+1}**	**R_{i+1}**
A	204	A	204
A	51	A	51
B	153	B	153

Table 5.1.9. Class rule selection for hybrid ELCA

First rule class	**First rule R_{i-1}**	**Intermediate rule class**	**Intermediate rules R_i**	**Last rule class**	**Last rule R_{i+1}**
A	204	A, B, C	51,60,102,195	A, B	51,204
A	51	A, B, C	51,60,102,153,195,204	A, B, C	51,102,153,204
B	153	A, B, C	102,153	A, B, C	51,60,204
B	102	A, B, C	51,60,102,153,195,204	A, B	51,153
C	195	A, B, C	60,195	A, B, C	60,102,153,195
C	60	A, B, C	60,102,153,195	A, B, C	51,60,153

Table 5.1.10. Special cases for class rule selection for hybrid ELCA

Cell size (n)	**First rule**	**Intermediate rules in alternative selection**	**Last rule**
Odd	204	153/204	204
Even	204	153/204	153

Table 5.1.11. Special cases for class rule selection for hybrid ELCA

Cell size (n)	**First rule**	**Intermediate rules in alternative selection**
n>=4	204	99/204
	204	57/204
	51	57/51
	51	99/51
	99	204/99
	99	204/51
	57	204/57
	57	51/57

5.1.5. Summary

All balanced CA rules, uniform and hybrid scenario responsible for generation of equal length cycles are explored. Synthesis for all reported ELCA rules from primary ELCA rule and hierarchical relationship among ELCA rules are described.

5.2. Analysis of ELCA generating linear rules

5.2.1. Introduction

An n-cell CA with linear rules is characterized by an $[n \times n]$ square matrix. Construction of characteristics matrix M [i,j] is defined as Equation 5.2.1 [73].

$$\text{M [i,j]}=\begin{cases} 1, \; \textit{if the next state of the } i-th \textit{ cell is dependent} \\ \quad \textit{on the present state of the } j-th \textit{ cell} \\ 0, \; \textit{otherwise} \end{cases} \qquad (5.2.1)$$

Example 5.2.1.

Let R=<102, 150, 170, 204>

$$M_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{4\times 4}$$

Characteristics Polynomial of matrix is obtained by constructing the matrix [M] + x [I] and computing the corresponding determinant [M+xI]; where [I] is identity matrix.

State transition of a linear CA is defined as Equation 5.2.2 [73].

$$Y=M(x) \qquad (5.2.2)$$

where, CA input is denoted by an n-bit vector ‘x’;

‘Y’ denotes the output bit vector of the CA;

Construction of Characteristics polynomial from characteristics matrix has been illustrated in Example 5.2.2.

Example 5.2.2.

$$[M + xI] = \begin{vmatrix} 1+x & 1 & 0 & 0 \\ 1 & 1+x & 1 & 0 \\ 0 & 0 & x & 1 \\ 0 & 0 & 1 & x \end{vmatrix}$$

$$= \begin{vmatrix} 1 & 1+x & 0 & 0 \\ 1+x & 1 & 1 & 0 \\ 0 & 0 & x & 1 \\ 0 & 0 & 1 & x \end{vmatrix} (\text{swap } C_0 \text{ and } C_1)$$

$$= \begin{vmatrix} 1 & 0 & 0 & 0 \\ 1+x & (1+x)^2+1 & 1 & 0 \\ 0 & 0 & x & 1 \\ 0 & 0 & 1 & x \end{vmatrix} (C_1 = C_0*(1+x)+ C_1)$$

$$= \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & x^2 & 1 & 0 \\ 0 & 0 & x & 1 \\ 0 & 0 & 1 & x \end{vmatrix} (R_1 = R_0*(1+x)+ R_1) \text{ and } (1+x)^2+1 \Rightarrow x^2)$$

$$= \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & x^2 & 0 \\ 0 & x & 0 & 1 \\ 0 & 1 & 0 & x \end{vmatrix} (\text{swap } C_1 \text{ and } C_2)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & x & x^3 & 1 \\ 0 & 1 & x^2 & x \end{vmatrix} (\mathrm{C}_2 = \mathrm{C}_1 * \mathrm{x}^2 + \mathrm{C}_2)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & x^3 & 1 \\ 0 & 0 & x^2 & x \end{vmatrix} (\mathrm{R}_2 = \mathrm{R}_1 * \mathrm{x} + \mathrm{R}_2) \text{ and } (\mathrm{R}_3 = \mathrm{R}_1 + \mathrm{R}_3)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & x^3 \\ 0 & 0 & x & x^2 \end{vmatrix} (\text{swap } \mathrm{C}_3 \text{ and } \mathrm{C}_2)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & x & x^4 + x^2 \end{vmatrix} (\mathrm{C}_3 = \mathrm{C}_2 * x^3 + C_3)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x^4 + x^2 \end{vmatrix} (\text{swap } \mathrm{R}_3 = \mathrm{R}_2 * x + R_3)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & x & x^4 + x^2 \end{vmatrix} (\mathrm{C}_3 = \mathrm{C}_2 * x^3 + C_3)$$

$$=\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x^4 + x^2 \end{vmatrix} (\mathrm{R}_3 = \mathrm{R}_2 * x + R_3)$$

Minimal polynomial for given matrix is (x^4+x^2).

Therefore, the characteristic polynomial is 1.1.1.(x^4+x^2).

Characteristics Polynomial is given by Equation 5.2.3.

$$Y = x^4 + x^2 \tag{5.2.3}$$

Aim of this section is to explore the algebraic properties of ELCA generating linear CA rules and to find out the mathematical relationship between cell length and cycle length.

Rest of the section is organized as follows: ELCA generating linear and complemented linear rules are reported in Sub-section 5.2.2; formal analysis on ELCA generating linear and complemented linear rules are presented in Sub-section 5.2.3; experimental observations are reported in Sub-section 5.2.4 and finally summary is in Sub-section 5.2.5.

5.2.2. ELCA generating linear and complemented linear rules

ELCA generating linear and complemented linear rules are reported in Table 5.2.1.

Table 5.2.1. ELCA generating linear and complemented linear rules

Serial number	Rule	Binary representation	Next state function	Cell dependency
1.	60	00111100	i-1(t) XOR i(t)	Left & self
2.	90	01011010	i-1(t) XOR i+1(t)	Left & right
3.	102	01100110	i(t) XOR i+1(t)	Self & right
4.	150	10010110	i-1(t) XOR i(t) XOR i+1(t)	Left, self & right
5.	170	10101010	i+1(t)	Right
6.	204	11001100	i(t)	Self
7.	195	11000011	i-1(t) XNOR i(t)	Left & self
8.	165	10100101	i-1(t) XNOR i+1(t)	Left & right
9.	153	10011001	i(t) XNOR i+1(t)	Self & right
10.	105	01101001	i-1(t) XNOR i(t) XNOR i+1(t)	Left, self & right
11.	85	01010101	NOT i+1(t)	Right
12.	51	00110011	NOT i(t)	Self

5.2.3. Formal analysis on ELCA generating rules

Characteristic matrix for ELCA generation using linear rule '204' (in an n-cell uniform CA) is shown in Equation 5.2.4. Equation 5.2.2 and cell dependencies (refer Table 5.2.1) are utilized for construction of characteristic matrix.

$$T_{n_{linear}} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{n \times n} \qquad (5.2.4)$$

Characteristic polynomial for given matrix is shown in Equation 5.2.5.

$$Y_{204}= (x+1) \qquad (5.2.5)$$

Characteristic matrix for ELCA generation using complemented linear rule '51' is shown in Equation 5.2.6. Equation 5.2.6 and cell dependencies (refer Table 5.2.1) are used for construction of characteristic matrix.

$$T_{n_{complementedLin}} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{n \times n} \qquad (5.2.6)$$

Characteristic polynomial for given matrix is shown in Equation 5.2.7.

$$Y_{51}= (x+1) \qquad (5.2.7)$$

Observation 5.2.1. Minimal polynomials for an n-cell uniform CA using rule '204' and '51' are same.

Observation 5.2.2. Characteristic polynomials for an n-cell uniform CA using rule '204' and '51' are same.

Theorem 5.2.1. Fixed length ELCA is generated from uniform CA, only if characteristic polynomial has a form of (x+1).

Proof:

All single length ELCA are produced using linear rule '204' and all double length ELCA are generated using complemented linear rule '51' in null-boundary uniform CA. Rules

‘204’ & ‘51’ have next state dependency on self-cell only. Other linear ELCA rules do not have next state dependency on self-cell only (refer Table 5.2.1 and Table 5.2.2).

Therefore, characteristic polynomial (x+1) (refer Equation 5.2.4 and Equation 5.2.6) only produces fixed length ELCA.

(End of proof.)

Characteristic matrix for linear rule ‘153’ in an n-cell uniform CA is illustrated in Equation 5.2.8. Equation 5.2.8 and cell dependencies (refer Table 5.2.1) are used for construction of characteristic matrix.

$$T_{n_{complementedLin}} = \begin{bmatrix} 1 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 1 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ . & & & & \\ . & & & & \\ . & & & & \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}_{n \times n} \qquad (5.2.8)$$

Characteristic polynomial for given matrix is shown in Equation 5.2.9.

$$Y_{153} = (x+1)^n \qquad (5.2.9)$$

Theorem 5.2.2. Variable length ELCA is generated from uniform CA, only if characteristic polynomial has a form of (x+1)^n.

Proof:

ELCA are produced using linear rule ‘153’ in null-boundary uniform CA. Next state dependency for rule ‘153’ is on self-cell and right cell only (refer Table 5.2.1). Characteristic polynomial only produces a polynomial $(x+1)^n$ (refer Equation 5.2.9).

(End of proof.)

Example 5.2.3.

Let, R = <153, 153, 153, 153>.

Characteristic matrix is considered as follows.

$$T_{153}=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}_{4\times 4}$$

All elements in main-diagonal and on-diagonal are ‘1’.

Characteristic matrix for ELCA generation using an n-cell hybrid CA is shown in Equation 5.2.10 for given rule set “R = <204, Ri…..Ri>” having “Ri = <60, 195, 51>”.

$$T_n = \begin{bmatrix}1 & 0 & 0 & \ldots & 0\,0\\ 1 & 1 & 0 & \ldots & 0\,0\\ 0 & 1 & 1 & \ldots & 0\,0\\ 0 & ..\,0 & 1 & \ldots & 0\,0\\ . & \ldots & & & .\\ 0 & \ldots & & & 1\;0\\ 0 & 0 & 0 & \ldots & 0\;1\end{bmatrix}_{n\times n} \qquad (5.2.10)$$

Characteristic polynomial for given hybrid matrix is shown in Equation 5.2.11.

$$Y_{\text{hybrid}} = (x+1) \qquad (5.2.11)$$

Different characteristic matrices for ELCA generation using n-cell hybrid CA are presented in Equation 5.2.12, Equation 5.2.13, Equation 5.2.14, and Equation 5.2.15.

$$T_{n_1} = \begin{bmatrix}1 & 0 & 0 & \ldots & 0\,0\\ 1 & 1 & 0 & \ldots & 0\,0\\ 0 & 0 & 1 & \ldots & 0\,0\\ 0 & ..\,1 & 1 & \ldots & 0\,0\\ . & \ldots & & & .\\ 0 & \ldots & & & 1\;0\\ 0 & 0 & 0 & \ldots & 0\;1\end{bmatrix}_{n\times n} \qquad (5.2.12)$$

$$T_{n_2} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & \ldots & 0 & 0 \\ 0 & ..0 & 1 & \ldots & 0 & 0 \\ . & \ldots & & & & . \\ 0 & \ldots & & & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}_{n\times n} \qquad (5.2.13)$$

$$T_{n_3} = \begin{bmatrix} 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ 0 & ..0 & 1 & \ldots & 0 & 0 \\ . & \ldots & & & & . \\ 0 & \ldots & & & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}_{n\times n} \qquad (5.2.14)$$

$$T_{n_4} = \begin{bmatrix} 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 0.. & 0 & 0 \\ 0 & ..0 & 1 & \ldots & 0 & 0 \\ . & \ldots & & & & . \\ 0 & \ldots & & & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}_{n\times n} \qquad (5.2.15)$$

Equation 5.2.12 is generated for "R = <204, R_{nl},....R_{nl}>" having "R_{nl} = <60, 51>".

Equation 5.2.13 is generated for "R = <153, R_{n2},...R_{n2}>" having "R_{n2} = <60, 195, 51>".

Equation 5.2.14 is generated for "R = <153, $R_{n3....}$ R_{n3}>" having "R_{n3} = <51, 102>".

Equation 5.2.15 is generated for "R = <153, $R_{n4..........}$ R_{n4}, 51>" having "R_{n4} = <102, 51, 60>".

Observation 5.2.3. Variable length ELCA is generated in hybrid CA scenario, if characteristic matrix forms a tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal or off-diagonal positions are occupied by "1, 0" pattern sequence. Corresponding determinant of characteristic matrix is equal to one.

Observation 5.2.4. Variable length ELCA is generated from hybrid CA, if characteristic matrix forms a tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal or off-diagonal positions are occupied by "1, 1, 0" pattern sequence. Corresponding determinant of characteristic matrix is equal to one.

Observation 5.2.5. ELCA could not be generated from hybrid CA, if characteristic matrix forms a tri-diagonal matrix with all '1' in diagonal positions, and on-diagonal & off-diagonal positions are occupied by "1, 0" or "1, 1, 0" pattern sequence. Corresponding determinant of characteristic matrix is equal to zero.

Example 5.2.4.

Let, R_{11} = <51, 204, 51, 204, 51>, R_{12} = <204, 60, 51, 60, 51>,

R_{13} = <153, 51, 102, 51, 102>, and, R_{14} = <153, 60, 153, 195, 51>.

$$T_{R11=}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

determinant = 1;

capable to produce ELCA = yes;

characteristic polynomial = (x+1).

$$T_{R12=}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

determinant = 1;

capable to produce ELCA = yes;

characteristic polynomial = (x+1).

$$T_{R13=}\begin{bmatrix}1 & 1 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0\\0 & 0 & 1 & 1 & 0\\0 & 0 & 0 & 1 & 0\\0 & 0 & 0 & 0 & 1\end{bmatrix}_{5\times5}$$

determinant = 1;

capable to produce ELCA = yes;

characteristic polynomial = (x+1).

$$T_{R14=}\begin{bmatrix}1 & 1 & 0 & 0 & 0\\1 & 1 & 0 & 0 & 0\\0 & 0 & 1 & 1 & 0\\0 & 0 & 1 & 1 & 0\\0 & 0 & 0 & 0 & 1\end{bmatrix}_{5\times5}$$

determinant = 0;

capable to produce ELCA = no;

Characteristic polynomial = (x+1).

Corollary 5.2.1. Variable length ELCA is generated from hybrid CA, if characteristic polynomial has a form of (x+1)^n, and corresponding determinant of characteristic matrix is equal to one.

Proof:

ELCA cycles are generated using linear rule '153' in null-boundary uniform CA scenario. Next state dependency for rule '153' is on self-cell and right cell only (refer Table 5.2.1). Therefore characteristic polynomial only produces a polynomial (x+1)^n (refer Equation 5.2.9).

(End of proof.)

5.2.3.2. Algebraic Properties

5.2.3.2.1. Algebraic operations, performed on $[n\times n]$ characteristic matrix as reported in Equation 7 and Equation 9, are mentioned as follows.

rank = n;

determinant = 1;

trace = n;

signature = (n,0).

5.2.3.2.2. Algebraic operations, performed on $[n\times n]$ characteristic matrix as reported in Equation 11, are mentioned as follows.

rank = n;

determinant = 1;

trace = n;

not a symmetric matrix, hence no signature.

Example 5.2.5.

Let, R_1 = <51, 51, 51, 51>, R_2 = <204, 204, 204, 204>, R_3 = <153, 153, 153, 153>, and, R_4 = <90, 150, 90, 150>.

$$T_{R1=}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}_{4\times 4}$$

rank = 4;

determinant = 1;

trace = 4;

signature = (4,0).

$$T_{R2=}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}_{4\times 4}$$

rank = 4;

determinant = 1;

trace = 4;

signature = (4,0).

$$T_{R3=}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4}$$

rank = 4;

determinant = 1;

trace = 4.

not a symmetric matrix, hence no signature.

$$T_{R4=}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{4\times 4}$$

rank = 4;

determinant = 1;

trace = 2;

signature = (2, 2).

