The Japan Society of Mechanical Engineers

B103 Hierarchical Cellular Automata Model for *VLSI* Testing

Biplab K Sikdar[1] **Niloy Ganguly**[2] **P Pal Chaudhuri**[1]

[1]Department of Computer Science & Technology, Bengal Engineering College (D U), India 711103 {biplab@,ppc@ppc.}becs.ac.in
[2]Computer centre, IISWBM, India 700073, n_ganguly@hotmail.com

Abstract: In recent years, the Cellular Automata (*CA*) technology has been gaining wider acceptance in various fields - image processing, language recognition, pattern recognition, VLSI testing, study of fractals and chaos, etc. The user community look forward for higher versatility and robustness of *CA* based models employed to study hierarchical systems. In this context, this paper introduces a new concept of Hierarchical Cellular Automata (*HCA*). Theory of *HCA* is developed over the Galois extension field $GF(2^{p^{q^{r}}})$, where each *HCA* cell can store a symbol from the set $\{0,1,2,\cdots,2^{pqr-1}\}$. The potential application of *HCA* has been demonostrated for test solution of *VLSI* circuits specified in hierarchical structural description. The test solution is designed to meet the test requirements of the circuit by exploiting its structural dependencies.

1 Introduction

In the mid of twentieth century, many people in Artificial Life have been enamoured of a computing model known as the Cellular Automata (*CA*). The *CA* is sufficiently complex to develop an entire universe as sophisticated as the one in which we live. Von Neumann [1] envisioned the modeling of self-reproducing automata empowered to simulate the bacterial growth, the growth of patterns on seashells, fluid dynamics, and the voting patterns of individuals who made decisions based on their local neighbors.

One of the most important milestones in the history of development of the simple homogeneous structure of cellular automata is due to Wolfram [2]. He proposed one/two dimensional structure of simple cells, each having only two states with uniform three-neighborhood dependence. This simplified structure motivated a number of researchers [3] to explore innovative applications of the *CA* machine in various fields - image processing, language recognition, pattern recognition, testing of *VLSI* circuits, study of fractals and chaos, etc [3]. All these computing models employ $GF(2)$ cellular automata where each cell is capable of storing either 0 or 1 ∈ GF(2).

While developing *CA* based models we observed that the power of $GF(2)$ *CA* is inadequate to handle complex physical systems which by nature exhibits abstraction and hierarchy in a fundamental manner. For example, a set of proteins controls the growth and activities of every living organism. A protein is a linear sequence of amino acids in a polypeptide chain. An amino acid, in turn, is derived from a codon which is a triplet of nitrogenous bases (Adenine, Cytosine, Guanine, and Thymine). Thus, the bottom up hierarchical structure of a protein starts from nitrogenous bases, followed by a sequence of codons/amino acids that leads to a *protein*. On the otherhand, while designing a *VLSI* circuit we fall back on top down hierarchy. At the highest level the *VLSI* chip is a collection of subsystems, each subsystem being designed with a network of modules. Each module in turn is a collection of submodules that realize the function of module. In general, a system when viewed at different levels of hierarchy & abstraction offers better insight into the system behavior. This establishes the need of a modeling tool to exploit the natural hierarchy of a system - observed in nature or artificially built.

In the above background scenario, we have introduced the concept of Hierarchical Cellular Automata (*HCA*) as a modeling tool. The theory of *HCA* machine is developed over the Galois extension field $GF(2^{p^{q^{r}}})$, where each *HCA* cell can store a symbol from the set $\{0,1,2,\cdots,2^{pqr-1}\}$. From the theoritical view point, the extension field structure of $GF(2^{p^{q^{r}}})$ is isomorphic to that of $GF(2^{pqr\cdots})$. However, for engineering applications the hierarchical field structure of $GF(2^{p^{q^{r}}})$ can be effectively employed to model the inherent hierarchy available in a physical system. The potential application of *HCA* has been demonostrated for an engineering application - *VLSI* testing.

The preliminaries on $GF(2)$ cellular automata and the *VLSI* test problem are introduced in *Section 2*. *Section 3* introduces the *HCA* and its characterization. The characterization is based on the theory of Galois extension field. For the sake of completeness, a review of the theory of extension field is also covered in this section. In *Section*

日本機械学会〔No.01-63〕 セルオートマトン・シンポジウム講演論文集〔2001.11.15,16,横浜〕

NII-Electronic Library Service

4, we present the application of *HCA* in designing the hierarchical Test Pattern Generator (*HCATPG*), customized for a *VLSI* circuit, as an effective solution to the *VLSI* testing. The efficiency of the proposed design is established through exhaustive experimentation.

