VLSI Architecture of a Cellular Automata Machine

A. R. KHAN
Departments of Electronics & Computer Science
University of Kashmir, Srinagar 190006, India

P. P. CHOUDHURY* AND K. DIHIDAR
Computer Science Unit, Indian Statistical Institute
203, B. T. Road, Calcutta 700035, India
pabitra@isical.ernet.in

S. MITRA
Department of Electrical Engineering
Stanford University, Stanford, CA, 94305, U.S.A.

P. SARKAR
Computer Science Unit, Indian Statistical Institute
203, B. T. Road, Calcutta 700035, India

(Received and accepted September 1996)

Abstract—In the past, Cellular Automata based models and machines [1] have been proposed for simulation of physical systems without any analytical insight into the behaviour of the underlying simulation machine. This paper makes a significant departure from this traditional approach. An elegant mathematical model using simple matrix algebra is reported in this paper for characterizing the behaviour of two-dimensional nearest neighbourhood linear cellular automata with null and periodic boundary conditions. Based on this mathematical model, a VLSI architecture of a Cellular Automata Machine (CAM) has been proposed. Interesting applications of CAM in the fields of image analysis and fractal image generation are also reported.

Keywords—Cellular automata, Simulation machine, VLSI architecture, Matrix algebra, Image analysis, Fractal image generation.

1. INTRODUCTION

As the semiconductor technology is moving towards the submicron era, the system designers try to embed complex functions from software domain to hardware blocks on the silicon floor. At the same time for keeping the design complexity within a feasible limit, the designers are forced to look for simple, regular, modular, cascadable, and reusable building blocks for implementing various complex functions. The homogeneous structure of cellular automata (CA) is a right candidate to fulfill all the above objectives. Moreover, the demand for parallel processing architectures has gained importance with the ever increasing need for faster computing. To this end, we are motivated to use the two-dimensional cellular automata model to arrive at the easily implementable parallel processing architecture in VLSI. This parallel architecture built around the CA machine (CAM) suits ideally for a variety of applications.

*Author to whom all correspondence should be addressed.

The study of Cellular Automata (CA) dates back to John von Neumann in the early 50's. Von Neumann [2] framed CA as a cellular space capable of self-reproduction. Since then, many researchers have taken interest in the study of CA for modeling the behaviour of complex systems. Wolfram *et al.* [3] studied one-dimensional CA with the help of polynomial algebra. Pries *et al.* [4] studied one-dimensional CA exhibiting group properties based on a similar kind of polynomial algebra. Later, Das *et al.* [5] extended the characterization of one-dimensional CA with the help of matrix algebra. In recent years, many applications of one-dimensional CA have been reported [6–9]. On the other hand, two-dimensional CA is not yet a well-studied area. Packard *et al.* [10] reported some empirical studies on two-dimensional Cellular automata depending on five neighbourhood CA. Chowdhury *et al.* [11] extended the theory of 1-D CA built around matrix algebra for characterizing 2-D CA. However, emphasis was laid on special class of additive 2-D CA, known as Restricted Vertical Neighbourhood (RVN) CA. In this class of 2-D CA, the vertical dependency of a site is restricted to either the sites on its top or bottom, but not both.

To the best of our knowledge, the research reported in this paper is the first attempt to develop an analytical tool to study all the nearest neighbourhood 2-D CA linear transformations. It deals with the characterization of 2-D nine neighbourhood linear CA. A general framework has been proposed for the study of the state transition behaviour of this class of 2-D CA. A few interesting results of some specific CA transformations are reported using matrix algebra. A few applications of 2-D CA are also reported.

A CA machine (CAM) has been proposed around the parallel architecture of 2-D CA. Such a CAM can be economically built with the currently available VLSI technology. A wide variety of applications can be developed around the parallel architecture of CAM. Such a CAM can serve as a simulation engine to study a wide variety of hybrid CA configurations. Analysis of 2-D images can be undertaken in such a machine in order to identify their specific features. This simulation engine can also be employed for generation of fractal images. Some of these applications are briefly reported.

Section 2 introduces CA preliminaries. Section 3 highlights a few sample results on the characterization of uniform 2-D CA transformations. Two simple applications of 2-D CA are discussed in Section 4. A VLSI architecture of Cellular Automata Machine (CAM) is reported in Section 5 along with a few applications in the field of image analysis and fractal image generation.

2. BASIC CONCEPTS

Prior to introducing the 2-D CA framework, a brief introduction on 1-D CA [5,6,12] is reported below. The one-dimensional (1-D) CA structure can be viewed as a discrete lattice of sites or cells, where each cell can assume either the value 0 or 1. The next state of a cell is assumed to depend on itself and on its two neighbours (for 3-neighbourhood dependency). The cells evolve in discrete time steps according to some deterministic rule that depend only on local neighbourhood. Mathematically, the next state transition of the i^{th} cell can be represented as a function of the present states of the i^{th}, $(i+1)^{\text{th}}$, and $(i-1)^{\text{th}}$ cells:

$$q_i(t+1) = f\left(q_i(t), q_{i+1}(t), q_{i-1}(t)\right),$$

where 'f' is known as the rule of a CA.

If the next state function of a cell is expressed in the form of a truth table, then the decimal equivalent of the output is conventionally called the rule number for the cell.