Theorem 5.2.3. An n-cell CA is responsible for ELCA generation, iff determinant of characteristic matrix is one, and, trace of characteristic matrix is equal to 'n'.

Proof:

Algebraic properties as reported in 3.2.1 and 3.2.2 prove the statement for an n-cell CA.

(End of proof.)

Lemma 5.2.1. An n-cell CA is responsible for fixed length ELCA generation, iff determinant of characteristic matrix is equal to one, and, trace is equal to 'n' and signature is (n, 0).

Proof:

Algebraic properties as reported in 3.2.1 and 3.2.2 prove the statement for an n-cell CA. Illustrations related to Lemma 5.2.1 are in Example 4 (refer R1 and R2 in Example 5.2.5).

(End of proof.)

Lemma 5.2.2. An n-cell CA is responsible for variable length ELCA generation, iff determinant of characteristic matrix is equal to one, and, trace is equal to 'n' and there is no signature for characteristic matrix.

Proof:

Algebraic properties as reported in 3.2.1 and 3.2.2 prove the statement for an n-cell CA. Illustrations related to Lemma 5.2.2 are in Example 4 (refer R3 in Example 5.2.5).

(End of proof.)

Lemma 5.2.3. An n-cell CA is responsible for maximum length CA (MaxCA) generation, iff determinant of characteristic matrix is equal to one, and, trace is equal to '$\lceil \frac{n}{2} \rceil$' and signature of characteristic matrix is ($\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor$).

Proof:

Illustrations related to Lemma 5.2.3 are in Example 5 (refer R4 in Example 5.2.5). MaxCA is generated for this rule set in null-boundary CA.

(End of proof.)

Transition diagrams for Example 5.2.5 have been shown in Fig. 5.2.1.

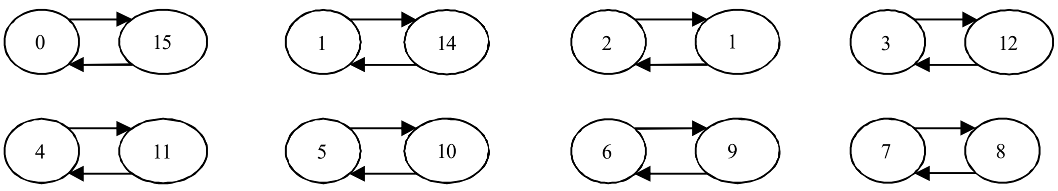


Fig. 5.2.1(a). Transition diagram for <51, 51, 51, 51>

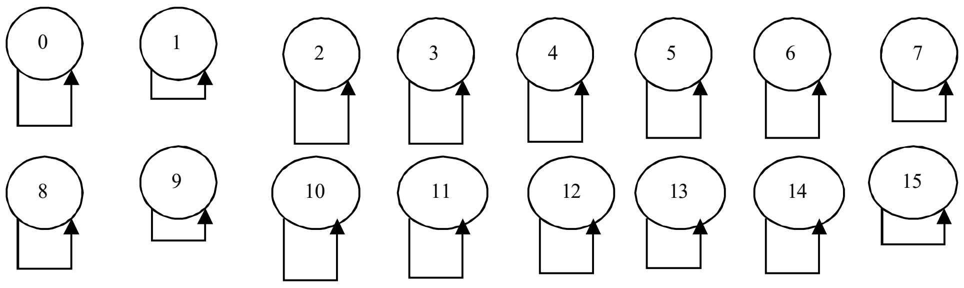


Fig. 5.2.1(b). Transition diagram for <204, 204, 204, 204>

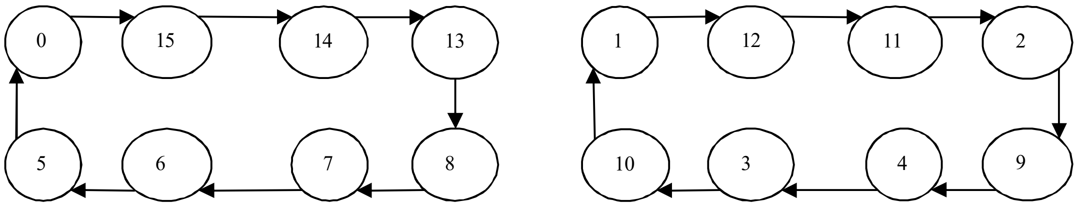


Fig. 5.2.1(c). Transition diagram for <153, 153, 153, 153>

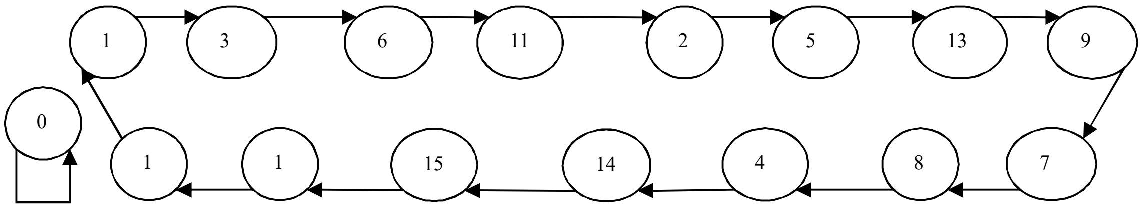


Fig. 5.2.1(d). Transition diagram for <90, 150, 90, 150>
Fig. 5.2.1. Transition diagrams using 51, 204, 153, 90, 150

Eight equal length cycles of length two are shown in Fig. 5.2.1(a). Sixteen equal length cycles of length one are presented in Fig. 5.2.1(b). Two equal length cycles of length eight are shown in Fig. 5.2.1(c). One MaxCA of length fifteen is shown in Fig. 5.2.1(d). A set of necessary and sufficient conditions are achieved using Lemma 1, Lemma 2, and Lemma 3.

Necessary condition for ELCA and MaxCA generation: Determinant of $[n \times n]$ characteristic matrix should be equal to one.

Sufficient condition for ELCA generation: Trace of $[n \times n]$ characteristic matrix should be equal to 'n'.

Sufficient condition for MaxCA generation: Trace of $[n \times n]$ characteristic matrix should be equal to '$\lceil \frac{n}{2} \rceil$'.

Discussions on characteristic matrix of Example 5.2.4 are provided in Example 5.2.6.

Example 5.2.6.

$$T_{R11=} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

rank = 5;

determinant = 1;

trace = 5;

signature = (5, 0);

capable to produce ELCA = yes.

$$T_{R12=} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

rank = 5;

determinant = 1;

trace = 5;

not a symmetric matrix, hence no signature;

capable to produce ELCA = yes.

$$T_{R11=} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

rank = 5;

determinant = 1;

trace = 5;

not a symmetric matrix, hence no signature;

capable to produce ELCA = yes.

$$T_{R11=} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 5}$$

rank = 3;

determinant = 0;

trace = 5;

signature = (3, 0);

capable to produce ELCA = no.

Transition diagrams for Example 5.2.6 are shown in Fig. 5.2.2.

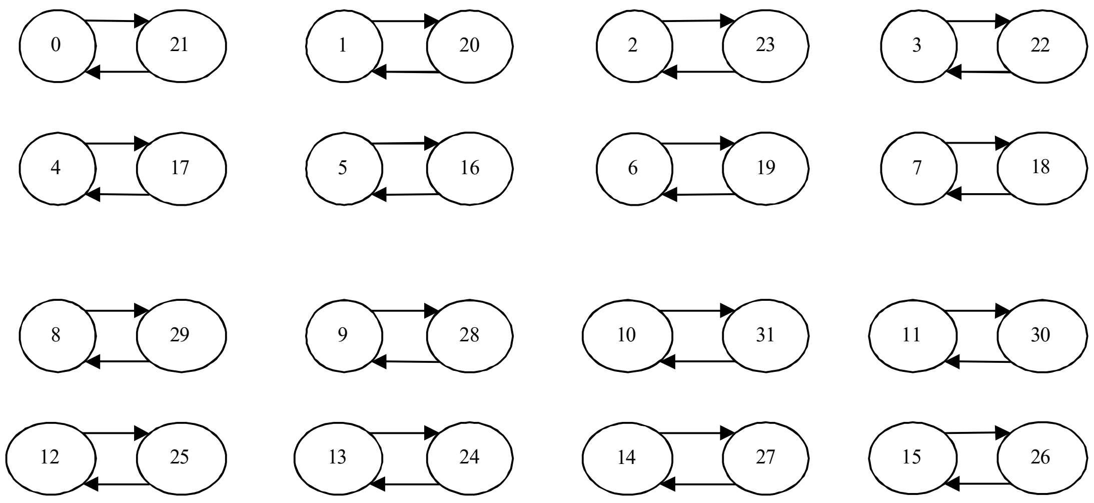


Fig. 5.2.2(a). Transition diagram of R_{11}=<51, 204, 51, 204, 51>

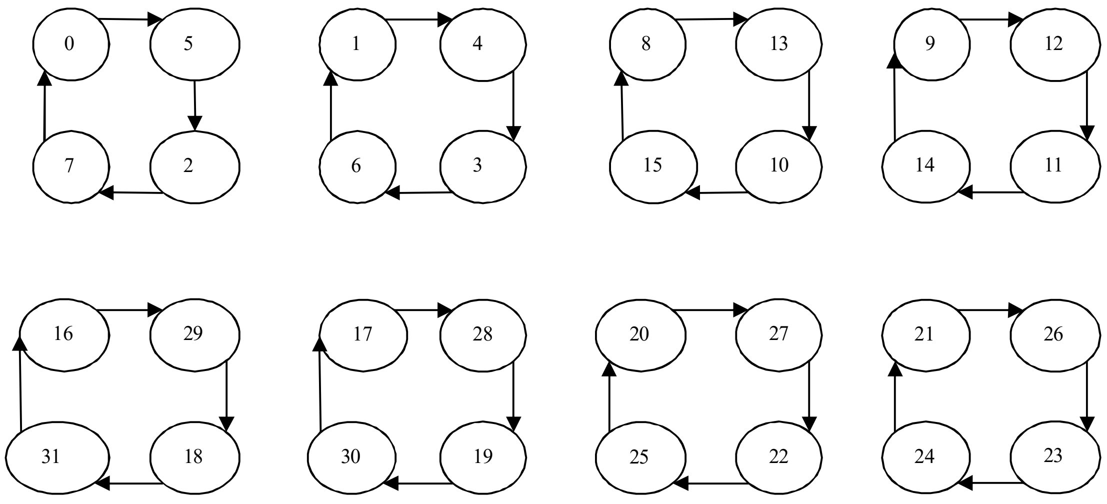


Fig. 5.2.2(b). Transition diagram of R_{12}=<204, 60, 51, 60, 51>

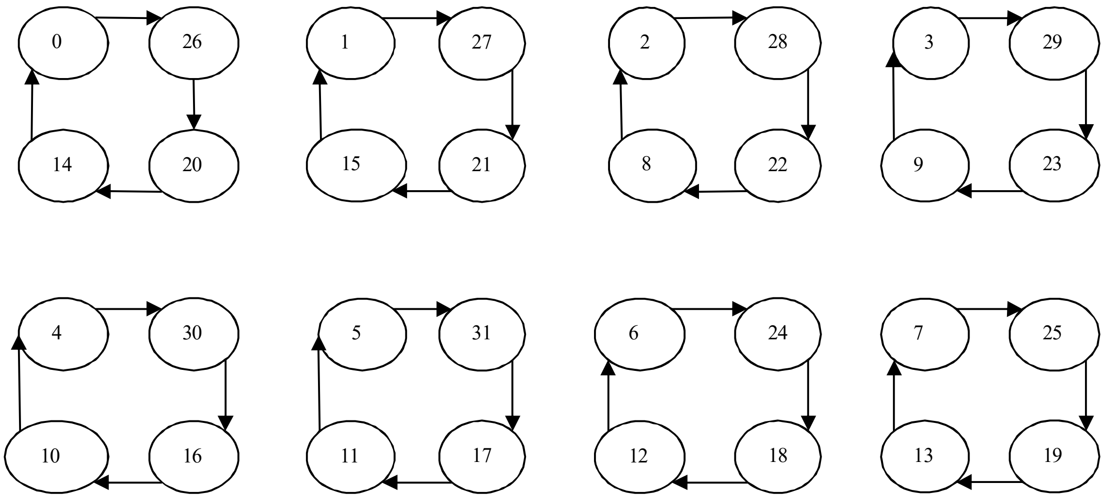


Fig. 5.2.2(c). Transition diagram of R_{13}=<153, 51, 102, 51,102>

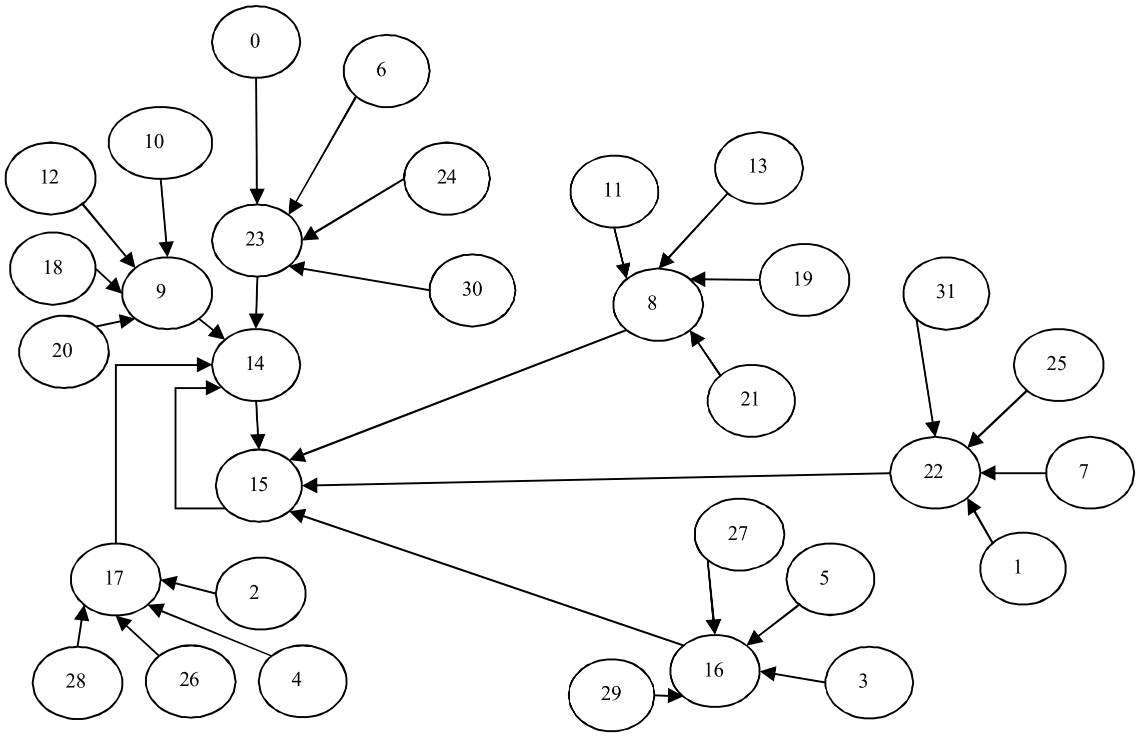


Fig. 5.2.2(d). Transition diagram of R_{14}=<153, 60, 153, 195, 51>
Fig. 5.2.2. Transition diagrams using 51, 204, 60, 153, 102, 195

Equal length cycles are observed in Fig. 5.2.2(a), Fig. 5.2.2(b), and, Fig. 5.2.2(c). No equal length cycle is found in Fig. 5.2.2(d) (refer Theorem 5.2.1, Lemma 5.2.1, Lemma 5.2.2, and, Lemma 5.2.3).

5.2.4. Experimental observations & results

ELCA patterns are generated in simulation. Intel Pentium Dual CPU E2180 @ 2.00 GHz processing support was availed in execution of computer simulation. Equal length cycles generated with ELCA generating linear and complemented linear rules in three-neighborhood null-boundary scenario are reported in Table 5.2.2.