2 Preliminaries

This section introduces $GF(2)$ *CA* and the *VLSI* test problem that will be addressed in this paper.

2.1 $GF(2)$ Cellular Automata

The Cellular Automata (*CA*) is the simplest model of decentralized spatially extended system, made up of a number of individual components (cells). The state q of each individual unit (cell) changes over time depending on the states of its neighbors [2]. For $q \in \{0,1\}$, the *CA* is referred to as $GF(2)$ *CA*. In 3-neighborhood (left, right and self) dependence, the state q of the i^{th} cell at time $(t+1)$ is

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

where q_i^t denotes the state of the i^{th} cell at time t and f is the next state function called the 'Rule' of the automata [2].

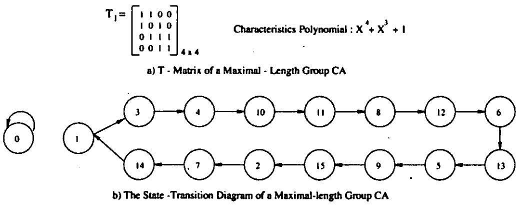

Fig. 2.1: A 4-cell maximal length group *CA* state transition

Definition 1 *If the next-state generating logic of CA employs only XOR then it is called a linear rule. A CA with all the cells having linear rules is called a linear CA.*

An n-cell linear $GF(2)$ *CA* is characterized by its characteristic matrix $[T]_{n \times n}$ [3], where

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

The polynomial of which T is a root is called the *characteristic polynomial* of the *CA*. The characterization of $GF(2)$ *CA* behavior, from its T matrix and characteristic polynomial, has been reported in [3].

Definition 2 *If all the states in the state transition graph of a CA lie in some cycles, it is called a* group CA*; otherwise it is a* non-group CA. *For a group CA,* $det[T] \neq 0$.

The group *CA* is classified as maximal and non-maximal length *CA*. In an n-cell maximal length *CA (Fig.2.1)* there is a cycle of length $2^n - 1$ with all non-zero states.

2.2 The *VLSI* Test Problem

Rapid advances in semiconductor technology have made possible fabrication of complex *VLSI* circuits within feasible cost. However, the problem of testing *VLSI* circuits has become a major cause of concern. In recent years, the test solution incorporating *BIST* (Built-In Self Test) methodology has been gaining increasing popularity with the test design community. Linear feedback shift registers (*LFSR*s) [4] are extensively used as the *BIST* Test Pattern Generators (*TPG*s). A wide variations of such structures have also been proposed [5, 6]. In last one decade, the simple and regular structure of Cellular Automata (*CA*) has been getting acceptance as an alternative to *LFSR* [7, 3].

BIST schemes are aimed to meet the basic requirements of high fault coverage with minimal test application time and low overhead. However, desired level of fault coverage for any arbitrary random logic is difficult to achieve with the conventional *BIST* structures built around *CA/LFSR*. These are typically designed without due consideration for the structure of the given *CUT* (Circuit Under Test).

The current *VLSI* chips associated with *SOC* (system on chip) application are becoming increasingly complex day by day. While designing test solution, such circuits should be viewed as a hierarchical (verilog/*VHDL*) structure. An efficient *BIST* scheme for these circuits should exploit its hierarchical structure. The Hierarchical Cellular Automata (*HCA*) can be employed as an effective tool to model the desired *BIST* structure for this class of circuits.

3 Hierarchical Cellular Automata

The class of *CA* dealt with in [3] is defined over Galois field $GF(2)$ and can handle the elements from the set $\{0,1\}$. By contrast, each cell in a *HCA*, defined over Galois extension field $GF(2^{p^{q^{r\cdots}}})$, can store a symbol $\in \{0,1,2,3, \cdots, (2^{pqr\cdots} - 1)\}$.

3.1 Preliminaries of Extension Field

For any positive integer p, it is possible to extend the Galois field $GF(m)$ to a field with m^p elements, where m is a prime number. This field is known as an extension field of $GF(m)$ and denoted as $GF(m^p)$ [8].