Neighbourhood state:	111	110	101	100	011	010	001	000	
Next state:	0	1	0	1	1	0	1	0	rule-90
Next state:	0	1	1	1	1	0	0	0	rule-120

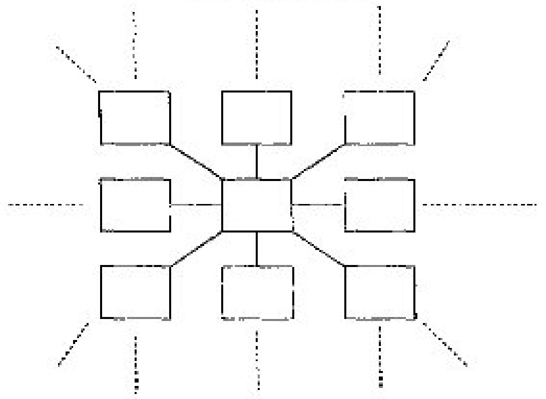

Figure 1. A 2-D CA structure. Note: A box represents a CA cell with a D flip-flop and next state logic.

The top row gives all the 8 possible states of the 3 neighbouring cells at the time instant 't', while the second and third rows give the corresponding states of the i^{th} cell at time instant $(t+1)$ for two illustrative CA rules. A few definitions are next introduced.

DEFINITION 1. *If the same rule is applied to all the cells in a CA, then the CA is called a uniform or regular CA.*

DEFINITION 2. *If different rules are applied to different cells in a CA, then the CA is called a hybrid CA.*

DEFINITION 3. *If in a CA, the neighbourhood dependence is on EX-OR or EX-NOR only, then the CA is called an additive CA. Specifically, a linear CA employs XOR rules only.*

DEFINITION 4. *A Periodic Boundary CA is the one in which the extreme cells are adjacent to each other.*

DEFINITION 5. *A Null Boundary CA is the one in which the extreme cells are connected to logic 0-state.*

DEFINITION 6. *A CA whose transformation is invertible is called a group CA; otherwise it is a nongroup CA. For a group CA, the dimension of kernel is 0 (that is, the transformation is a full rank one); for a nongroup CA, the dimension of the kernel is nonzero.*

2.1. Mathematical Model for Study of 2-D CA State Transition Behaviour

In a two-dimensional nearest neighbourhood linear CA, the next state of a particular cell of the 2-D CA is affected by the current state of itself and 8 cells in its nearest neighbourhood (Figure 1). Different dependencies are taken into account by means of various CA rules. For the sake of simplicity, in this section we take into consideration only the linear rules, i.e., the rules which can be realized by EX-OR operation only. XNOR rules will be dealt with in Section 5. A specific rule convention evolved by us is noted below.

64	128	256
32	1	2
16	8	4

The central box represents the current cell (that is, the cell being considered) and all other boxes represent the eight nearest neighbours of that cell. The number within each box represents the rule number associated with that particular neighbour of the current cell. That is, if the central cell has got dependency only on itself—it is referred to as Rule 1, if it depends only on its top neighbour, it is Rule 128, and so on. In case the cell has dependency on two or more neighbouring cells, the rule number will be arithmetic sum of the numbers of the relevant cells. For example, the 2-D CA rule $171N$ $(128+32+8+2+1)$ refers to the five neighbourhood dependency of the (central) cell (top, left, bottom, right, and self) under null boundary condition, whereas $171P$ refers to the same neighbourhood dependency under periodic boundary condition.

This 2-D CA behaviour can be analyzed with the help of an elegant mathematical model, where we use two fundamental matrices to obtain row and column dependencies of the cells. Let the two-dimensional binary information matrix be denoted as X_t that represents the current state of a 2-D CA configured with a specific rule. The next state of any cell will be obtained by EX-OR operation of the states of its relevant neighbours associated with the rule. The global transformation associated with different rules can be made effective with the following two fundamental matrices referred to as T_1 and T_2 in the rest of the paper.

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = T_1 \quad \text{and} \quad \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = T_2.$$

The following lemma specifies the value of the next state of a 2-D CA referred to as X_{t+1} given that its current state is X_t. The CA is assumed to be configured with a primary rule only—that is, the dependence is only on one of the nine neighbours.

LEMMA 1. *The next state transition of all the* **Primary Rules** $(1,2,4,8,16,32,64,128,256)$ *can be represented as*

$$\begin{aligned}
&\textit{Rule 1} &&\Longrightarrow [X_{t+1}] = [X_t] \\
&\textit{Rule 2} &&\Longrightarrow [X_{t+1}] = [X_t]\,[T_2] \\
&\textit{Rule 4} &&\Longrightarrow [X_{t+1}] = [T_1]\,[X_t]\,[T_2] \\
&\textit{Rule 8} &&\Longrightarrow [X_{t+1}] = [T_1]\,[X_t] \\
&\textit{Rule 16} &&\Longrightarrow [X_{t+1}] = [T_1]\,[X_t]\,[T_1] \\
&\textit{Rule 32} &&\Longrightarrow [X_{t+1}] = [X_t]\,[T_1] \\
&\textit{Rule 64} &&\Longrightarrow [X_{t+1}] = [T_2]\,[X_t]\,[T_1] \\
&\textit{Rule 128} &&\Longrightarrow [X_{t+1}] = [T_2]\,[X_t] \\
&\textit{Rule 256} &&\Longrightarrow [X_{t+1}] = [T_2]\,[X_t]\,[T_2].
\end{aligned}$$

PROOF. Let

$$[X_t] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad (\text{where } a_{11}, a_{12} \dots a_{33} \text{ takes values 0 or 1})$$

be the state of CA at time t.

(i) Postmultiplying X_t by T_2, we get the state of CA at time $(t+1)$ as

$$[X_{t+1}] = \begin{bmatrix} a_{12} & a_{13} & 0 \\ a_{22} & a_{23} & 0 \\ a_{32} & a_{33} & 0 \end{bmatrix},$$

which is the same as Rule 2 applied to X_t.