Table 5.2.2. ELCA generation table using a combination of rule “204”, “51” and “153”

Serial Number	**Rule**	**n-cell CA**	
		Uniform	**Hybrid**
1.	204	Capable for producing ELCA	Capable for producing ELCA
2.	51	Capable for producing ELCA	Capable for producing ELCA
3.	153	Capable for producing ELCA	Capable for producing ELCA

Rule “51”, “153” and “204” are capable of generation of ELCA patterns both in uniform and hybrid CA conditions. Cycle lengths in generated ELCA patterns are enlisted in Table 5.2.3, Table 5.2.4 and Table 5.2.5.

Table 5.2.3. ELCA cycle length table for rule “204”

Serial Number	**Cell Size**	**Cycle Length**
1.	3	1
2.	4	1
3.	5	1
4.	6	1
5.	12	1
6.	23	1
7.	63	1

All single length ELCA patterns are achieved in Table 5.2.3. Hence a conclusion is drawn using that all single length ELCA patterns are generated for rule “204” in uniform CA scenario.

Table 5.2.4. ELCA cycle length table for rule "51"

Serial Number	Cell Size	Cycle Length
1.	3	2
2.	4	2
3.	5	2
4.	6	2
5.	12	2
6.	23	2
7.	63	2

All double length ELCA patterns are achieved in Table 5.2.4. Hence a conclusion is drawn that all double length ELCA patterns are generated for rule "51" in uniform CA scenario.

ELCA length is found as varying over different cell sizes for null boundary uniform CA with rule "153". Different cycle lengths have been reported in Table 5.2.5.

Table 5.2.5. ELCA cycle length table for rule "153"

Serial Number	Cell Size range	Minimum number of bits required to represent in binary equivalent(n)	Cycle Length
1.	3	2	4
2.	4→7	3	8
3.	8→15	4	16
4.	16→31	5	32
5.	32→63	6	64

Generalized relationship among cell size of CA and generated ELCA cycle length for rule "153" in uniform CA scenario is reported in Table 5.2.6.

Table 5.2.6. ELCA cycle length calculation table for rule "153"

Cell Size range	Cell Size range (in decimal value)	Minimum number of bits required to represent in binary equivalent(n)	Cycle Length
$(2^n)\rightarrow(2^{n+1}-1)$	$2^n\rightarrow 2^{n+1}-1$	n+1	2^{n+1}

Equation 5.2.15 is presented based on Table 5.2.6.

$$Cycle\ length = 2^{number\ of\ minimum\ bits\ required\ to\ represent\ in\ equivalent\ binary\ form\ for\ given\ cell\ numbers\ in\ decimal} \quad (5.2.15)$$

Results obtained in Table 5.2.2, Table 5.2.3 and Table 5.2.4 are graphically shown in Fig. 5.2.3.

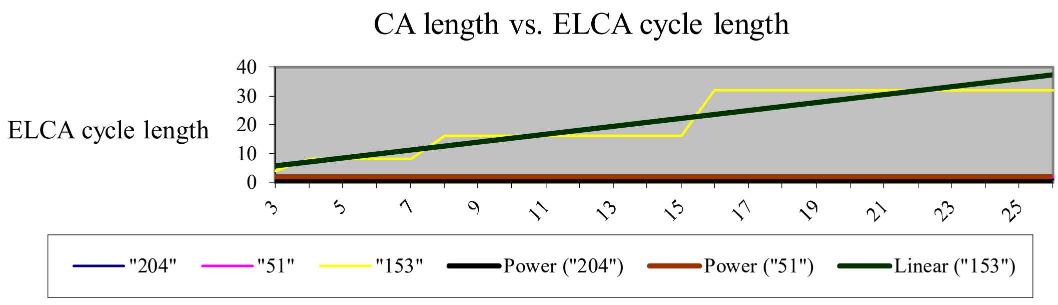


Fig. 5.2.3. ELCA cycle length

ELCA cycle length in uniform CA scenario is shown in Fig. 5.2.3 for rule “204”, “51” and “153”. A trend for corresponding graph is shown with Power Curves (refer Fig. 5.2.3). A linear growth in power curve is achieved for ELCA patterns with constant cycle length (refer Table 5.2.3 and Table 5.2.4). Stepped growth is only been observed for rule “153” (refer Table 5.2.5 and Equation 5.2.15).

Discussions for all other ELCA rules (refer Table 5.2.2 and Table 5.2.3) are reported in Table 5.2.7.

Table 5.2.7. ELCA generation table for other ELCA generating linear rules

CA rules	**n-cell CA**	
	Uniform condition	**Hybrid condition**
60, 85, 90, 102, 105, 150, 165, 170, 195	Not Capable for producing ELCA	Capable for producing ELCA

ELCA generating capability by ELCA generating balanced rules in different CA scenarios (i.e. uniform, and hybrid scenario) are explored in the experiments. Lengths of generated ELCA cycles in different experimental results strengthen the analysis (refer Sub-section 5.2.3).

5.2.5. Summary

A formal analysis on linear CA rules in null boundary scenario is presented for ELCA generation. Algebraic approach is focused in the Sub-section. General form of characteristics matrix, relationship between ELCA cycle and CA size are shown.

5.3. ELCA formulation using Discrete Mathematics

5.3.1. Introduction

Discrete Mathematics [83] is referred to as the study of mathematical configurations that are essentially disconnected rather than continuous in character. Set of finite or infinite number of stuffs is considered for theoretical discrete mathematics. Finite mathematics is applied to parts of discrete mathematics field that deals with finite sets for the physical implementation of the computation. Tessellation concept is introduced for efficient designs. Tessellation is referred as a tiling of plane (flat surface) using one/multiple geometric shapes (tiles) with no overlapping and gap between two consecutive tiles. Aperiodic tiling is referred as a non-periodic tiling with a restricted property that arbitrarily large periodic patches are excluded from concerned tile [84].

Repeating arrangement of points in a graph (tile) is described with the help lattice theory. Lattice is described as an algebra over a non-empty set and a pair of binary operations (“And operation” and “Or operation”) on that non-empty set [85]. Mapping is required for better understanding of computation. Mapping is referred as a relationship among elements of domain and range. Injection, surjection and bijection mapping as found in [86-89] are briefly discussed below.

Injective- *A function* $f: A \rightarrow B$ *is referred as injective (one-to-one) iff each member elements of the 'domain'(A) is exactly mapped with at-most one-member element of the 'range' (B).*

Surjective- *A function* $f: A \rightarrow B$ *is referred as surjective (onto) iff each member elements of the 'domain'(A) is mapped with at-least one-member element of the 'range' (B).*

Bijective- *A function* $f: A \rightarrow B$ *is referred as bijective iff injection and surjection, both properties are satisfied for that function.*

Injection, surjection and bijection as described in [89] are presented in Fig. 5.3.1.

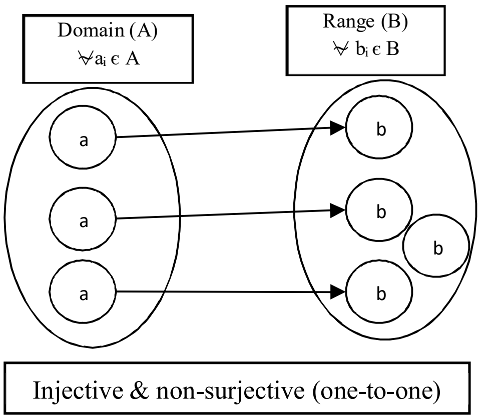


Fig. 5.3.1. (a)

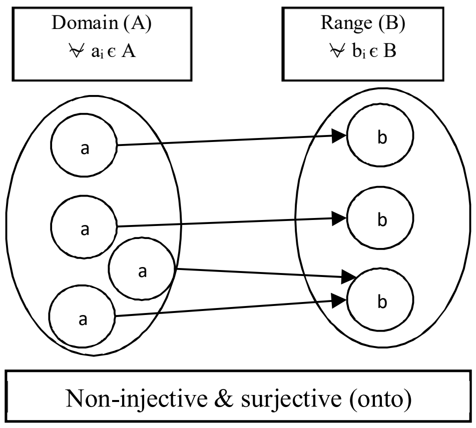


Fig. 5.3.1. (b)

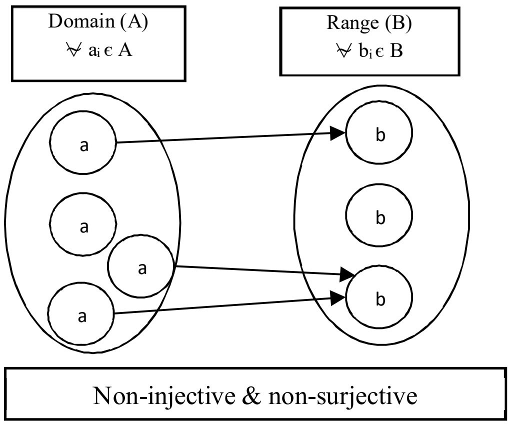


Fig. 5.3.1. (c)

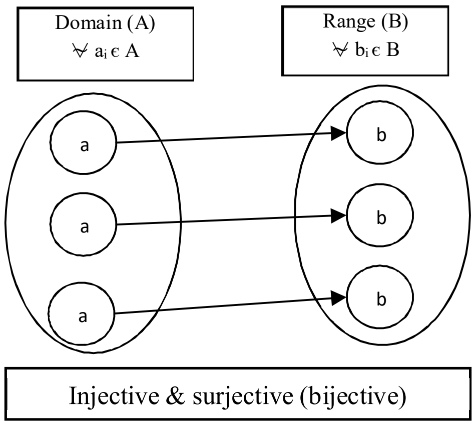


Fig. 5.3.1. (d)

Fig. 5.3.1. Typical diagrammatic illustrations for injection, surjection, projection and bijection.

Lattice- *Lattice in 'R^n' is defined as a discrete subgroup of 'R^n' which spans the real vector space 'R^n'. Every lattice in 'R^n' is generated from a base for the vector space by forming all linear combinations with integer co-efficient. Lattice is also views as a regular tiling of a space by primitive cell.*

Aim of this section is to explore the principles for ELCA generation using FSM and to carry out a thorough mathematical investigation for ELCA characteristics polynomial to investigate randomness generation capacity from ELCA patterns.

Rest of the section is organized as follows: ELCA formulation and analysis using discrete math is in Sub-section 5.3.2; experimental observations are reported in Sub-section 5.3.3 and summary is in Sub-section 5.3.4.

5.3.2. ELCA formulation

CA are expressed as a function of sextuple in Equation 2.1.3 of Sub-section 2.1. ELCA is a special class of group CA. Hence, ELCA is capable of being represented as a function of sextuple [47]. Generated ELCA is capable of being transformed into some finite number of tessellations. Equal number of states is found in each tessellation. Sextuple function for ELCA is discussed in Equation 5.3.1. Equation 5.3.1 is based on the Equation in [47].

$$ELCA = < \Gamma, S, s, s_0, N, \Phi > \qquad (5.3.1)$$

where, finite number 2^m of tessellations of length 2^{n-m} are represented by Γ;

finite set of states ($k = 2$, where possible number of state values are represented by k) is referred to as S and often $S \subset \mathbb{N}$;

output mapping function $s\colon \Gamma \times \mathbb{N} \to S$ produces the state value of cell c_i at the t^{th} discrete time step denoted by $s(c_i, t)$;

initial condition for every cell c_i i.e. $s(c_i, 0) = s_0(c_i)$ is assigned by function $s_0\colon \Gamma \to S$; every cell c_i is mapped to a finite sequence $N(c_i) = (c_{ij})_{j=1}^{|N(c_i)|}$ by neighborhood function $N\colon \Gamma \to \bigcup_{P=1}^{\infty} \Gamma^P$ and $|N(c_i)|$ is the number of all distinct cells c_{ij} ;

$\Phi = (\phi_i)_{i \in \mathbb{N}}$ is a family of functions $\phi_i\colon S^{|N(c_i)|} \to S$ where each ϕ_i is responsible for the dynamics of cell c_i, i.e., $s(c_i, t+1) = \phi_i(\tilde{s}(N(c_i), t))$, as $(\tilde{s}(N(c_i), t)) = (\tilde{s}(N(c_i), t))_{j=1}^{|N(c_i)|}$.

Bijection mapping is present in ELCA state transition diagrams (refer Fig. 5.3.3(d)). Hence FSM representation for ELCA is possible. Number of cycles generated in ELCA state space is discussed in Theorem 5.3.1 and output mapping function for ELCA generating FSM is discussed in Theorem 5.3.2.

Motivation of Theorem 5.3.1 is to show that ELCA structure follows bijection mapping.

Theorem 5.3.1: An output mapping function employing Bijection mapping is responsible for generation of ELCA.

Proof:

Output mapping function $s\colon \Gamma \times \mathbb{N} \to S$ as described in Equation 5.3.1 is responsible for generation of transitions in form of group CA. Injective [86] and surjective [87] mapping both are simultaneously found in output mapping function, $s\colon \Gamma \times \mathbb{N} \to S$. A "one-to-one" mapping is considered as a special case of "one-to-many" mapping. Thus, surjective (onto) mapping is found as a collection of multiple injections (one-to-one). Hence, existence of bijection mapping [88, 89] in output mapping function of ELCA has been proved.

(End of proof.)

Equation 5.3.4 is obtained from Theorem 5.3.1.

$$P(\frac{B}{C_1}) = P(I) \cap P(\frac{S}{C_2}) \qquad (5.3.4)$$

where, ‘C_1’ and ‘C_2’ are conditions; ‘P’ is referred as probability; ‘B’ is referred as bijection; ‘S’ is referred as surjection; and ‘I’ is referred as injection.

Observation 5.3.2. Lattice Isomorphism [90] is found in ELCA transition mapping “$s: \Gamma \times \mathbb{N} \rightarrow S$”.

A pair of Necessary and Sufficient Conditions are found from Theorem 5.3.1.

Necessary Condition for ELCA generation: *Characteristics polynomial for ELCA is a form of minimal polynomial along with a complete bijection mapping in ELCA transition function.*

Sufficient Condition for ELCA generation: *Characteristics polynomial for ELCA is a form of recursive primitive polynomial.*

Randomness is assured for a CA generated pattern if characteristics polynomial for that CA is a primitive polynomial. Primitive characteristics polynomial is found for MaxCA. Hence degree of randomness is found in MaxCA cycle generated with a unique combination of rules “90” and “150” in null-boundary condition only. No primitive characteristics polynomial is ever possible for combination of rules “90” and “150” in periodic-boundary condition. Hence MaxCA is not found in periodic boundary CA [5].

Motivation of Theorem 5.3.2 is to show the presence of primitive polynomial or primitive recursive polynomial in ELCA characteristic polynomial.

Theorem 5.3.2. If characteristics polynomial for ELCA is primitive polynomial or primitive recursive polynomial, then randomness is assured in generated pattern.

Proof:

The characteristics polynomials for uniform and hybrid ELCAs have been reported as “(1+x)” and “$(1+x)^m$”. It has been found that the characteristics polynomial of ELCA is of the form of a minimal polynomial (Sub-section 5.2).

A primitive function$f(x)$ as defined in [91] is shown in Equation 5.3.5.

$$f(x) = a_0 + a_1x^1 + a_2x^2 + \cdots . + a_nx^n \quad \ldots\ldots\ldots\ldots\ldots.. \text{(5.3.5)}$$

where, $a_0, a_1, a_2, \ldots, a_n$ are integer coefficient, the greatest common divisor of $a_0, a_1, a_2, \ldots, a_n$ is one, and $f(x)$ is a primitive function. Additionally, if $f(x)$ and $g(x)$ are two primitive polynomials, $f(x)g(x)$ is also a primitive polynomial.

The characteristics polynomial for ELCA is "(1+x)" and "$(1+x)^n$".

Assume $f_1(x) = (x + 1)$ and $g_1(x) = (x + 1)^n$.

The greatest common divisor of all coefficients of $f_1(x)$ is one.

Hence, $f_1(x)$ is a primitive polynomial.

$g_1(x)$ is represented as $g_1(x) = (f_1(x))^n$. Hence, $g_1(x)$ is also a primitive recursive polynomial.

Application of primitive recursive, reducible polynomials in generation of pseudo-random sequences can also be found in [92-94].

Therefore, all ELCAs are capable of producing pseudo-random sequences.

(End of proof.)

5.3.3. Experimental results

Experimental results are reported in Sub-section 5.3.3.1, Sub-section 5.3.3.2 and Sub-section 5.3.3.3. ELCA and MaxCA generated patterns are collected from computer simulation to carry out investigations as reported in Sub-section 5.3.3.1 and Sub-section 5.3.3.2.