There exists an element α in the extension field $GF(2^p)$ that generates all the non-zero elements $(\alpha, \alpha^2, \ldots, \alpha^{2^p-1})$ of the field. The α is referred to as the *generator*. It is the root of an irreducible polynomial $A(x)$, known as the *generator polynomial* $A(x)$ of extension field. The coefficients of $A(x) \in GF(2)$. If $A(x)$ is a primitive polynomial, all the 2^p elements of the extension field are distinct.

Definition 3 *Primitive Element: The element* $\alpha^i \in GF(2^p)$ *is called the primitive element of* $GF(2^p)$ *if it is a root of any primitive polynomials in degree p.*

Representations of $GF(2^p)$ field elements: The element $\alpha \in GF(2^p)$ can be represented as a vector or by a $p \times p$ matrix M having its elements $0, 1 \in GF(2)$. The characteristic polynomial of M is the generator polynomial $A(x)$. The *matrix representation* of an element α^j $(j = 2, 3, \cdots, (2^p-1))$ is given by M^j. A column vector of M^j can be used as its vector representation [9]. The decimal

NII-Electronic Library Service

counterpart of it denotes the element of $GF(2^p)$. *Fig.3.1* illustrates the $GF(2^2)$ elements.

Operations in extension field: The *star_table* and *plus_table* guide the multiplication and addition operations respectively on $GF(2^p)$ elements. These tables correspond to the generator polynomial $A(x)$. The *star* and *plus* tables for x^2+x+1 are shown in *Table 3.1*. The first row and column represent the $GF(2^2)$ elements in decimal notation.

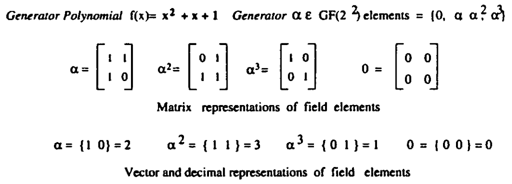

Fig. 3.1: The field elements of $GF(2^2)$

Table 3.1: The star_table and plus_table for x^2+x+1

*	0 1 2 3
0	0 0 0 0
1	0 1 2 3
2	0 2 3 1
3	0 3 1 2

+	0 1 2 3
0	0 1 2 3
1	1 0 3 2
2	2 3 0 1
3	3 2 1 0

Extension of extension field: The extension field $GF(2^{pq})$ over $GF(2)$, where p & q positive integers, can be represented as $GF(2^{p^q})$ - that is, the *extension of extension field* $GF(2^p)$ with same number of 2^{pq} elements from the set $\{0,1,2,\cdots,(2^{pq}-1)\}$ and isomorphic field operators *plus* & *star*. Similarly, the extension field $GF(2^{pqr\cdots})$ over $GF(2)$ can be viewed as the extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$; $p, q, r, ..$ are the positive integers. For example, $GF(2^6)$ can be viewed as the extension of extension field $GF(2^2)$ or $GF(2^3)$ - that is, as $GF(2^{2^3})$ or $GF(2^{3^2})$.

In $GF(2^{p^q})$ the co-efficients of the generator polynomial $B(x) \in GF(2^p)$. The generator β can be represented by a $q\times q$ matrix with its elements $\in GF(2^p)$ field having α as its generator. The α in turn, as noted earlier, can be represented by a $p\times p$ binary matrix; in effect the elements of extension field $GF(2^{p^q})$ are hierarchically partitioned.

Note: From the point of extension field theory the $GF(2^{pq})$ and $GF(2^{p^q})$ are isomorphic. However, for the current engineering application dealing with hierarchical circuit structure, the hierarchical partitioning of field elements has distinct advantages.

3.2 Hierarchical CA Structure

Fig.3.2.1 shows the general structure of an n-cell hierarchical $GF(2^{p^{q^{r^{\cdot\cdot}}}})$ *CA*. The interconnection among the cells are weighted in the sense that to arrive at the next state $q_i(t+1)$ of the i^{th} cell, the present states of $(i-1)^{th}$, i^{th} and $(i+1)^{th}$ are multiplied respectively with w_{i-1}, w_i and w_{i+1} and then added. w_i's $\in \{0,1,2,3,\cdots,(2^{pqr\cdots}-1)\}$ are the elements of extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$.