(ii) Premultiplying X_t by T_1, we get the state of CA at time $(t+1)$ as

$$[X_{t+1}] = \begin{bmatrix} a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ 0 & 0 & 0 \end{bmatrix},$$

which is the same as Rule 8 applied to X_t.

(iii) Premultiplying X_T by T_1 and postmultiplying the result by T_2, we get state of CA at time $(t+1)$ as

$$[X_{t+1}] = \begin{bmatrix} a_{22} & a_{23} & 0 \\ a_{32} & a_{33} & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

which is the same as Rule 4 applied to X_t.

Similarly, others can be proved. ∎

Rules (other than primary) having dependence on more than one cell are referred to as Secondary Rules.

LEMMA 2. *The next state transition of a CA configured with a secondary rule can be represented as modulo 2 sum of the matrices of the concerned primary rules.*

PROOF. For example,

(i) Rule 3 = Rule 1 + Rule 2, the next state transition can be represented as

$$[X_{t+1}] = [X_t] + [X_t]\,[T_2],$$

and

(ii) Rule 170 = Rule 2 + Rule 8 + Rule 32 + Rule 128, the next state transition for Rule 170 can be represented as

$$\begin{aligned}[X_{t+1}] &= [X_t]\,[T_2] + [T1]\,[X_t] + [X_t]\,[T_1] + [T_2]\,[X_t] \\ &= [X_t]\,[T_1 + T_2] + [T_1 + T_2]\,[X_t] \\ &= [X_t]\,[S] + [S]\,[X_t],\end{aligned}$$

where $[S] = [T_1 + T_2]$.

Let

$$[X_t] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

(i) Therefore,

$$\begin{aligned}[X_{t+1}] &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix},\end{aligned}$$

(ii)

$$\begin{aligned}[X_{t+1}] &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.\end{aligned}$$

∎

3. CHARACTERIZATION OF 2-D CELLULAR AUTOMATA

For the convenience of analysis, we will convert each of the rules (both Primary and Secondary) into a transformation denoted by T. We want to look at the transformation T such that T operating on the current CA state X (the binary information matrix of dimension $m \times n$) generates the next state $[X']_{m\times n}$. The convention followed is as noted below:

$$T(X)_{m\times n} = [T]_{mn\times mn}, \qquad \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix}_{mn\times 1} = \begin{bmatrix} X'_1 \\ X'_2 \\ \vdots \\ X'_m \end{bmatrix}_{mn\times 1},$$

where $X_1, X_2, \dots, X_m$ are the rows of X and $X'_1, X'_2, \dots, X'_m$ are the rows of the next state X'.

In the next lemma, we try to formulate the above T for a given Rule R applied uniformly over all the 2-D CA cells.

LEMMA 3. *The equivalent one-dimensional map matrix for any Rule R can be represented as*

$$T_R = \begin{bmatrix} D & U & 0 & 0 & \cdots & 0 & 0 & 0 \\ L & D & U & 0 & \cdots & 0 & 0 & 0 \\ 0 & L & D & U & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & L & D & U \\ 0 & 0 & 0 & 0 & \cdots & 0 & L & D \end{bmatrix}_{mn\times mn},$$

where D, L, and U are one of the following matrices of the order of $n \times n$:

$$[0], [I], [T_1], [T_2], [I+T_1], [I+T_2], [S], \text{ and } [I+S].$$

PROOF. For getting the first column of the above matrix, we utilize

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{m\times n}.$$

Now applying Rule R to each cell starting from first row, we get the first column of T_R. Then, shifting 1 towards the right just once, and applying the same rule again, we get the second column, and so on. ∎

EXAMPLE 1. Let us consider the dimension of the 2-D CA to be 3×3. In order to obtain the T matrix corresponding to Rule 2 applied over all the cells, we follow the following steps.

To obtain the i^{th} column of matrix T corresponding to Rule 2, we take a 3×3 binary matrix with all zeroes excepting the position $(i/3, i \bmod 3)$, which contains 1.

For example, to obtain the 0th column of T, we use the following binary matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{3\times 3}.$$

Now, considering this matrix as a state of the 2-D CA, we apply Rule 2 (right dependency) to obtain the first (0th) column of T_{R_2} (matrix T for Rule 2).

$$T_{R2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{9\times 9}$$

$$= \begin{bmatrix} (T_1)_{3\times3} & 0 & 0 \\ 0 & (T_1)_{3\times3} & 0 \\ 0 & 0 & (T_1)_{3\times3} \end{bmatrix}_{3\times3}. \qquad \blacksquare$$

For the subsequent results, proofs are omitted due to shortage of space.

LEMMA 4. *Rank of fundamental matrices $(T_1)_{n\times n}$ and $(T_2)_{n\times n}$ is $n-1$.*

LEMMA 5. *All primary rules other than Rule 1 are nongroup CA.*

Next, we are going to highlight some of the most interesting results for some specific linear transformations viz. 170 (null boundary condition), i.e, $170N$; 170 (periodic boundary condition), i.e, $170P$; and $171N$, etc. For analyzing Rule $170N$, we first converted it into a *uniformly partitioned* one-dimensional map matrix. Next, applying matrix algebraic formulations, we can prove the following results.