5.3.3.1. Performance evaluation

Randomness in generated patterns for MaxCA and ELCA are graphically shown in Fig. 5.3.2.

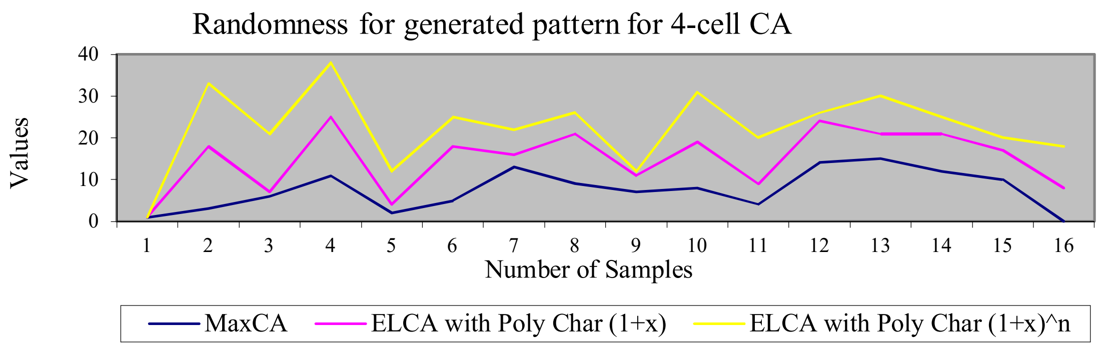


Fig. 5.3.2(a).

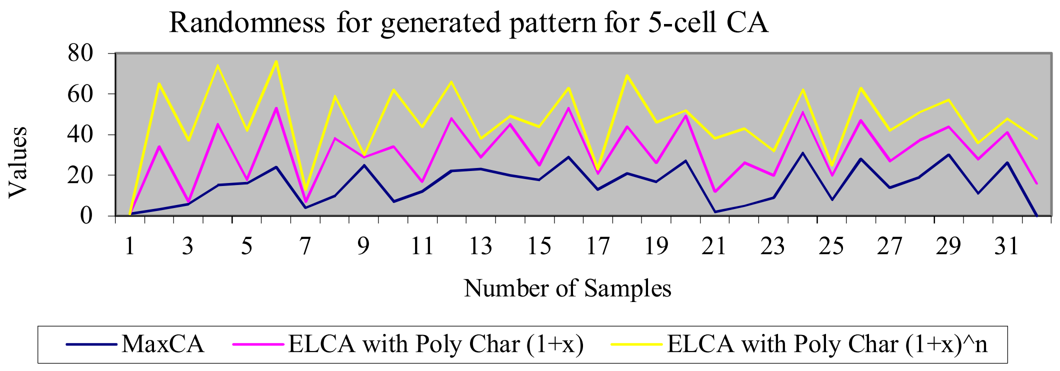


Fig. 5.3.2(b)

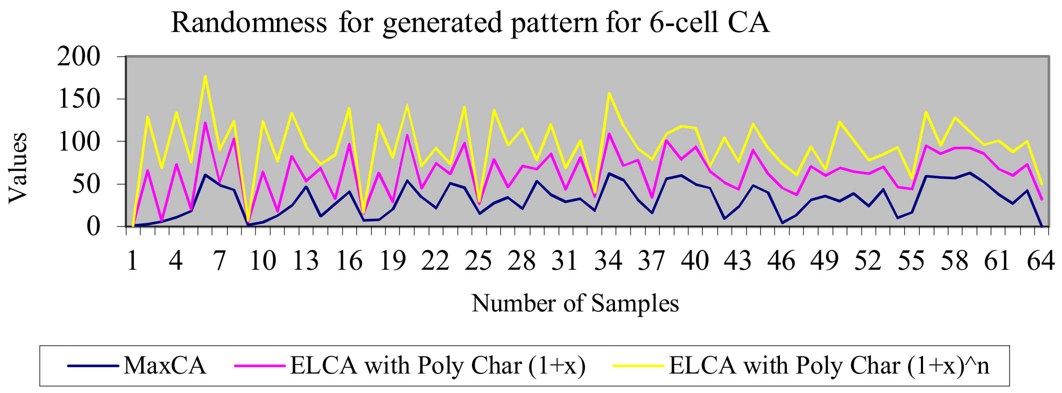


Fig. 5.3.2(c).

Fig. 5.3.2. Randomness in different CA generated patterns

Pseudo-random patterns in Fig. 5.3.2 were produced using a unique combination of rule ‘90’ and ‘150’ for MaxCA, rule ‘51’ for ELCA with characteristics polynomial ‘(1+x)’ and rule ‘153’ for ELCA with characteristics polynomial ‘$(1+x)^n$’. MaxCA patterns were generated using <90, 150, 90, 150>, <150, 150, 90, 90, 150>, <90, 150, 90, 150, 90, 150> respectively for Fig. 5.3.2(a), Fig. 5.3.2(b) and Fig. 5.3.2(c). ELCA patterns were achieved in Fig. 5.3.2(a), Fig. 5.3.2(b) and Fig. 5.3.2(c) using uniform CA of rule ‘51’ and uniform CA of rule ‘153’ respectively. It is evident from Fig. 5.3.2 that degrees of randomness of the MaxCA and the ELCA for cell size 4, 5, and 6 are competitive.

5.3.3.2. Real time activities

Randomness of patterns generated by MaxCA and ELCA PRNGs are also measured using statistical analysis tool called Runs Test [95]. The Wald–Wolfowitz runs test (or simply runs test) is named after Abraham Wald and Jacob Wolfowitz. It is a non-parametric statistical test that checks a randomness hypothesis for a data sequence. It can be used to test the hypothesis that the elements of the sequence are mutually independent [96]. The Runs Test results in terms of p-values and corresponding conclusions obtained in [95] have been reported in Table 5.3.1.

Table 5.3.1. RUNs Test Results

Serial number	**Cell (n)**	**MaxCA**		**ELCA with characteristics polynomial (1+x)**		**ELCA with characteristics polynomial $(1+x)^n$**	
		p-value	***Conclusion***	***p-value***	***Conclusion***	***p-value***	***Conclusion***
1.	4	0.13292	Little or no real evidence against randomness	0.00015	Very strong evidence against randomness	0.30239	Little or no real evidence against randomness
2.	5	0.42976	Little or no real evidence against randomness	almost zero	Very strong evidence against randomness	0.35965	Little or no real evidence against randomness
3.	6	0.5	Little or no real evidence against randomness	almost zero	Very strong evidence against randomness	0.5	Little or no real evidence against randomness

If the p-value is very small (almost equal to zero) or very high (almost equal to 1), the hypothesis that the data set is random is rejected. The p-value of the single length cycle for MaxCA in null boundary CA scenario is zero or very small. The p-value of the ELCA with characteristics polynomial (1+x) is also either zero or very small. But the p-values for MaxCA and ELCA with characteristics polynomial $(1+x)^n$ are in the range to prove existence randomness [95, 96]. Hence the degree of randomness for ELCA cycles are comparable to MaxCA cycles.

5.3.4. Summary

Discrete mathematics based analysis on ELCA generation is reported in this section. Existence of bijection mapping in ELCA generation ensures a close-fitting with FSM. Characteristics polynomials for ELCA are primitive polynomials or primitive recursive polynomials. Hence randomness in ELCA generated patterns is confirmed.

5.4. Energy efficient task-pull scheduling using ELCA

5.4.1. Introduction

Efficient scheduling policy play an important role in reliable distributed and parallel computing. Load balanced task-pull scheduling for maximum number of available processing units in distributed computing is important in efficient uses of processing units, faster result processing and optimized energy consumption. Task scheduling to computing devices or processors is a NP-complete problem [97, 98]. Parallel programming approaches are followed in distributed computing. Algorithmic parallelism, geometric parallelism and processor framing approaches have been defined in [99] depending on their characteristics. Static task allocation is implemented in case of algorithmic parallelism and geometric parallelism. Algorithmic parallelism is maintained for decomposition using pipeline of processors as needed. Data is transferred through computing elements. Bigger tasks are decomposed into several independent task modules in case of geometric parallelism allocated to available computing units. Each computing device executes the subsets of total processed data sets. Each isolated processor is responsible for execution of same task with different initial data in processor farm. Static and dynamic scheduling of tasks are being used for optimization of performance in parallel computing. A balance between communication time and computation time is required for ensuring efficient usage of processing resources [99].

Sequential and parallel CA based scheduling algorithms were demonstrated by researchers [31-34]. Parallel scheduling [31], heuristics based near optimum solutions for scheduling problems [32-34] were described to enhance parallel and distributed computing. Genetic Algorithm (GA) based CA rule discovery for scheduling policy were reported in [33, 34]. In our studies, we have not found a simple, cost effective CA based equally populated job scheduling, which may be beneficial towards cost-effective scheduling in distributed computing. Hence a simple and cost-effective ELCA based scheduler is presented in this section.

Rest of the section is organized as follows: proposed work is described in Sub-section 5.4.2; experimental results and analysis are reported in Sub-section 5.4.3; finally, summary is in Sub-section 5.4.4.

5.4.2. Proposed approach

ELCA based design for energy efficient task-pull scheduling in distributed computing is proposed in present research. Different tasks and available processors are mapped with ELCA to obtain load balanced scheduling in distributed environment.

Consider, a typical scenario for distributed task scheduling. Let, there exists ‘P’ number of independent task modules of similar complexities to be computed over ‘Q’ number of available processors. ‘Q’ numbers of available processors and ‘P’ numbers of independent task modules are mapped with the ‘M’ numbers of equal length cycles of length ‘N’ generated by ELCA as described in Equation 3.1.1. Illustration of proposed approach is as follows.

Total ‘2^m’ (let M) numbers of equal length cycles of length ‘2^{n-m}’ (let N) are found for an n-cell CA by Equation 3.1.1. All independent ‘P’ task modules are being grouped together by mapping with ‘M’ numbers of equal length cycles of length ‘N’. ‘M’ numbers of task modules are assigned to available ‘Q’ numbers of processors/computing units. The state number of any equal length cycle of ELCA is mapped with the task number (p_i). Task modules are being assigned randomly to the available processors and resulting task scheduling is having the advantage of equal load (equal task module) distribution to available processors. Proposed scheduling policy is applicable for both sequential and parallel execution. Mathematical expressions for task pull allocation scheme are shown in following Equations.

Number of task modules is described in Equation 5.4.1.

$$P = \sum_{i=0}^{n} P_i \qquad (5.4.1)$$

Number of available processors is described in Equation 5.4.2.

$$Q = \sum_{i=0}^{n} Q_i \qquad (5.4.2)$$

By Equation 1.1,

$$2^n = 2^m * 2^{n-m}$$

Equation 5.4.2 is expressed as Equation 5.4.3.

$$2^n = M * N \qquad (5.4.3)$$

where, $M=2^m$ and $N=2^{n-m}$.

Now mapping for task scheduling is performed as described in Equation 5.4.4 and Equation 5.4.5.

$$P \in N \qquad (5.4.4)$$

$$Q \in M \qquad (5.4.5)$$

Example 5.4.1.

An n-cell CA for n=4 has been decomposed into some equal length smaller cycles. In our illustrated example as described in Fig. 5.4.1(a) is decomposed into 2 smaller cycles of length 8; or, as described in Fig. 5.4.1(b) is decomposed into 8 smaller cycles of length 2. Pattern generation for task scheduling in this scenario is followed as referred in Fig. 5.4.1. Fig. 5.4.1 is based on Null Boundary condition. The synthesis of this example to generate ELCA is achieved for a combination <195, 195, 195, 195> for Fig. 5.4.1(a) and <51, 51, 51, 51> for Fig. 5.4.1(b).

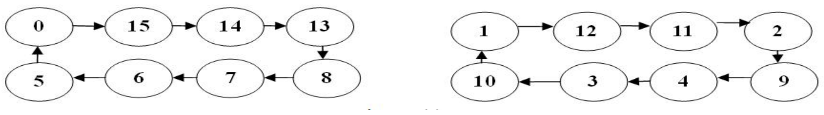


Fig. 5.4.1(a)

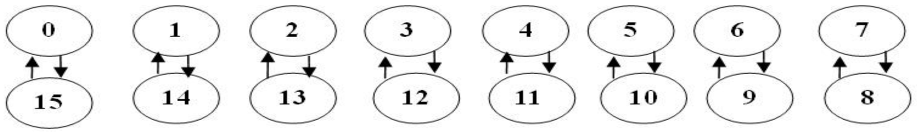


Fig. 5.4.1(b).
Fig. 5.4.1.
Fig. 5.4.1(a). Proposed 2 equal length cycles of cycle size 8 for < 153, 153, 153, 153 >
Fig. 5.4.1(b). Proposed 8 equal length cycles of cycle size 2 for < 51, 51, 51, 51 >

Let there exist two processing units in a distributed computing system with sixteen independent tasks. All these task modules are grouped together in two task-pulls as described in Fig. 5.4.1(a). Different task-pulls are signified by each equal length cycles and the randomly scheduled independent task modules in every task-pull are signified by corresponding i= 0, 1, 2... etc. Generated task modules are allocated to the two different processing units. A random and load balanced task-pull is assigned to the available processing units.

The scenario as illustrated in Fig. 5.4.1(b) that all those sixteen task modules are capable for equally load distributed condition among eight processing units in a distributed computing environment. All these tasks-pools are capable of being processed by individual processing units simultaneously or, one after another. Now consider a special case where nine number of tasks are to be allocated in available two processors. Two equal length task-pulls of length four are formed and allocated to available processing units and remaining single task is assigned to any of the available processing units. Proposed flowchart of ELCA based task scheduling policy is presented in Fig. 5.4.2.

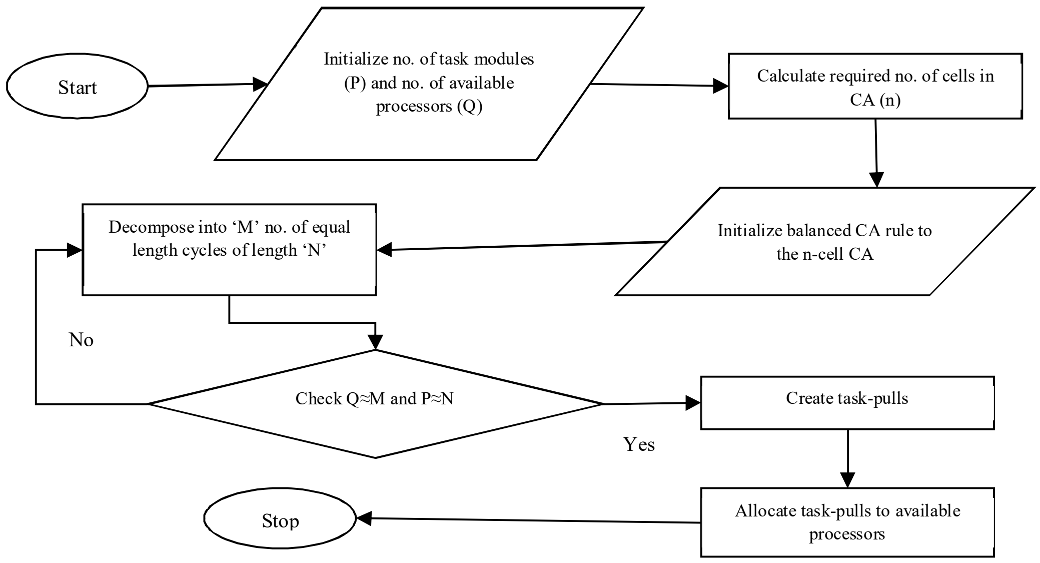

Start
Initialize no. of task modules (P) and no. of available processors (Q)
Calculate required no. of cells in CA (n)
Initialize balanced CA rule to the n-cell CA
Decompose into 'M' no. of equal length cycles of length 'N'
No
Check Q≈M and P≈N
Yes
Create task-pulls
Allocate task-pulls to available processors
Stop

Fig. 5.4.2. Proposed flowchart for task scheduling in distributed computing environment

ELCA based task scheduling policy for distributed computing environment is designed using Algorithm 5.4.1, Algorithm 5.4.2 and Algorithm 5.4.3.