In $GF(2)$ *CA*, p=q=r...=1, a cell consists of one memory element (FF) and w_is can be either 0 (not connected) or 1 (connected). *Fig.3.2.2* depicts the structure of a *CA* cell in three level of hierarchy - $GF(2^{p^{q^r}})$. Each cell of the *HCA* consists of r number of subcells. A subcell in turn contains q number of next lower level subcells, each having p FFs.

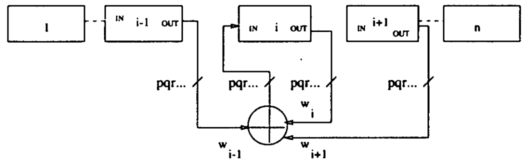

Fig. 3.2.1: General structure of an Hierarchical *CA*

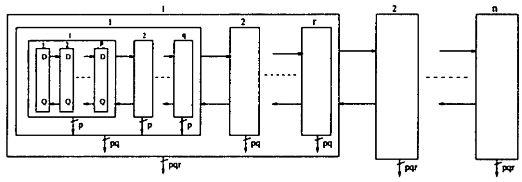

Fig. 3.2.2: Structure of a $GF(2^{p^{q^r}})$ Hierarchical *CA* cell

As in $GF(2)$ *CA* an n-cell hierarchical *CA* is characterized by an $n\times n$ characteristic matrix T, where

$$T_{ij} = \begin{cases} w_{ij}, & \text{if the next state of the } i^{th} \text{ cell depe-} \\ & \text{nds on the present state of the } j^{th} \\ & \text{cell by a weightage } w_{ij} \in GF(2^{p^{q^{r}}}) \\ 0, & \text{otherwise.} \end{cases}$$

The next state (pattern) X_{next} of a *HCA* can be derived as $X_{next} = T \times X_{current}$, where X_{next} and $X_{current}$ are n-symbol strings. An example $GF(2^2)$ *CA* with single level hierarchy follows for the sake of illustration.

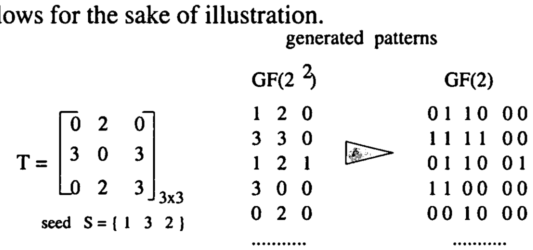

Fig. 3.2.3: The patterns generated by a 3-cell $GF(2^2)$ *CA*

Example 1 *Let us consider the example 3-cell* $GF(2^2)$ *CA of* Fig.3.2.3. *The* $[T]_{3\times3}$ *matrix defines the interconnection among the CA cells. The patterns generated by the HCA with seed* $S = \{1\ 3\ 2\}$ *are also shown in the figure.*

3.3 Design of Hierarchical *CA*

The design of a hierarchical *CA* boils down to construction of T matrix of the *CA*. It involves two steps: (i) formation of the dependency matrix (D) of the *HCA* that identifies dependencies of one *CA* cell on its neighbors, and (ii) specification of the weight values (w_i's) of the dependencies. The dependency matrix of the 3-cell $GF(2^2)$ *HCA* of *Fig.3.2.3* is given by, $D = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$, where $D_{ij} \in GF(2)\ \forall\ i,j$. The characteristic matrix T, as shown in *Fig.3.2.3*, can be generated from D by specifying the weights of the dependencies - that is, by replacing 1's of D with the non-zero weight values.

NII-Electronic Library Service

3.4 Characterization of *HCA*

Depending on the design, a *HCA* can show cyclic or non-cyclic behavior as in *GF*(2) *CA*.

Theorem 1 *The hierarchical CA with characteristic matrix T is a group CA iff* $det[T] \neq 0$. *A HCA with* $det[T] = 0$ *is a non-group CA.*

The proof is analogous to that reported in [3] for *GF*(2) *CA*.