LEMMA 6. *If A is any matrix of the form*

$$\begin{bmatrix} I & 0 & 0 & \cdots & 0 & 0 & 0 \\ S & I & 0 & \cdots & 0 & 0 & 0 \\ I & S & I & 0\cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & I & S & I \end{bmatrix},$$

then

$$A^{-1} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ S & I & 0 & \cdots & 0 \\ I+S^2 & S & I & \cdots & 0 \\ S^3 & I+S^2 & S & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{m-2}(S) & p_{m-3}(S) & p_{m-4}(S) & \cdots & p_0(S) = I \end{bmatrix}.$$

LEMMA 7. *Finding the dimension of the kernel for T_{170} is equivalent to finding the dimension of the kernel of the matrix $p_m(S_n)$, where $p_m(\lambda)$ is the characteristic polynomial of the matrix S.*

As a consequence of these results, we can prove the following theorem which was first arrived at by Sutner [13] in connection with σ-game. The same result was derived by Barua and Ramkrishnan [14] who viewed this $170N$ rule as superposition of 1-D vertical and horizontal dependency transformations. However, using the above simple matrix algebraic formulations, we can easily deduce the following results.

THEOREM 1. *The dimension of the kernel of T_{170N} is gcd $(m+1, n+1) - 1$.*

Similar to Theorem 1, we have an interesting result for Rule $170P$.

THEOREM 2. *The dimension of the kernel of T_{170P} is $2gcd(m, n)$ when m or n or both are even and $2gcd(m, n) - 1$, when both m and n are odd.*

Next, we present another result for Rule $171N$ which will be used in encryption methodology discussed in this paper.

THEOREM 3. *A sufficient condition for invertibility of T_{171N} is that $m = 2^{x_1} - 1$ and $n = 2^{x_2} - 1$, for some x_1 and x_2.*

The above results relate to the algebraic properties of the corresponding transformations—that is, for a given m and n, the number of predecessors for any reachable state in the state transition diagram (STD) can easily be calculated. To complete the characterization, we need to consider the minimal polynomial for the corresponding transformation.

EXAMPLE 2. We consider a 2×3 2-D CA configured as $170N$. The corresponding matrices are

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}_{2\times 3} \Rightarrow \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}_{6\times 6} \Rightarrow \begin{pmatrix} S_3 & I_3 \\ I_3 & S_3 \end{pmatrix}_{6\times 6}.$$

On diagonalization [15], the factor polynomials are $(x^2 + 1)$ and $(x^4 + 1)$. Using Elspas's Theorem [16], we find that the cyclic structure is $[4(1), 6(2), 12(4)]$. So, the state transition diagram corresponding to this 2-D CA contains 4 cycles of length 1, 6 cycles of length 2, and 12 cycles of length 4. ∎

A large volume of interesting results of 2-D CA (with both XOR and XNOR rules) behaviour have been developed that are not reported in this paper since the major motivations of this paper are

1. to develop the mathematical foundation of the analytical model to study 2-D CA behaviour,
2. to develop a few applications of 2-D CA, and
3. to build a CAM (CA Machine) based on this mathematical model that can be used to solve problems in diverse fields.

Items (2) and (3) of the above list will be dealt with in the next two sections.

4. A FEW APPLICATIONS OF 2-D CA

4.1. VLSI Testing

Test technology has failed to match the growth of circuit complexity and size. The only viable option available to the design community is to employ DFT (Design For Testability) techniques. Notable among the DFT techniques is the full scan design. However, with the growth of circuit size, serial scan in and scan out of larger volume of test and response data have become a major bottleneck along with the larger test circuit overhead. Partial scan techniques attempt to reduce this overhead. In both cases, automatic test pattern generators are used—as a result test generation, and test application time becomes significant. Built-In-Self-Test (BIST) structures provide an on-chip test generation and test evaluation methodology with an aim to reduce the drawback of DFT techniques discussed previously. In this section, we project 2-D CA as a BIST structure for VLSI circuits.

4.1.1. Pseudo-exhaustive testing

For a complex circuit with a large number of inputs, pseudo-exhaustive testing has been found to be suitable where each of the outputs depend only on a subset of the inputs—this results in a test size much less compared to the exhaustive test size 2^n for an n-input circuit under test (CUT). Given any group 1-D CA, Das and Chaudhuri [6] present an algorithm for finding out the bit positions where the pseudo-exhaustive patterns are generated. The same result holds for the 2-D CA case also. Moreover, due to the incorporation of nine neighbourhood, we have wide variety of 2-D CA characteristic matrices and its factor polynomials—this gives us much wider flexibility to employ 2-D CA for pseudo-exhaustive test pattern generation.

EXAMPLE 3. Let us consider a 2×4 2-D CA where each cell is configured with Rule 34 or Rule 35. For the cells on the row boundaries (except the cell (1,1) and (2,4)) (i.e., cell (2,1))—its left dependence comes from the output of cell (1,4)—something like a mesh. The corresponding T matrix is

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

The characteristic polynomial is $f(x) = (1+x)(1+x^2+x^3)(1+x+x^4)$. The factors of the characteristic polynomial are

- $f_1(x) = (1+x)$,
- $f_2(x) = (1+x^2+x^3)$,
- $f_3(x) = (1+x+x^4)$.

A primitive factor of the characteristic polynomial is $f_2(x) = (1+x^2+x^3)$. Per the algorithm presented in [6], the pseudo-exhaustive bit positions are cells $(1,1)$, $(1,2)$, and $(1,3)$. ∎

4.1.2. Generation of pseudo-random test patterns

1-D CA has been projected as a pseudo-random pattern generator in a large number of publications. As noted in [11], the 2-D CA structure is expected to generate much better pseudo-random patterns so far as randomness is concerned—this is because of the varieties of neighbourhood which are not available in case of three neighbourhood 1-D CA. In [12], a variety of tests for randomness, viz. Equidistribution Test, Correlation, etc., has been considered [17]. It has been established that 2-D CA is a much better PRPG with superior randomness qualities than 1-D CA or LFSR. The 2-D CA described in [11] is RVN-CA (Reduced Vertical Neighbourhood CA) which is a subset of the general 2-D CA we have considered in this paper. Hence, it is natural that we achieve improvement in randomness of the patterns by using the general 2-D CA structure. Such 2-D CA structures can be effectively employed as multiple parallel pseudo-random pattern generator which provide much better randomness of patterns than RVN-CA, 1-D CA, or LFSR.