Algorithm 5.4.1. CA size_computation

Input: *Number of tasks (P), number of processors (Q)*

Output: *CA size (n)*

Step 1: Start

Step 2: Initialize the number of tasks (P), number of processors (Q)

Step 3: Compute minimum value of 'm' such that 2m ≈ Q

Step 4: Compute minimum value of 'n' such that 2n-m ≈ P

Step 5: Stop

Algorithm 5.4.2. Task_pull_generation_using_ELCA

Input: *CA size (n), balanced rules*

Output: *'M' numbers of equal length task-pull of length 'N'*

Step 1: Start

Step 2: Initialize the number of n-cell CA to generate random numbers using n-cell CA

Step 3: Initialize balanced CA rule to all the cells for generation of equal length task-pull

Step 4: Decompose the cell number (n) into equal numbers (m) such that $2^n = 2^m*(2^{n-m})$ i.e., 'm' number of equal length cycles of (n-m) length) for $n \geq 1$ and m=1,2,3…….(n-1)

Step 5: Schedule task-pull using appropriate scheduling procedure

Step 6: Stop

Algorithm 5.4.3. Task-pull_scheduling

Input: *Task-pulls (M), available processors (Q)*

Output: *task-pull scheduling*

Step 1: Start

Step 2: If M<=Q then randomly allocate 'M' number of task-pulls into 'Q' number of available processors else follow Step 3.

Step 3: Scheduling error and follow Algorithm 5.4.2

Step 4: Process 'M' number of task-pulls at 'M' number of processors sequentially or simultaneously

Step 5: Stop

An efficient approach for optimum energy consumption in concerned distributed computing environment is found from proposed ELCA based scheduling approach. Discussions are included in Example 5.4.2 to report about optimum energy consumption.

Example 5.4.2.

Let a distributed computing environment where number of independent task modules is 'P' and available number of processors is 'Q'; all are following same architecture. 'M' numbers of equal length task cycles of length 'N' are produced by ELCA based scheduler. The equal length task-pulls are then allocated to available 'Q' number of processors using Equation 5.4.4 and Equation 5.4.5.

Let energy consumption for a processing unit in active state is 'E_{actv}' and in idle state is 'E_{idle}' respectively. 'Active' state is achieved by processing unit while it is in the process of execution for a task module else it is in 'idle' state, if the processor is in powered on mode. Energy consumption in powered off mode is zero and it has already been established that $E_{actv} > E_{idle} > 0$ (zero) [34]. Energy consumption for every processing unit is E_{actv} for every individual task module. So, total energy consumption in the distributed computing system is given as following relation.

E_{total}=no. of processors in active state (M)* no. of task modules in task-pull (N)*energy consumption for one processor in active state (E_{actv}) + no. of processors in idle state (Q-M) *energy consumption for one processor in idle state (E_{idle}).

Thus,

$$E_{total} = M * N * E_{actv} + (Q - M) * E_{idle} \tag{5.4.6}$$

Maximum power consumption for that distributed system for equally load distributed condition in an ideal scenario where all available processors are scheduled with ELCA generated task schedules is achieved from Equation 5.4.6. Thus, Equation 5.4.7 is achieved form Equation 5.4.6.

$$E_{\max} = M * N * E_{actv} \tag{5.4.7}$$

where (Q-M) = 0;

Here no processing unit is in idle state. Hence, Equation 5.4.8 is achieved from Equation 5.4.7.

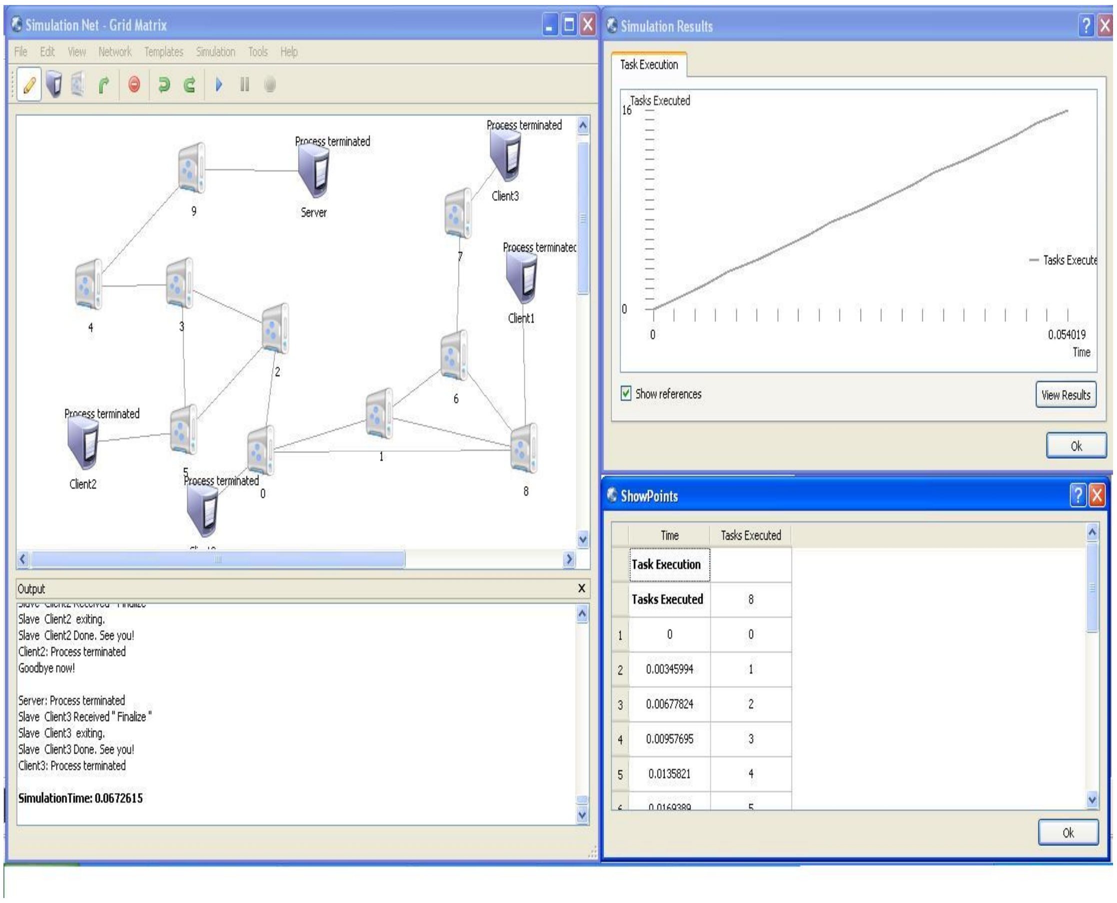
$$E_{\max} = P * E_{actv} \tag{5.4.8}$$

where P=M*N.

An efficient use of available processing units is found for ELCA based scheduler. Equal length task modules are allocated to maximum number of available processors for an equal load distribution.

5.4.3. Experimental observations & result analysis

Different task-pulls were simulated with Grid Matrix Simulator [100, 101] based on SimGrid Toolkit [102]. Simulations were performed in client-server architecture for different number of task modules and varying numbers of processing units. Computation size task 5000 and communication size of task 1000 were used in simulation. Simulation set up is shown in Fig. 5.4.3.



```
Output
Slave  Client2  exiting.
Slave  Client2 Done. See you!
Client2: Process terminated
Goodbye now!

Server: Process terminated
Slave  Client3 Received " Finalize "
Slave  Client3  exiting.
Slave  Client3 Done. See you!
Client3: Process terminated

SimulationTime: 0.0672615
```

	Time	Tasks Executed
	Task Execution	
	Tasks Executed	8
1	0	0
2	0.00345994	1
3	0.00677824	2
4	0.00957695	3
5	0.0135821	4
6	0.0169389	5

Fig. 5.4.3. Screenshot of task-pull execution in client-server architecture in GridMatrix

Different scenarios for equal task allocations in client-server architecture are shown in Fig. 5.4.4. Different numbers of tasks are assigned to available fixed numbers of computing units.

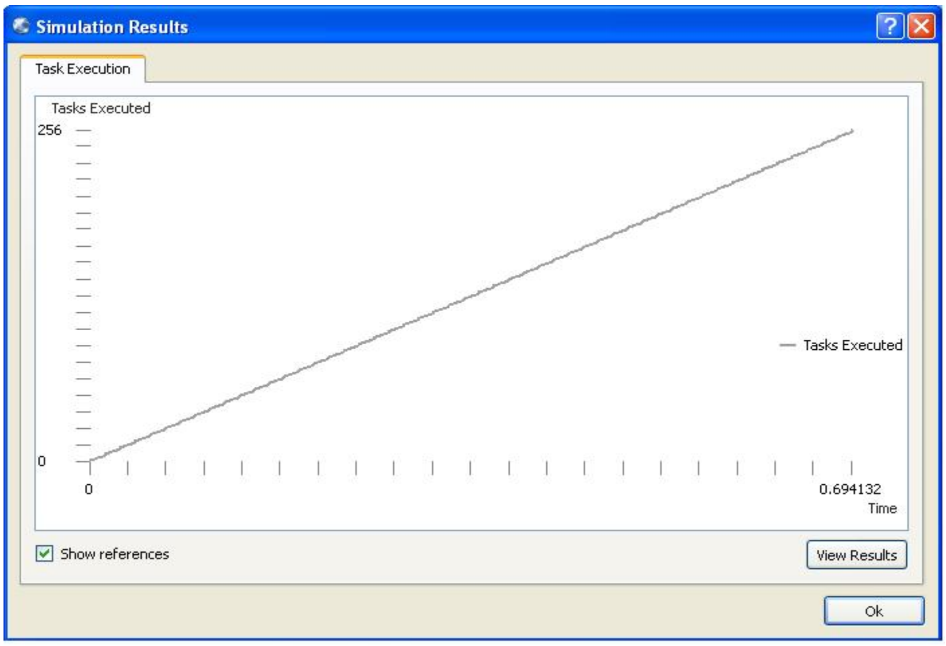


Fig. 5.4.4(a).

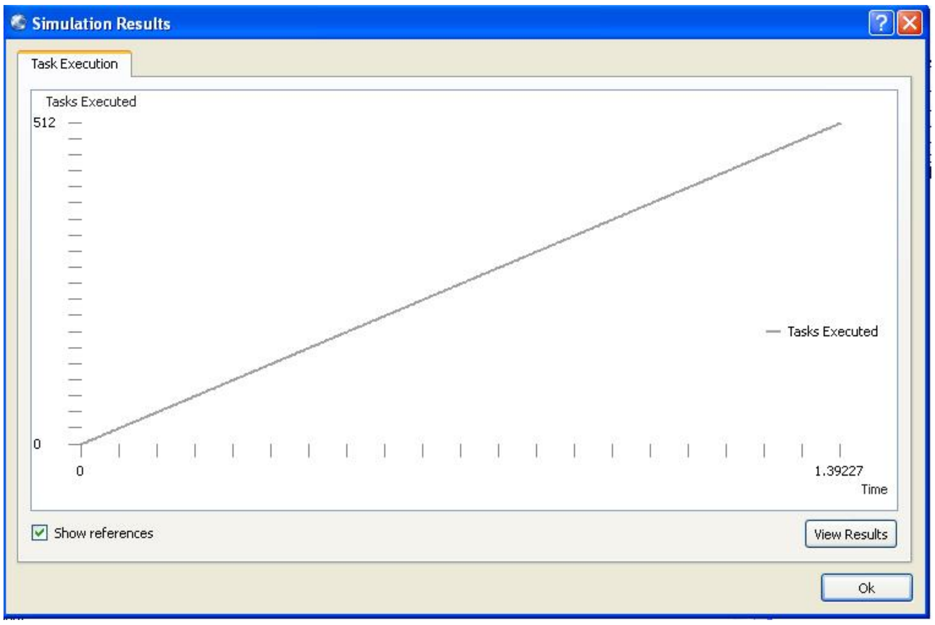


Fig. 5.4.4(b).

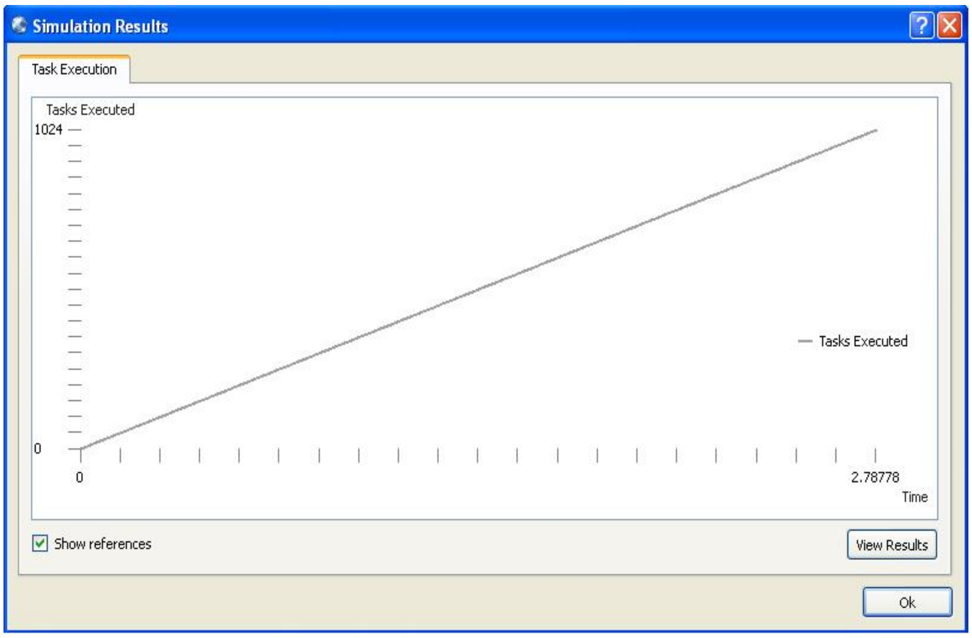


Fig. 5.4.4(c).
Fig. 5.4.4.
Fig. 5.4.4(a). Simulation result for total task size of 256 tasks
Fig. 5.4.4(b). Simulation result for total task size of 512 tasks
Fig. 5.4.4(c). Simulation result for total task size of 1024 tasks

A linear growth in the computation time is achieved in Fig. 5.4.4 for complete execution of the varying number of task pulls by a fixed number of processing units. Execution time in curves increases linearly in Fig. 5.4.4 (a), Fig. 5.4.4(b) and Fig. 5.4.4(c).

Time consumed for a balanced task load distribution (fixed length task pull of length 256) to different number of processors is reported in Table 5.4.1. Graphical representation of information achieved from Table 5.4.1, is shown in Fig. 5.4.5.

Table 5.4.1. Simulation time

No. of processing units	Time
2	0.641374
4	0.777577
16	0.933235

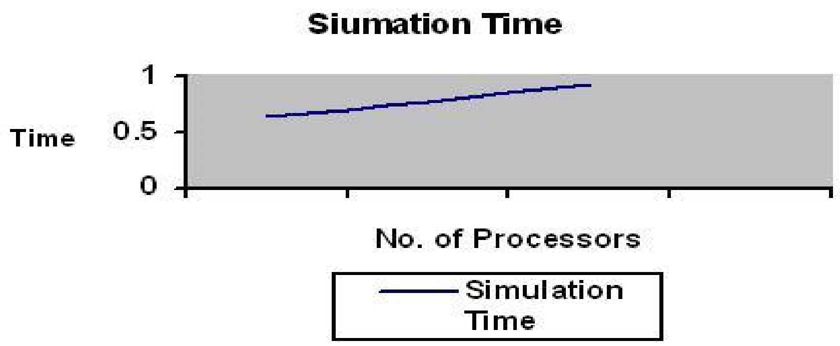


Fig. 5.4.5. Task pull compliance graph

A relationship between time and number of used processing units is found in Fig. 5.4.5. Detailed explanation of Fig. 5.4.5 is as follows.

The relation between numbers of used processors and length of task-pull is in Equation 5.4.9.

$$NoOfPocess\ ors(M) \propto \frac{1}{TaskPullLe\ ngth(N)} \qquad (5.4.9)$$

Relation between task execution time and task-pull length is followed in Equation 5.4.10.

$$TaskExecut\ \ ionTime\ (T_e) \propto TaskPullLe\ \ ngth\ (N) \qquad (5.4.10)$$

Hence, task execution time (T_e) increases as the task pull length (N) increases.