However, some circuits exist which are inherently random pattern resistant and thus require large number of pseudo-random patterns for achieving high fault coverage. So, use of weighted-random patterns has been proposed [18]. The next section briefly highlights an elegant scheme for generation of weighted random patterns.

4.1.3. Generation of weighted random test patterns

In this section, we propose a scheme for generating *weighted-random* patterns using 2-D CA. Given the neighbourhood dependencies of the cells and the boundary conditions, we can construct the T-matrix, thereby generating the entire state transition behaviour of the 2-D CA. Given a cycle C belonging to the state transition diagram of the 2-D CA, let n be the total number of states in C (that is the cycle length is equal to n)—for any particular bit i, the probability of 1 is $2^k - 1$, where k is the number of bits of the CA, then we can obtain the probability of 1 at any bit to be equal to 0.5 by loading the CA with any nonzero seed. On the other hand, if C is of nonmaximal length, then the probability of 1 will be different from 0.5. Thus, corresponding to each cycle C in the state transition diagram of any group $m \times n$ 2-D CA, we can define the following tuple $\tau = \langle$cycle length, probability of 1 in bit position $(1,1), \ldots,$ probability of 1 in bit position (m,n), seed$\rangle$. Thus, if we load the CA with the seed and look at any bit position (i,j), we can distinctly tell the probability of getting 1 in that position—this is possible because of the matrix algebraic characterization of 2-D CA. We can generate a *program* for the 2-D CA by specifying the rules, the initial seed, and the number of cycles for which the 2-D CA has to be evolved. On running the 2-D CA with such a program, wide varieties of probability of 1 at different bit positions can be generated. The basic architecture is shown in Figure 2.

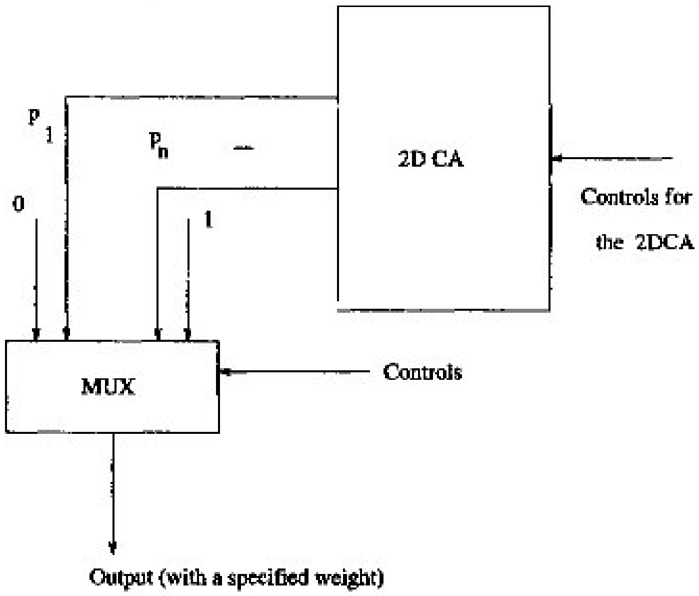

Figure 2. Architecture for weighted random pattern generation.

4.2. Cryptographic Application

Two-dimensional binary text or image information is a *natural* state of 2-D CA. For an invertible CA, the state transition diagram consists only of cycles, and hence, any state lies on a cycle. Let c be a configuration on a cycle of length L. If the CA is evolved for x $(0 \le x < L)$ steps starting with initial state c, then a new state d is reached. Given the state d, it is possible to get back to c by loading the CA with initial state d and evolving for $(L - x)$ steps. Thus, d can be considered to be the cipher text. The evolution for x steps, the enciphering algorithm, and the evolution for the next $(L - x)$ steps is used in the deciphering algorithm.

Based on this idea, we now describe an enciphering scheme for two-dimensional information. A 2-D uniform $m \times n$ CA using only Rule $170N$ is invertible if and only if $(m+1)$ and $(n+1)$ are coprime. For the case of CA with Rule 171, a sufficient condition for invertibility is that $m = 2^{x_1} - 1$ and $n = 2^{x_2} - 1$, for some x_1 and x_2. Given an $M \times N$ 2-D state, we form a partition of M and N, of the form $M = m_1 + m_2 + \cdots + m_{r_1}$ and $N = n_1 + n_2 + \cdots + n_{r_2}$ such that for each pair (m_i, n_j), $(1 \le i \le r_1)$, $(1 \le j \le r_2)$, we have either

(a) $gcd(m_i + 1, n_j + 1) = 1$ (Theorem 1), or
(b) $m_i = 2^{x_1} - 1$ and $n_j = 2^{x_2} - 1$ (Theorem 3),

for some x_1 and x_2. This will divide the $M \times N$ into blocks of $m_i \times n_j$ grids. We will separately apply Rule $170N$ or $171N$ on these $m_i \times n_j$ blocks depending on whether condition (a) or (b) is satisfied. The conditions ensure that each individual transformation is invertible.

Such a CA structure is set up at both the sender and the receiver end. To encipher a given information, we load the $M \times N$ 2-D state into the structure described above. For each $m_i \times n_j$ information block (c_{ij}), we evolve for x_{ij} steps with Rule $170N$ or $171N$ to get d_{ij} which is also a $m_i \times n_j$ block. Let l_{ij} be the length of the cycle on which c_{ij} lies. Then along with d_{ij}, we also send $l_{ij} - x_{ij}$ to the receiver. The partition of M and N, l_{ij}, x_{ij} are kept secret to the adversary.

4.2.1. Complexity and versatility

For any cryptosystem, the invulnerability is equivalent to showing that breaking the cipher text is computationally infeasible. Thus, a cipher is *secure* under the intractibility assumption of the problem. However, popular cryptosystems like DES are not altogether secure in this sense. In fact, extensive study of DES has shown some potential weaknesses [19] though up till now it has not been cryptoanalysed.

We first calculate the size of the key space. It is determined by the following factors.

1. For each m_i and n_j, we have the information matrix of size $m_i \times n_j$. The possible number of one-dimensional map matrices of dimension $(m_i n_j \times m_i n_j)$ is $2^{(m_i n_j)^2}$. The number of bits required to represent the possible number of one-dimensional map matrices is $(m_i n_j)^2 = k_1$ (say).
2. The basic difficulty presented to the adversary is to guess the partition of M and N. The number of partitions of a positive integer K, P_K is the coefficient of x^K in the expansion of $(1 + x + x^2 + x^3 + \cdots)(1 + x^2 + x^4 + x^6 + \cdots)(1 + x^3 + x^6 + x^9 + \cdots) \cdots (1 + x^i + x^{2i} + x^{3i} + \cdots) \cdots = (1 - x)^{-1}(1 - x^2)^{-1}(1 - x^3)^{-1} \cdots (1 - x^i)^{-1} \cdots = 1/((1 - x)(1 - x^2)(1 - x^3) \cdots (1 - x^i) \cdots)$. Thus an upper bound on the number of possible divisions of the $M \times N$ grid is $P_M P_N$, which is certainly a large number. Suppose k_1 number of bits are required to fix the partition matrix. Now, for each partition matrix, since the cycle length will be varying, we have to keep provision of $m_i n_j$ bits (to take care of the maximum length cycle). Thus, the length of the key for each partition matrix is $k_1 + l_{ij} - x_{ij} + 1)$ bits—the extra one bit is required to indicate whether Rule $170N$ or Rule $171N$ has been applied.

5. ARCHITECTURE OF 2-D CA BASED CAM

In Section 2, we have proposed a local neighbourhood 2-D CA where the next state of each cell depends on the current state of its neighbours viz. 4 orthogonal neighbours, 4 diagonal neighbours and itself. In order to design the most general structure, we should be able to configure a cell with a rule out of the 512 available rules. Moreover, to provide wider flexibility provisions should be kept to incorporate XNORs also—however, the 2-D CA transformation thus obtained will be an *affine* one.

Each 2-D CA cell is connected through nine switches to its nine nearest neighbours. In order to apply a particular rule, we have to apply 1 or 0 to the corresponding switches thereby closing or opening them. Thus, a nine bit word is required to control the nine switches corresponding to a single 2-D CA cell. In addition to nine, another bit is required to configure the cell in XOR or XNOR mode. Hence, we have a separate control plane where entry (i, j) stores a 10 bit control word corresponding to the particular rule employed to configure the cell (i, j) of the 2-D CA. The CAM architecture is shown in Figure 3. In effect, by providing a generalized 2-D CA structure, we have incorporated *programmability*. Hence, we can introduce the concept of a rule program which is a tuple containing the rules to be applied to the cells, the initial seed, and the number of cycles the 2-D CA has to be iterated.

The motivation for the parallel architecture of the CAM is derived from following considerations. The reader may recall that computation related to any $m \times n$ 2-D CA requires manipulation of $mn \times mn$ binary matrix which is computationally intensive, if not infeasible, when m and n cross 10—such a situation arises in a variety of applications. Therefore, we require an all-purpose *hardware simulation engine* which is provided by our architecture.

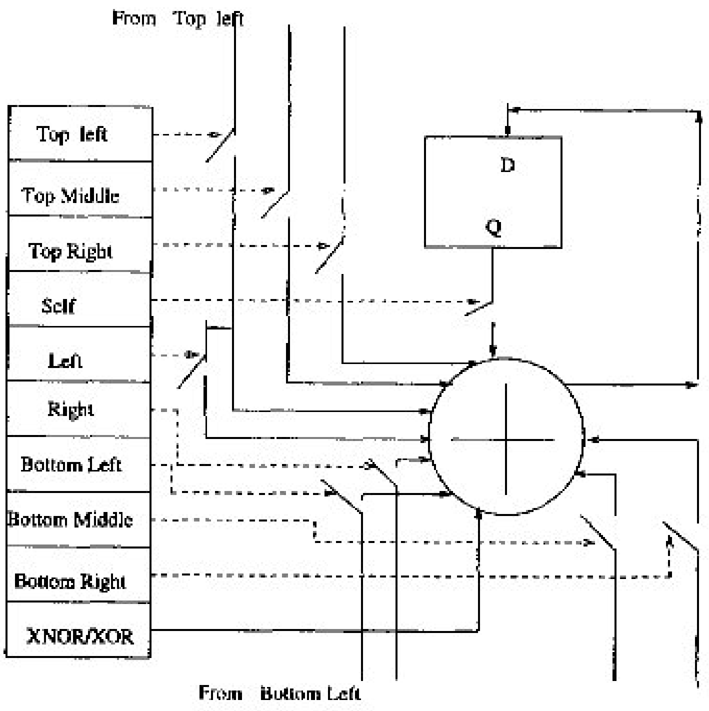

Figure 3. Architecture of the CAM.

It may be noted that such a machine can achieve a high simulation performance with an appreciably low hardware overhead compared to the *universal synthesizer* proposed by Toffoli [1].

5.1. Architectural Details of the CAM

This section provides the architectural details of a CAM corresponding to an $m \times n$ 2-D CA. As shown in Figure 3, there are three major blocks.