Relation between task-pull scheduling time and numbers of used processors is followed in Equation 5.4.11.

$$TaskPullSchedulingTime(T_s) \propto NoOfProcessors(M) \qquad (5.4.11)$$

Hence, task scheduling time (T_s) increases as the number of available processing units (N) increases. More task scheduling time (T_s) is required to schedule balanced tasks to all available processing units (N).

5.4.4. Summary

Random task assignment is found in ELCA based equally populated task-pulls. All these generated task-pulls are efficiently utilizing the maximum number of available processing units.

CHAPTER 6

EQUAL LENGTH CELLULAR AUTOMATA DYNAMICS

6.1. Dynamics of ELCA rules

6.1.1. Introduction

Studies with CA explore the mathematical properties of system dynamics for any complex system. Langton's λ-parameter (activity parameter) and Z-parameter are used discuss the system dynamics [103-105]. Langton's λ-parameter is used to determine the probability that a CA cell will have next state as one [104].

Let 'n' number of transitions present for quiescent state (an arbitrary space "s ϵ ∑") 's_q' in the transition function 'Δ'. Let the remaining "(K^N-n)" transitions in 'Δ' be filled by pinching randomly and uniformly over the other "K-1" states in "∑ - s_q". Thus, Langton's λ-parameter is calculated as in Equation 6.1.1 [104].

$$\lambda = \frac{\left(K^N - n\right)}{K^N} \qquad (6.1.1)$$

If "$n = K^N$", then all the transitions in the rule table will be to the quiescent state 's_q' and $\lambda = 0.0$. Most homogeneous distribution is indicated by $\lambda = 0.0$. If $n = 0$, then there will be no transitions to 's_q' and $\lambda = 1.0$. When all states are represented equally in the rule table, 'λ' is defined as in Equation 6.1.2. Most homogeneous rule table has been described by Equation 6.1.2.

$$\lambda = 1.0 - 1/K \qquad (6.1.2)$$

Range of 'λ' for different dynamic behavior as discussed in [104, 105] is presented in Table 6.1.1.

Table 6.1.1. Typical 'λ' values and observed system dynamics

Serial	λ- value	Conclusion
1.	0.40	Dynamical activity is collapsing down onto periodic configurations.
2.	0.45	Dynamical activity is at a balance point between collapse and expansion.
3.	0.50	Large fluctuations are observed in the area covered by dynamical activity; eventual collapse of the dynamics is found due to fluctuations.
4.	0.55	Dynamical activity effectively settles down to chaotic behavior.
5.	0.60	Chaotic dynamic activities are observed.
6.	0.65	Typical Chaotic activities are observed.

Observed system dynamics is graphically reported in Fig. 6.1.1. Dominate behavior of the rules is changed from homogeneous fixed point to inhomogeneous fixed points, periodic, complex spatial-temporal dynamics and chaotic dynamics as "λ-parameter value" changes from “0.0” to “0.5”. For “λ-parameter value” higher than “0.5” is found in reverse order of the mentioned behavior of “λ-parameter value” for the range “0.0” to “0.5” [104, 105].

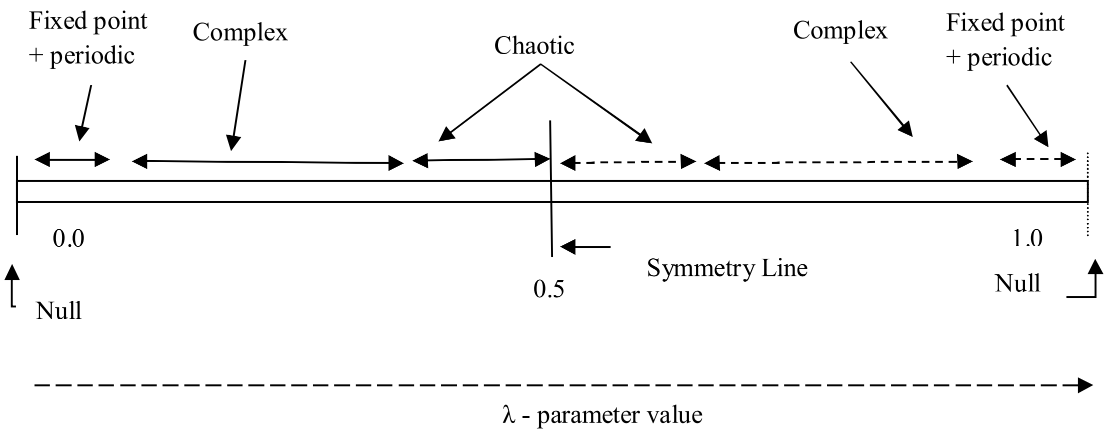


Fig. 6.1.1. Typical scale for λ-parameter values and system behaviors

Aim in this section is to calculate the λ-parameter value for analysis of ELCA dynamics.

Rest of the section is organized as follows: λ-parameter value for ELCA generating rules are computed in Sub-section 6.1.2; analysis of system dynamics based computed λ-parameter values is in Sub-section 6.1.3 and summary is in Sub-section 6.1.4.

6.1.2. Computation of λ-parameter for ELCA rule

Linear and non-linear rules are explored for ELCA pattern generation (Sub-section 6.1). Detailed characteristics for linear rules are explored using matrix algebraic tools (Sub-section 6.1). No such matrix algebraic computation has been reported for non-linear rules. A generalized study towards the CA dynamics based the non-linear and linear rules are focused in this Sub-section. The computation of the “λ-parameter” is emphasized for dynamics analysis of ELCA rules. “λ-parameters” are computed using Equation 6.1.1. Equal number of 0’s and 1’s is present in the binary representation of each ELCA generating rules (Sub-section 6.1). Hence density of 1’s or 0’s in any ELCA generating rule is “0.5”.

Example 6.1.

Let an arbitrary ELCA generating rules as enlisted in Table 4. Say rule “204”. Binary representation of rule “204” is “11001100”. λ-parameter value for “11001100” is 4/8 (=0.5).

Value of λ-parameter for each ELCA rule is in the “symmetry line” (refer Fig. 6.1.1) [104, 105]. Hence chaotic property is observed ELCA rules.

Theorem 6.1.1. All ELCA rules are positioned in the "symmetry line" of the λ-parameter based global dynamics of CA.

Proof:

There exists equal number of 0's and 1's in any ELCA rule. Hence λ-parameter value for any ELCA rule is always ".0.50" (Refer Equation 8.5). All ELCA rules are placed in the symmetry line of the global dynamics CA (refer Fig. 6.1.1).

(End of proof.)

Corollary 6.1.1. Chaotic nature is exhibited by the ELCA rules.

Proof:

All ELCA rules are situated in the global dynamics CA (refer Fig. 6.1.1) by Theorem 6.1.1. Symmetry line in Fig. 6.1.1 is at the center position of chaotic behavior region. Hence an inherent chaotic characteristic is found in the dynamics of the ELCA rules.

(End of proof.)

6.1.3. Analysis of ELCA dynamics

It is found in "λ-parameter" values for ELCA rules is "0.50" and thus chaotic dynamics is present in ELCA generated patterns. Degree of randomness as a measurement of chaos present in ELCA generated patterns has been reported in Table 6.2. Statistical analytical approach based on the RUNs test [95] is presented in Table 6.1.2.

Table 6.1.2. Degree of randomness as a measurement of inherent chaotic dynamics for ELCA

Serial Number	**Rule vector**	**ELCA pattern**	
		p-value	***Conclusion***
1.	<51, 51,51,51>	0.30239	Little or no real evidence against randomness
2.	<153, 153, 153, 153, 153>	0.35965	Little or no real evidence against randomness
3.	<51, 204, 51, 204>	0.00015	Very strong evidence against randomness
4.	<51, 204, 51, 204, 51>	Almost zero	Strong evidence against randomness

Competitive dynamics are achieved for ELCA with respect to MaxCA. Reported CA rules "90" and "150" responsible for generation of MaxCA pattern and possess same "λ-parameter" value equals to "0.50". Hence similar chaotic dynamics is expected for ELCA and MaxCA rules (Fig. 6.1.1). Competitive degree of randomness for CA patterns as found in [95] is described in Table 6.1.1. Test results in Sub-section 6.1.2 confirm the existence of almost same chaotic nature in both CA patterns.

6.1.4. Summary

Computed λ-parameters for ELCA rules is “0.50” which is same as the λ-parameter values for MaxCA composing CA rules. All ELCA rules are situated in the symmetry line of CA global dynamics Fig. (refer Fig. 6.1.). Thus inherent chaotic characteristics are found for ELCA dynamics. It is found that ELCA dynamics is similar to MaxCA dynamics.

6.2. OTP based authentication using ELCA

6.2.1. Introduction

Distributed system is susceptible to a variety of security threats mounted by intruders. Brief discussion on different security and authentication aspects in distributed environment is reported below.

Inherent threats in data communications have been categorized in [110] as "Host Compromise" and "Communication Compromise". Combination of hardware and software has been suggested as a solution to host compromise [116]. On the other hand, "Communication Compromise" refers to the threat associated with message communication. Loss of privacy in conversations (eavesdropping), arbitrary modification(s) to received message and replay of old messages are common security hazards faced during transmission of message. Authentication is accepted as a solution to this problem. Authentication means an appropriate arrangement of identification and verification. Three different authentication categories are discussed in the literature: (i) authentication of content (ii) authentication of origin (iii) authentication of general identity [116].

OTP scheme has been suggested as an efficient and simple solution for message authentication in distributed and cloud computing. OTP is known to be a time synchronized random password which is used in authentications; this password can be used at most once. The list of generated passwords is stored in the client and the server. A single password from the list is used in a sequential manner for every distinct session. OTP protocol is known to be an effective measure for security in cloud to defend against "Replay Attacks" and "Dictionary Attacks" [116]. Several authentication techniques have been presented by researchers [42-44, 111-116]. Smart phone-based authentication has been described in [114]. Besides, multiple factor-based authentication [112], attachment of small piece of high-performance trusted hardware with untrusted units [115], anonymous node ID Assignment [116] are some of the different approaches adopted now-a-days to enhance authentication. Uses of CA in authentication is also very popular among researchers. M. Mukherjee et al. have presented CA based authentication [42]. J. C. Jeon et al. have presented non-group CA based one time password (OTP) authentication scheme in wireless networks [43]. R. Yampolskiy et al. have presented CA rule 30 based data security and authentication [44]. In our studies, we have not found a simple, cost effective generation of equally populated OTP sets. Hence a simple and cost-effective generation of equally populated OTP sets using ELCA has been introduced, which may be advantageous towards low-cost authentication in distributed computing.

Rest of the section is organized as follows: Preliminary concepts of authentication are briefly described in Sub-section 6.2.2; ELCA based proposed design is in Sub-section 6.2.3; Results are discussed in Sub-section 6.2.4; finally, summary is in Sub-section 6.2.5.

6.2.2. Preliminary concepts of authentication

During authentication, one node (sender) is verified by another node (receiver). In basic authentication, encrypted messages independently generated by the sender and the receiver using a symmetric key have to match [106]. Building a complex crypto system for authentication purposes is not typical in distributed computing [106].

6.2.3. Proposed design

Linear CA rules "51", "204" and "153" are suggested for OTP-based authentication in distributed computing. Selection of these CA rules is based on application suitability, Langton's λ-parameter, and the characteristics polynomials of enlisted CA rules. Primitivity is found in ELCA characteristic polynomial, which is an important criterion for generation of randomness. Enlisted linear CA rules can be used in both uniform and hybrid scenarios. The characteristics matrices ELCA is explored in Sub-section 5.2. Use of rectangular matrix in cryptography and authentication is discussed in [107]. Formation of a concatenated matrix from two square matrices is illustrated in Equation 6.2.1. Matrix concatenation is used to examine the characteristics of ELCA rules of high order.

$$T_{n_{204\ or\ 51}} = \begin{bmatrix} 1\,0\,0 \cdots 0 \\ 0\,1\,0 \cdots 0 \\ 0\,0\,1 \cdots 0 \\ \vdots \\ 0\,0\,0 \ldots 1 \end{bmatrix}_{nxn} \circ T_{n_{153}} = \begin{bmatrix} 1\,1\,0 \cdots 0 \\ 0\,1\,1 \cdots 0 \\ 0\,0\,1 \cdots 0 \\ \vdots \\ 0\,0\,0 \ldots 1 \end{bmatrix}_{nxn} \qquad (6.2.1)$$

Horizontal concatenation "(T (Hori))" is defined in Equation 6.2.2.

$$T(Hori)_{n_{(204\ or\ 51) o\ 153}} = \begin{bmatrix} 1\,1\,0 \ldots 0\,1\,1\,0 \ldots 0 \\ 0\,1\,0 \ldots 0\,0\,1\,1 \ldots 0 \\ 0\,0\,1 \ldots 0\,0\,0\,1 \ldots 0 \\ . \\ . \\ . \\ 0\,0\,0 \ldots 1\,0\,0\,0 \ldots 1 \end{bmatrix}_{nx2n} \qquad (6.2.2)$$

and $Rank_{T(hori)_{n(204\ or\ 51) o153}} = n.$

Vertical concatenation "(T (Vert))" is defined in Equation 6.2.3.

$$T(Vert)_{n\,(204\;or\;51)o153} = \begin{bmatrix} 1\;1\;0\ldots0 \\ 0\;1\;0\ldots0 \\ 0\;0\;1\ldots0 \\ . \\ . \\ . \\ 0\;0\;0\ldots1 \\ 1\;1\;0\ldots0 \\ 0\;1\;1\ldots0 \\ 0\;0\;1\ldots0 \\ . \\ . \\ . \\ 0\;0\;0\ldots1 \end{bmatrix}_{2nxn} \qquad (6.2.3)$$

and $Rank_{T(Vert)_{n\,(204\;or\;51)o153}} = n.$

Ranks of two rectangular matrices in Equation 6.2.2 and Equation 6.2.3 possess a value equal to the minimum value between row and column values of these matrices. Hence maximal ranks are obtained for these two matrices [108]. Maximal rank matrices might be used for data security and authentication applications [109]. Therefore, selected ELCA rules are suitable for usage in OTP-based authentication.

6.2.3.1. Proposed ELCA classification

Proposed classification design has been introduced in Fig. 6.2.1.

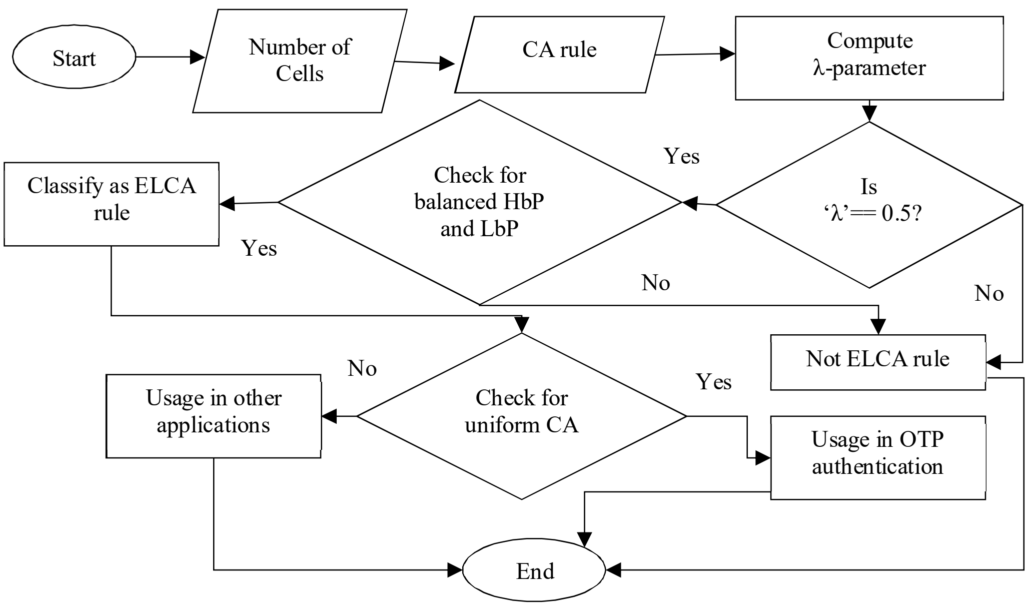


Fig. 6.2.1. ELCA classification procedure

The number of CA cells and CA rules are initialized first. The number of cells is dependent on the total number of OTPs required. The λ-parameter value is computed for each CA rule. It is checked whether the λ-parameter value is equal to 0.5. Once the λ-parameter value for ELCA rule is equal to 0.5, it is checked for balanced 'HbP' & 'LbP' conditions. Thereafter, it is checked whether the ELCA rules are in uniform CA condition. Use of ELCA rules in uniform condition is suggested for OTP-based authentication because the design becomes simple and requires less storage in such cases. Use of hybrid CA configuration for generation of ELCA is also acceptable.