- An $m \times n$ matrix of CA cells—the details of a representative cell is shown in Figure 3.
- **The Control Memory** is required to configure the 2-D CA cells. Since any nearest neighbourhood 2-D CA cell has dependencies on 9 of its nearest neighbours, a typical control word should contain at least 9 bits—moreover, another bit is required to configure the cell in XOR or XNOR mode. Thus, for each cell, we require a 10 bit control word. Hence, the size of the control memory is $(m \times n \times 10)$.
- **A Seed Memory** is also kept to store b initial seeds of the 2-D CA. For that purpose, the size of the Seed Memory will be $(m \times n \times b)$.

If we restrict ourselves to uniform 2-D CA, then the size of the control plane reduces drastically to only $9 + (m \times n)$ bits. The term $(m \times n)$ is kept to take care of the XOR/XNOR for each cell. Thus, a drastic cost reduction is possible with a uniform 2-D CA based CAM architecture. However, we will be losing the programmability which is one of the essential requirements for carrying out exhaustive simulations with the CAM. So, we can adopt a hybrid approach where each of the m rows will be configured with the same rule. This reduces the size of the control memory to $m \times 9 + (m \times n)$ while preserving the programmability to some extent. Next, we report a few applications of CAM.

5.2. Application of CAM in Image Analysis

- **Zooming** of orthogonal structures can be performed by applying the following steps.

1. Apply Rule $170N$ for any square or rectangular structure lying inside the screen. The four corner pixels will become 0.

2. These four corners have to be tagged by Rule 2 or Rule 32.
3. Next, by (repeated) application of hybrid rules (generally 2 and 32), we fill up the inner cells with 0's.
4. Then we apply Rule 170N followed by tagging of the corners with Rules 2 and 32.
5. The above procedure will continue to zoom to any extent until the screen boundary is reached.

For the purpose of illustration, we consider the following example.

EXAMPLE 4. Let us consider a 9×12 2-D CA based CAM with the initial configuration as shown in Figure 4.

```
.  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .
.  .  *  *  *  *  *  *  *  *  .  .
.  .  *  .  .  .  .  .  .  *  .  .
.  .  *  .  .  .  .  .  .  *  .  .
.  .  *  .  .  .  .  .  .  *  .  .
.  .  *  *  *  *  *  *  *  *  .  .
.  .  .  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .
```

Figure 4.

After running Rule 170N, the resultant configuration is shown in Figure 5.

```
.  .  .  .  .  .  .  .  .  .  .  .
.  .  .  *  *  *  *  *  *  .  .  .
.  .  *  .  .  .  .  .  .  *  .  .
.  .  *  .  *  *  *  *  .  *  .  .
.  .  *  .  *  .  .  *  .  *  .  .
.  .  *  .  *  *  *  *  .  *  .  .
.  .  *  .  .  .  .  .  .  *  .  .
.  .  .  *  *  *  *  *  *  .  .  .
.  .  .  .  .  .  .  .  .  .  .  .
```

Figure 5.

Next, we apply Rules 2, 32 to tag the corners followed by an application of hybrid rules so that the contents of the inner cells become 1's. Now we apply Rule 170N followed by Rules 2 and 32 at the corners when we obtain the zoomed image (Figure 6).

```
*  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *
*  *  .  .  .  .  .  .  .  .  *  *
*  *  .  .  .  .  .  .  .  .  *  *
*  *  .  .  .  .  .  .  .  .  *  *
*  *  .  .  .  .  .  .  .  .  *  *
*  *  .  .  .  .  .  .  .  .  *  *
*  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *
```

Figure 6.

• **Boundary** of any orthogonal structure can be obtained with Rule 170N independent of the thickness of the vertical and horizontal columns in the original configuration.

• **Thinning** is an important procedure in image analysis. The following example demonstrates that vertical/horizontal columns can be thinned using different *uniform* rules.

EXAMPLE 5. Let us consider the initial configuration of a 8×5 2-D CA containing all 1's (Figure 7).

```
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
```

Figure 7.

After applying Rule $33N$ uniformly, we obtain all 1's in the first column and all 0's in the rest (Figure 8).

```
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
*  .  .  .  .
```

Figure 8.

Next, successively applying Rule $32N$, we can obtain the required thinned image (i.e., 1's in a particular column and 0's elsewhere) (Figure 9). ∎

```
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
.  .  *  .  .
```

Figure 9.

5.3. Generation of Fractal Image

Fractals are becoming increasingly popular in the fields like image compression [20], texture analysis, etc. Extensive research work has been carried out to characterize fractals of different dimensions using *Iterated Function Systems* (IFS) [21]. We are trying to generate the fractal patterns by employing a programmable CAM. It has been found that using a special type of 2-D CA, *single attractor transformation*, any prespecified fractal pattern can be generated starting from any initial configuration (*garbage*) after running the 2-D CA for a specified number of cycles. Hence, we can project CAM as a simulation engine for generating and studying fractal images as illustrated in the following example. We are carrying on investigation to characterize the

2-D CA analogue of different IFS transformations so that the self-similarity property is appropriately reflected by the 2-D CA CAM program.

EXAMPLE 6. Let us consider a 32×32 2-D CA CAM with a uniform rule of $170P$. Since this is a single attractor transformation, we can easily generate any standard fractal patterns **from any garbage information** by applying *XNOR's in selected positions* and then running the 2-D CA for 4 or more number of cycles. We are presenting two such fractal patterns (Figure 10). ∎

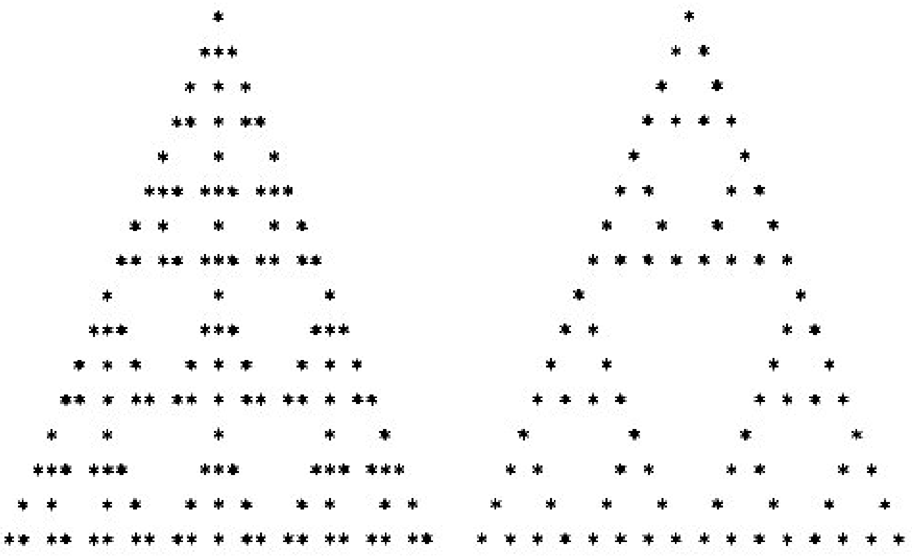

Figure 10.

6. CONCLUSION

In this paper, we have developed an analytical tool based on matrix algebra to characterize all the nearest neighbourhood 2-D CA transformations and highlighted its application in the fields of VLSI testing and cryptology. Next, we have proposed an architecture of a programmable CAM which suits VLSI implementation. Applications of this CAM-based simulation engine for image analysis and fractal pattern generation have also been reported. Further, efforts are being made to simulate various physical/physiological procedures with the help of this simulation engine—specifically, we are investigating to derive a one-to-one correspondence between a CAM program and the actual physical procedure that transforms a set of healthy human cells (tissue) into a set of cancerous cells. This may open up new avenues in cancer research.

REFERENCES

1. T. Toffoli and N. Margolus, *Cellular Automata Machines*, MIT Press, (1987).
2. J. Von Neumann, *The Theory of Self-Reproducing Automata*, (Edited by A.W. Burks), Univ. of Illinois Press, Urbana, (1966).
3. S. Wolfram, Statistical mechanics of cellular automata, *Rev. Mod. Phys.* **55** (3), 601–644 (July 1983).
4. W. Pries, A. Thanailakis and H.C. Card, Group properties of cellular automata and VLSI Applications, *IEEE Trans. on Computers* **C-35** (12), 1013–1024 (December 1986).
5. A.K. Das, Additive cellular automata: Theory and application as a built-in self-test structure, Ph.D. Thesis, I.I.T. Kharagpur, India, (1990).
6. A.K. Das and P.P. Chaudhuri, Vector space theoretic analysis of additive cellular automata and its applications for pseudo-exhaustive test pattern generation, *IEEE Trans. on Computers* **42** (3), 340–352 (March 1993).
7. D.R. Chowdhury, S. Basu, I.S. Gupta and P.P. Chaudhuri, Design of CAECC—Cellular automata based error correcting code, *IEEE Trans. on Computers* **43** (6), 756–764 (June 1994).
8. D.R. Chowdhury, I.S. Gupta and P.P. Chaudhuri, CA based byte error correcting code, *IEEE Trans. on Computers* **43** (3), 371–382 (March 1994).

9. S. Nandi, B.K. Kar and P.P. Chaudhuri, Theory and application of cellular automata in cryptography, *IEEE Trans. on Computers* **43** (12), 1346–1357 (December 1994).
10. N.H. Packard and S. Wolfram, Two-dimensional cellular automata, *Journal of Statistical Physics* **38** (5/6), 901–946 (1985).
11. D.R. Chowdhury, I.S. Gupta and P.P. Chaudhuri, A class of two-dimensional cellular automata and applications in random pattern testing, *Journal of Electronic Testing: Theory & Applications* **5**, 65–80 (1994).
12. D.R. Chowdhury, Theory and applications of additive cellular automata for reliable and testable VLSI circuit design, Ph.D. Thesis, I.I.T. Kharagpur, India, (1992).
13. K. Sutner, The σ game and cellular automata, *American Mathematical Monthly* **97** (1) (January 1990).
14. R. Barua and S. Ramkrishnan, σ game, σ^+ game and two dimensional additive cellular automata, Personal Communication.
15. K.B. Datta, *Matrix and Linear Algebra*, Prentice Hall, India, (1991).
16. B. Elspas, The theory of autonomos linear sequential networks, *TRE Trans. on Circuits* **CT-6** (1), 45–60 (March 1959).
17. D.E. Knuth, *The Art of Computer Programming—Seminumerical Algorithms*, Addison-Wesley, (1981).
18. A. Strole and H.J. Wunderlich, TESTCHIP: A chip for weighted random pattern generation evaluation and test control, *IEEE Journal of Solid-State Circuits* **26** (7), 1056–1063 (July 1991).
19. W. Patterson, *Mathematical Cryptography*, Rowman and Littlefield, (1987).
20. Y. Fisher, *Fractal Image Compression*, Springer-Verlag, (1994).
21. Peitgen *et al.*, *Chaos and Fractals*, Springer-Verlag, (1992).
22. P.P. Chowdhury, On cellular automata of different dimensions and their applications, *Lecture at CSU seminar, ISI* (April 1994).