6.2.3.2. Proposed design for OTP authentication

A simple design ensuring pseudo-random generated patterns is proposed. Different uniform CA scenarios are discussed briefly in Table 6.2.1.

Table 6.2.1. Uniform CA scenario for ELCA generation

Serial Number	CA rule	Binary representation	Rule Type	Cell Size (n)	Length of Cycles
1.	204	11001100	Linear	3→7	1
2.	51	00110011	Linear	3→7	2
3.	153	10011001	Linear	3	4
				4→7	8

The flowchart of the proposed OTP-based authentication design is depicted in Fig. 6.2.2. Algorithm 6.2.1, Algorithm 6.2.2, and Algorithm 6.2.3 are designed to facilitate cost effective OTP set generation and authentication in OTP-based communication for 'n' number of messages. An 'n' cell CA is proposed for OTP generation of 'n' number of message(s). Rule selection for uniform CA is based on Table 6.2.1.

It is found form Table 6.2.1 that a flexibility of selection of number set members in OTP set is available for ELCA based OTP authentication design. System flowchart of the proposed OTP authentication design has been depicted in Fig. 6.2.2.

Algorithm 6.2.1. OTP_Set_Generation

Input: *Number of cell (n), ELCA generating balanced CA rule (R)*

Output: *Number of OTP sets (m) containing equal number of OTP members*

Step 1: Start

Step 2: Initialize number of cells and ELCA generating rule (refer Table 11.1) in each cell

Step 3: Decompose into 'm' number of cycles of equal lengths

Step 4: Call Algorithm 8.2

Step 5: Communicate OTP lists to sender and receiver

Step 6: End

Algorithm 6.2.2. OTP_Assignment

Input: *Equal length cycles generated in Algorithm 1 (m), Number of messages (m)*

Output: *OTP set with equal number of members for every message*

Step 1: Start

Step 2: Assign 'm' number of equal length cycles to 'm' number of messages

Step 3: Assign states of each cycle as OTP for concerned message

Step 4: End

Algorithm 6.2.3. OTP_Authentication

Input: *Lists of OTP for communication from Algorithm 11.1*

Output: *Authenticated sender approval for communication*

Step 1: Start

Step 2: Use OTP in a sequenced manner one at a time

Step 3: If OTP is validated by receiver with time synchronization then follow Step 4
else follow Step 2

Step 4: Successful authentication of Sender

Step 5: Establish communication

Step 6: End

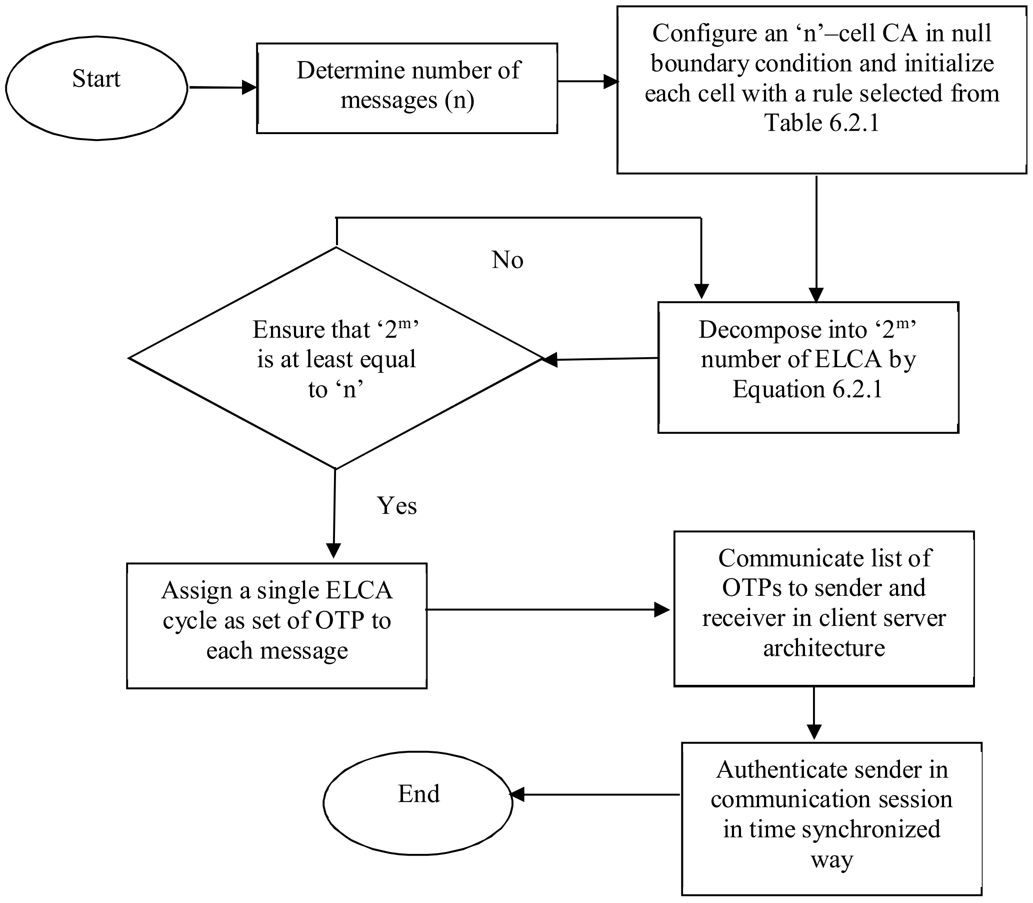

Start
Determine number of messages (n)
Configure an 'n'–cell CA in null boundary condition and initialize each cell with a rule selected from Table 6.2.1
No
Ensure that '2^m' is at least equal to 'n'
Decompose into '2^m' number of ELCA by Equation 6.2.1
Yes
Assign a single ELCA cycle as set of OTP to each message
Communicate list of OTPs to sender and receiver in client server architecture
End
Authenticate sender in communication session in time synchronized way

Fig. 6.2.2. Proposed flowchart of ELCA based OTP authentication

Cost effective OTP set generation is the objective of the Algorithm 6.2.1. The primary concern in the proposed ELCA based system is that at least the same number of OTP sets of equal population size is generated with reference to the number of messages participating in communication. Each OTP is valid for a time stamp; i.e., each OTP should be used for session authentication within a specific period. It is referred to as time synchronization in the Fig. 6.2.2 and the Algorithm 6.2.3. The OTP set so produced is assigned to each message. Efficiency in authentication of the sender in distributed computing has been obtained based on the method of message number dependent OTP set generation.

In the present work, de-correlation between generated ELCA cycles is not considered as no low cost and VLSI compatible CA model to generate multiple equal length OTP sets is found.

Example 6.2.1.

Let a scenario where three number of messages are needed to communicate between sender and receiver. Four number of OTP set containing two random OTPs in each set are generated (refer Fig. 6.2.3). State number of the state is assigned as generated random OTP.

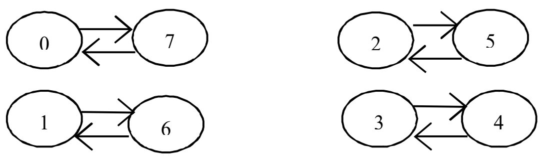


Fig. 6.2.3. Four ELCA structures for <51, 51, 51>

Generation of cost effective and equally populated ELCA based random OTP sets have been achieved as a final outcome in usage of the combined form of two approaches (refer Fig. 6.2.1 and Fig. 6.2.2). Unique OTP set is assigned to each message. Efficiency in authentication of sender for message communication in distributed computing has been achieved based on the method of message number dependent OTP set generation. De-correlation between generated ELCA cycles were not considered as no low cost and VLSI compatible CA model was found which is capable to produce multiple equal length OTP sets.

6.2.4. Experimental results

Degree of randomness for ELCA generated pattern and comparison with MaxCA pattern are described in detail in Sub-sections 5.3 and 6.1. Competitive degree of randomness is found for ELCA patterns with respect to the MaxCA patterns. Detailed comparison of cost effectiveness for ELCA based approach as presented in Sub-section 4.1 with reference to the space and time complexity, is presented again in Table 6.2.2.

Table 6.2.2. Complexity comparison

Complexity	Analysis
Space	Space complexity for both ELCA and MaxCA are O(n) as total number of an n-cell CA and number of states are unchanged. So, space requirement is the same for both cases.
Time	Time Complexity to generate ELCA cycles is $\sum O(m_i)$, where 'm' denotes length of cycle and 'i' denotes number of ELCA cycle; time complexity of to generate MaxCA cycles is O(n), where 'n' denotes length of the MaxCA cycle. As per Equation 2, length of ELCA cycle is lesser than length of the MaxCA cycle. Thus, time requirement for one single cycle generation in ELCA is less than the time requirement for a complete cycle generation in MaxCA.

The comparison of space and time complexities of the ELCA and the MaxCA in Table 6.2.2 suggests that the time complexity is less for ELCA while the space complexity for both ELCA and MaxCA remains similar.

6.2.5. Summary

Cost efficiency, quality randomness and design flexibility in set of OTPs are found in ELCA based set of OTP generation. Linear CA rule based uniform CA design in three neighborhood null boundary condition is capable of easy implementation of modular arithmetic, which is an essential requirement in cryptographic & authentication system.

CHAPTER 7

CONCLUSIONS AND FUTURE SCOPE

The contributions in different chapters of this thesis are summarized in this chapter. Major objectives of the current research are to explore the characteristics and properties of Equal Length Cellular Automata (ELCA), inherent cost efficiencies as a pseudo-random number generator (PRNG) and potential applications in Distributed Computing. Design of CA based alternative PRNG using ELCA and reliability assessment of distributed computing, ELCA based alternative design for BIST applications along with SUT design are presented in Chapter 3. Performance comparison among different random number generators (RNGs) and a CA based stress testing model targeting distributed computing are presented in Chapter 4. ELCA generating complete rule set exploration, detailed analysis of ELCA generating linear rules along with its representation in Discrete Mathematics and one potential application of ELCA model in energy efficient task-pull allocation are presented in Chapter 5. ELCA system dynamics and ELCA based classifier design for OTP based authentication in distributed computing are described in Chapter 6. Complete investigation, synthesis and analysis of ELCA along with potential applications of ELCA model for several distributed computing applications are described in this entire research.

Focus of the research is to provide detailed study of ELCA and potential applications of ELCA based cost effective model towards several distributed applications.

Important aspects of the researches reported in this dissertation may be extended for the benefit of research community. Probable future works are as follows.

- Programmable CA (PCA) based implementation of ELCA,
- Z-parameter analysis of ELCA rules,
- Phase shift analysis of ELCA,
- ELCA based complete cryptosystem.

References

1. Wolfram, S. (1986). Theory and applications of cellular automata (Vol. 1). Singapore: World scientific.

2. Chaudhuri, P.P., Chowdhury, D.R., Nandi, S., & Chattopadhyay, S. (1997). Additive Cellular Automata Theory and Applications (Vol. 1). John Wiley & Sons.

3. Chandy, K. M., Kiniry, J., Rifkin, A., & Zimmerman, D. (1998). A framework for structured distributed object computing, Parallel Computing, Journal of Parallel Computing-Special Issue on applications, 24(12-13), 1901-1922.

4. Distributed Computing. http://distributedcomputing.info/

5. Nandi, S., Kar, B. K., & Chaudhuri, P. P. (1994). Theory and applications of cellular automata in cryptography. IEEE transactions on computers, 43(12), 1346-1357.

6. Ganguly, N., Sikdar, B. K., Deutsch, A., Canright, G., & Chaudhuri, P. P. (2003). A survey on cellular automata.

7. Ghosh, S., Maiti, N. S., Pal Chaudhuri, P., & Sikdar, B. K. (2011). On invertible three neighborhood null-boundary uniform cellular automata. Complex Systems, 20(1), 47.

8. Ghosh, S., Bachhar, T., Maiti, N. S., Mitra, I., & Chaudhuri, P. P. (2010, September). Theory and application of equal length cycle cellular automata (ELCCA) for enzyme classification. In International Conference on Cellular Automata (pp. 46-57). Springer Berlin Heidelberg.

9. Aguiar, I., & Severino, R. (2015). Two Elementary Cellular Automata with a New Kind of Dynamic. Complex Systems, 24(2), 113-125.

10. Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2015). An Analysis of Equal Length Cellular Automata (ELCA) generating Linear Rules for Applications in Distributed Computing. Journal of Cellular Automata, 10.

11. Mitra, A., & Teodorescu, H. N. (2016). Detailed Analysis of Equal Length Cellular Automata with Fixed Boundaries. Journal of Cellular Automata, 11.

12. Zio, E. (2005). Solving advanced network reliability problems by means of cellular automata and Monte Carlo sampling. Reliability Engineering & System Safety, 89(2), 219-226.

13. Zio, E., Podofillini, L., & Zille, V. (2006). A combination of Monte Carlo simulation and cellular automata for computing the availability of complex network systems. Reliability Engineering & System Safety, 91(2), 181-190.

14. Canizes, B., Soares, J., Vale, Z., & Khodr, H. M. (2012). Hybrid fuzzy Monte Carlo technique for reliability assessment in transmission power systems. Energy, 45(1), 1007-1017.

15. Mitra, A., & Kundu, A. (2012, September). CA based cost optimized PRNG for Monte-Carlo simulation of distributed computation. In Proceedings of the CUBE International Information Technology Conference (pp. 332-337). ACM.

16. Kokolakis, I., Andreadis, I., & Tsalides, P. (1997). Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators. Microprocessors and Microsystems, 20(10), 643-658.

17. Das S., Sikdar B. K., Chaudhuri P. P., (2004) Nonlinear CA Based Scalable Design of On-Chip TPG for Multiple Cores, Asian Test Symposium, Taiwan.

18. Das S., Rahaman H., Sikdar B. K, (2005) Cost Optimal Design of Nonlinear CA Based PRPG for Test Applications, IEEE 14th Asian Test Symposium, India.

19. Jamuna, S., & Agrawal, V. K. (2011). Implementation of BIST structure using VHDL for VLSI circuits. International Journal of Engineering Science and Technology, 3(6).

20. Mitra, A., & Kundu, A. (2012). Cost optimized approach to random numbers in cellular automata. Advances in Computer Science, Engineering & Applications, 609-618.

21. Mitra, A., & Kundu, A. (2012). Cost optimized design technique for pseudo-random numbers in cellular automata. International Journal of Advanced Information Technology, 2(3), 21-36.

22. Mitra, A., & Kundu, A. (2014). Cost Optimized Random Sampling in Cellular Automata for Digital Forensic Investigations. Computational Intelligence in Digital Forensics: Forensic Investigation and Applications: Studies in Computational Intelligence, 555, 79-95.

23. Mitra, A., Kundu, A., & Das, C. (2014). Random number generators: performance comparison of ELCA and MaxCA. CSI transactions on ICT, 2(2), 117-127.

24. Mitra, A., Kundu, A., & Das, C. (2014, February). Cost effective PRNG using ELCA: A BIST application. In Automation, Control, Energy and Systems (ACES), 2014 First International Conference on (pp. 1-6). IEEE.

25. Cukier, M., Chandra, R., Henke, D., Pistole, J., & Sanders, W. H. (1999). Fault injection based on a partial view of the global state of a distributed system. In Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on (pp. 168-177). IEEE.

26. Trodhandl, C., & Weiss, B. (2008). A concept for hybrid fault injection in distributed systems. Windsor: TAIC PART Publishing.

27. Hsu, I., Gallagher, A., Le, M., & Tamir, Y. (2010). Using virtualization to validate fault-tolerant distributed systems. In Int. Conf. on Parallel and Distributed Computing and Systems (pp. 210-217).

28. Ulrich, A. W., Zimmerer, P., & Chrobok-Diening, G. (1999). Test architectures for testing distributed systems. In Proceedings of the 12th International Software Quality Week.

29. Ulrich, A. W., & Chanson, S. T. (1996). An approach to testing distributed software systems. In Proceedings of the 15th International Symposium on Protocol Specification, Testing, and Verification, Warsaw, Poland.

30. Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2014). A novel design with Cellular Automata for System-Under-Test in Distributed Computing. Journal of Convergence Information Technology, 9(6), 55.

31. Seredynski, F., & Zomaya, A. Y. (2002). Sequential and parallel cellular automata-based scheduling algorithms. IEEE Transactions on Parallel and Distributed Systems, 13(10), 1009-1023.

32. Laghari, M. S., & Khuwaja, G. A. (2012). Scheduling techniques of processor scheduling in cellular automaton. In Proc. Int. Conf. on Intelligent Computational Systems (pp. 96-100).

33. Laghari, M. S., & Khuwaja, G. A. (2012, January). Processor Scheduling on Parallel Computers. In Proc. Int. Conf. on Computer, Electrical, and Systems Sciences, and Engineering, Abu Dhabi.

34. Agrawal, P., & Rao, S. (2012, March). Energy-aware scheduling of distributed systems using cellular automata. In Systems Conference (SysCon), 2012 IEEE International (pp. 1-6). IEEE.

35. Mitra, A., Kundu, A., & Chattopadhyay, M. (2014, December). Energy Efficient Task-Pull Scheduling Using Equal Length Cellular Automata in Distributed Computing. In Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference of (pp. 40-45). IEEE.

36. Fischer, P. C. (1965). Generation of primes by a one-dimensional real-time iterative array. Journal of the ACM (JACM), 12(3), 388-394.

37. Mazoyer, J., & Terrier, V. (1999). Signals in one-dimensional cellular automata. Theoretical Computer Science, 217(1), 53-80.

38. Umeo, H., & Kamikawa, N. (2002). A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fundamenta Informaticae, 52(1-3), 257-275.

39. Umeo, H., & Kamikawa, N. (2003). Real-time generation of primes by a 1-bit-communication cellular automaton. Fundamenta Informaticae, 58(3-4), 421-435.

40. Umeo, H., Miyamoto, K., & Abe, Y. (2010, November). A construction of smallest real-time prime generators on cellular automata. In Computer Technology and Development (ICCTD), 2010 2nd International Conference on (pp. 338-342). IEEE.

41. Mitra, A., & Kundu, A. (2013). Cost optimized set of Primes Generation with Cellular Automata for Stress Testing in Distributed Computing. Procedia Technology, 10, 365-372.

42. Mukherjee, M., Ganguly, N., & Chaudhuri, P. (2002). Cellular automata based authentication (CAA). Cellular Automata, 259-269.

43. Jeon, J. C., Kim, K. W., & Yoo, K. Y. (2006). Non-group cellular automata based one time password authentication scheme in wireless networks. Secure Mobile Ad-hoc Networks and Sensors, 110-116.

44. Yampolskiy, R. V., Méndez, J. D. R., & Hindi, M. (2014). Password Protected Visual Cryptography via Cellular Automaton Rule 30. Trans. Data Hiding and Multimedia Security, 9, 57-67.

45. Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2014, August). Dynamics computation of Equal Length Cellular Automata (ELCA) rules. In Electronics Engineering & Computer Science, 2014Emerging Applications of Information Technology (EAIT), 2014 Fifth International Conference of (pp. 214-220). Elsevier.

46. Mitra, A., Kundu, A., Chattopadhyay, M., & Chattopadhyay, S. (2017). A cost-efficient one time password-based authentication in cloud environment using equal length cellular automata. Journal of Industrial Information Integration, 5, 17-25.

47. Baetens, J. M., & De Baets, B. (2010, September). Towards generalized measures grasping CA dynamics. In International Conference on Cellular Automata (pp. 177-187). Springer Berlin Heidelberg.

48. True Random Numbers. http://www.random.org.

49. Wolfram, S. (2008). Wolfram mathematica tutorial collection: random number generation.

50. Ganguly, N., Sikdar, B. K., & Chaudhuri, P. P. (2001). Theory of additive cellular automata. Fundamenta Informaticae, (21), 1001-1021.

51. Ganguly, N., Nandi, A., Das, S., Sikdar, B. K., & Chaudhuri, P. P. (2002, November). An evolutionary strategy to design an on-chip test pattern generator without prohibited pattern set (PPS). In Test Symposium, 2002.(ATS'02). Proceedings of the 11th Asian (pp. 260-265). IEEE.

52. Tirthapura, S., & Woodruff, D. P. (2011, September). Optimal random sampling from distributed streams revisited. In International Symposium on Distributed Computing (pp. 283-297). Springer Berlin Heidelberg.

53. Hortensius, P. D., McLeod, R. D., & Card, H. C. (1989). Parallel random number generation for VLSI systems using cellular automata. IEEE Transactions on Computers, 38(10), 1466-1473.

54. Wolfram, S. (1986). Random sequence generation by cellular automata. Advances in applied mathematics, 7(2), 123-169.

55. Robert G. Brown. dieharder: A Random Number Test Suite, 2006a. http://www.phy. duke.edu/~rgb/General/dieharder.php. C program archive dieharder, version 1.4.24.

56. Das, S., Kundu, A., & Sikdar, B. K. (2004, November). Nonlinear CA based design of test set generator targeting pseudo-random pattern resistant faults. In Test Symposium, 2004. 13th Asian (pp. 196-201). IEEE.

57. Das, S., Kundu, A., Sikdar, B. K., & Chaudhuri, P. P. (2005). Design of nonlinear CA based TPG without prohibited pattern set in linear time. Journal of Electronic Testing, 21(1), 95-107.

58. Metropolis, N., & Ulam, S. (1949). The monte carlo method. Journal of the American statistical association, 44(247), 335-341.

59. Zio, E., Podofillini, L., & Zille, V. (2006). A combination of Monte Carlo simulation and cellular automata for computing the availability of complex network systems. Reliability Engineering & System Safety, 91(2), 181-190.

60. Agrawal, V. D., Kime, C. R., & Saluja, K. K. (1993). A tutorial on built-in self-test. I. Principles. IEEE Design & Test, 10(1), 73-82.

61. Seth, S. C., Agrawal, V. D., & Farhat, H. (1990). A statistical theory of digital circuit testability. CSE Journal Articles, 32.

62. Wang, S. (2007). A BIST TPG for low power dissipation and high fault coverage. IEEE transactions on very large scale integration (VLSI) systems, 15(7), 777-789.

63. Lee, K.-J. Introduction to VLSI testing. http://www.ece.uc.edu/~wjone/intro.pdf.

64. BIST Analyzer & Diagonyzer (BISTAD). http://www.pld.ttu.ee/applets/bista.

65. Alvarez, G. A., & Cristian, F. (1997, May). Centralized failure injection for distributed, fault-tolerant protocol testing. In Distributed Computing Systems, 1997., Proceedings of the 17th International Conference on (pp. 78-85). IEEE.

66. Karlsson, J., Folkesson, P., Arlat, J., Crouzet, Y., & Leber, G. (1995). Integration and comparison of three physical fault injection techniques. In Predictably Dependable Computing Systems (pp. 309-327). Springer Berlin Heidelberg.

67. Dawson, S., Jahanian, F., Mitton, T., & Tung, T. L. (1996, June). Testing of fault-tolerant and real-time distributed systems via protocol fault injection. In Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on (pp. 404-414). IEEE.

68. Steininger, A. (2000). Testing and built-in self-test–A survey. Journal of systems Architecture, 46(9), 721-747.

69. Fushimi, M., & Hadano, T. (2008). Optimal Configurations of Cell Automata to Generate Test Stimuli for VLSI.

70. De la Guia Martinez, D., & Dominguez, A. P. (1999). Pseudorandom number generation based on nongroup cellular automata. In Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on (pp. 370-376). IEEE.

71. Prime95. http://en.wikipedia.org/wiki/Prime95.

72. Prime-Generating Cellular Automaton. http://demonstrations.wolfram.com/PrimeGeneratingCellularAutomaton/

73. Maji, P., Shaw, C., Ganguly, N., Sikdar, B. K., & Chaudhuri, P. P. (2003). Theory and application of cellular automata for pattern classification. Fundamenta Informaticae, 58(3-4), 321-354.

74. Das, D., & Misra, R. (2011). Programmable cellular automata based efficient parallel AES encryption algorithm. arXiv preprint arXiv:1112.2021.

75. Wolfram, S. (2002). A new kind of science (Vol. 5). Champaign: Wolfram media.

76. Prime-Generating Cellular Automaton. http://demonstrations.wolfram.com/PrimeGeneratingCellularAutomaton/

77. Stress (torture) Testing. https://en.wikipedia.org/wiki/Stress_testing

78. Fermat Primality Testing. http://en.wikipedia.org/wiki/Fermat_primality_test.

79. Rajasekaran, S., & Pai, G. V. (2003). Neural networks, fuzzy logic and genetic algorithm: synthesis and applications (with cd). PHI Learning Pvt. Ltd.

80. Mitchell, M., Crutchfield, J. P., & Das, R. (1996, May). Evolving cellular automata with genetic algorithms: A review of recent work. In Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96).

81. Chavoya, A., & Duthen, Y. (2006, July). Using a genetic algorithm to evolve cellular automata for 2D/3D computational development. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 231-232). ACM.

82. Itai, A. (2001). Generating permutations and combinations in lexicographical order. Journal of the Brazilian Computer Society, 7(3), 65-68.

83. Gu, J., Purdom, P. W., Franco, J., & Wah, B. W. (1996). DIMACS Series in Discrete Mathematics and Theoretical Computer Science.

84. Weisstein E. W., Tessellation, MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Tessellation.html.

85. Insall M., Weisstein Eric W., Lattice, MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Lattice.html.

86. Weisstein E. W., Injection, MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Injection.html.

87. Weisstein E. W., Surjection, MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Surjection.html.

88. Weisstein E. W., Bijection, MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Bijection.html.

89. Bijection, Injection and Surjection.
http://en.wikipedia.org/wiki/Bijection,_injection_and_surjection.

90. Insall M., "Lattice Isomorphism" From MathWorld--A Wolfram Web Resource, created by E. W. Weisstein.
http://mathworld.wolfram.com/LatticeIsomorphism.html.

91. Polynomial over the Rational Field.

http://cims.nyu.edu/~kiryl/Algebra/Section_3.10--Polynomials_Over_The_Rational_Field.pdf.

92. Wang, D. K., & Compagner, A. (1993). On the use of reducible polynomials as random number generators. Mathematics of Computation, 60(201), 363-374.

93. Cenzer, D., & Remmel, J. B. (2013, January). Sub-computable bounded pseudorandomness. In International Symposium on Logical Foundations of Computer Science (pp. 104-118). Springer Berlin Heidelberg.

94. Bojinov, H., Sanchez, D., Reber, P., Boneh, D., & Lincoln, P. (2012). Neuroscience meets cryptography: designing crypto primitives secure against rubber hose attacks. In Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), (pp 129-141).

95. Randomness of statistical sampling: the runs' test.
https://home.ubalt.edu/ntsbarsh/business-stat/otherapplets/Randomness.htm.

96. Test for randomness: the runs' test. https://home.ubalt.edu/ntsbarsh/business-stat/opre504.htm#rrunstest.

97. Gary, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-completeness.

98. NP Complete problems.
http://www.cs.berkeley.edu/~vazirani/algorithms/chap8.pdf

99. Parallel Programming Techniques.
http://jcsites.juniata.edu/faculty/rhodes/smui/parex.htm.

100. Grid Matrix Simulator. http://research.nektra.com/Grid_Matrix.

101. Mocskos, E. E., Yabo, P., Turjanski, P. G., & Slezak, D. F. (2012). Grid Matrix: a grid simulation tool to focus on the propagation of resource and monitoring information. Simulation.

102. SimGrid. http://simgrid.gforge.inria.fr/

103. Li, W., & Packard, N. (1990). The structure of the elementary cellular automata rule space. Complex Systems, 4(3), 281-297.

104. Langton, C. G. (1990). Computation at the edge of chaos: phase transitions and emergent computation. Physica D: Nonlinear Phenomena, 42(1), 12-37.

105. Li, W., & Packard, N. (1990). The structure of the elementary cellular automata rule space. Complex Systems, 4(3), 281-297.

106. Kshemkalyani, A. D., & Singhal, M. (2011). Chapter 16: Distributed computing: principles, algorithms, and systems. Cambridge University Press. 598-628.

107. Transposition Cipher.
http://global.britannica.com/EBchecked/topic/603174/transposition-cipher.

108. Injective and Surjective.
http://www.math.lsa.umich.edu/~speyer/417/InjectiveSurjective.pdf

109. Gaborit, P., Ruatta, O., Schrek, J., & Zémor, G. (2014, October). RankSign: an efficient signature algorithm based on the rank metric. In International Workshop on Post-Quantum Cryptography (pp. 88-107). Springer International Publishing.

110. Kshemkalyani, M. Singhal, Chapter 16: Authentication in distributed system, distributed computing: principles, algorithms and systems, Cambridge University Press, (2011) 598-628.

111. S. K. H. Islam, G. P. Biswas, K.-K. R. Choo, Cryptanalysis of an improved smartcard-based remote password authentication scheme, Information Sciences Letters, 3.1 (2014) 35-40.

112. Z. Siddiqui, A. H. Abdullah, M. K. Khan, S. Alghamdi, Smart environment as a service: three factor cloud based user authentication for telecare medical information system, Journal of medical systems, 38.1 (2014) 1-14.

113. S. Kolhe, S. Dhage, Trusted platform for support services in cloud computing environment, Int. Conf. System Engineering and Technology (ICSET), Bandung, (2012) 1-6.

114. J. Timpner, D. Schürmann, L. Wolf, Secure smartphone-based registration and key deployment for vehicle-to-cloud communications, ACM workshop on Security, privacy & dependability for cyber vehicles, (2013) 31-36.

115. H. J. Yang, V. Costan, N. Zeldovich, S. Devadas, Authenticated storage using small trusted hardware, ACM workshop Cloud Computing Security (2013) 35-46.

116. L. A. Dunning, R. Kresman, Privacy preserving data sharing with anonymous id assignment, IEEE Tran. Information Forensics and Security, 8.2 (2013) 402-413.

Appendix

A

AI- Artificial Intelligence

B

BIST- Built-In Self-Test

C

CUT- Circuit-Under-Test

CA- Cellular Automata

D

DFI- Digital Forensics Investigation

E

ELCA- Equal Length Cellular Automata

F

FI- Fault Insertion

H

HbP- Higher bit Position

I

IC- Integrated Circuit

L

LbP- Lower bit Position

LFSR- Linear Feedback Shift Register

M

MaxCA- Maximum Length Cellular Automata

M-C- Monte-Carlo

N

NoC- Network on Chip

NBCA- Null Boundary Cellular Automata

NP- Not Polynomial

O

ORA- Output Response Analyzer

OTP- One Time Password

P

P- Polynomial

PBCA- Periodic Boundary Cellular Automata

PRNG-Pseudo-Random Number Generator

PRPG- Pseudo-Random Pattern generator

PPS- Prohibited Pattern Set

R

RNG- Random Number Generator

RRNG- Recursive-Random Number Generator

S

SoC- System on Chip

SUT- System-Under-Test

T

TCS- Theoretical Computer Science

TRNG- True-Random Number Generator

TRPG- True-Random Pattern Generator

TPG- Test Pattern Generator

TF- Transition Faults

V

VLSI- Very Large Scale Integration

Author’s Biography

Mr. Arnab Mitra has completed his school level education form Katwa Kashi Ram Das Institution, Katwa-713130, India. He has qualified his Secondary level education from the West Bengal Board of Secondary Education, India in 1996 followed by Higher Secondary level education in 1998 from West Bengal Council of Higher Secondary Education, India. He has received his under-graduate degree in B. E. (Information Technology) in 2003 from University of North Bengal, Siliguri-734430, India followed by M. Tech (Computer Science & Engineering) in 2010 from West Bengal University of Technology, Kolkata-700064, India. He has been registered for his Ph.D. (Engg.) with Jadavpur University, Kolkata-700032, India. He has received and availed the prestigious EU- Erasmus Mundus sponsored 'cLink' Ph.D. mobility (grant agreement no. 212-26451001-001-EM action 2 partnerships) for the session July 2015 to May 2016 at "Gheorghe Asachi" Technical University of Iasi, Romania.

His research area includes Cellular Automata, Distributed Computing, Artificial Intelligence. He has served as invited reviewer for several international journals, and conferences of international repute.