Theorem 2 *A HCA, having same non-zero weight in each row (column) of its T matrix, is a group CA iff* $det[D] \neq 0$, *where D is the dependency matrix.*

Proof: Consider an n-cell $GF(2^{p^{q^{r^{..}}}})$ *HCA* with dependency matrix *D*. Assume, its *T* matrix is such that all the non-zero elements in each column are the same and in *column i* it is w_i ($w_i \in GF(2^{p^{q^{r^{..}}}})$). If $v_1, v_2, \cdots, v_n$ are the column vectors of T, then -
$det[T] = det[w_1v_1, w_2v_2, \cdots, w_nv_n]$
$= w_1w_2\cdots w_n\ det[v_1, v_2, \cdots, v_n] = w_{n+1}\ det[D]$, where $w_{n+1} = w_1w_2\cdots w_n \in GF(2^{p^{q^{r^{..}}}})$ and $w_{n+1} \neq 0$. Therefore, the value of $det(T) \neq 0$, iff $det[D] \neq 0$. Hence, as per *Theorem 1*, the *HCA* is a group *CA*.

Cycle structure: For the *HCA* of *Fig.3.2.3* having equal non-zero weights 3, 2, and 3 respectively in *column* 1, 2, and 3 of its *T* matrix, the $det[D] = 1$. It is a group *CA*. In general, the cycle structure of a group *CA* is defined as $[\mu_1(k_1), \mu_2(k_2), \cdots]$ - that is, it has μ_i number of cycles of length k_i, $\forall i$. The cycle structure of the *CA* of *Fig.3.2.3* is [1(1),1(3),4(15)] - it has 1 cycle of length 1, one of length 3, and 4 cycles of length 15.

Property 1: A *HCA* designed with primitive weight while satisfying *Theorem 2* generates larger length cycles.

This has been illustrated in *Table 3.4*. The first column shows the number of cells (*n*) and the extension field parameter (*p*) of different $GF(2^p)$ *CA*. The *HCA* (with one level of hierarchy) for a particular *n* & *p* are designed for a large number of trials with primitive & non-primitive weight sets. Cycle lengths produced by the *HCA* for both the cases are noted in *Column 2* and 3. For a given pair of *n* and *p*, an entry *l* in *Column 2* [*Column 3*] reports that out of 100 randomly generated group *HCA* at least 25 of such *HCA* produce cycles of length $\geq l$.

Table 3.4: Cycle structure: primitive vs non-primitive weights

HCA	cyl len (*l*) with prob. of occurrence 25%	
(n, p)	primitive weight	non-prim. weight
5, 8	331672625	65535
4, 8	131070	43690
6, 6	524286	58254
5, 6	17039295	298935
3, 6	262143	262143
8, 4	131070	8190
7, 4	1052415	65535
3, 4	4095	255

Theorem 3 *A* $GF(2^{p^{q^{r^{..}}}})$ *HCA designed with same non-zero weight does not produce maximal length cycle for* $(pqr..) > 1$.

Proof: Without loss of generality let us assume that, T_{eq} be the characteristic matrix of a $GF(2^p)$ *HCA* designed with same non-zero weight *w* and $p > 1$.

Case I: $det[T_{eq}] = 0$: The *HCA* is a non group *CA* (*Theorem 1*) and thus can't generate maximal length cycle.

Case II: $det[T_{eq}] \neq 0$: The *HCA* is a group *CA* (*Theorem 1*). If it is a maximal length *CA*, then

$$(T_{eq})^{2^{np}-1} = I \qquad (1)$$

By elementary row and column operations it can be written as $T_{eq} = wD$, where *D* is the dependency matrix of the *HCA* having non-zero elements as 1.
Since, $det[T_{eq}] \neq 0$, $det[wD] \neq 0$
that is, $w\ det[D] \neq 0 \Rightarrow det[D] \neq 0$.
Therefore, $(D)^{2^n-1} = I$, where *D* is a matrix in *GF*(2). From *equation 1*, $[wD]^{2^{np}-1} = I$ i.e., $w^{2^{np}-1} \times (D^{2^{np}-1}) = I$. As $w^{(2^p-1)} = 1$ ($w \in GF(2^p)$ & '1' is the identity element in $GF(2^p)$) and $(D)^{2^n-1} = I$, then
$[wD]^k = I$, where $k = lcm(2^p-1, 2^n-1)$. It implies, $[T_{eq}]^k = I$. But, as $k < (2^{np}-1)$, T_{eq} can't generate maximal length cycle of length $(2^{np}-1)$.

Theorem 4 *If an n-cell HCA with characteristic matrix T has cycle structure* $[1(1), \mu_1(m)]$, *where* $\mu_1 m = (2^{npqr..}-1)$, *and p,q,r, ... are the extension field parameters, then the HCA designed with* $[T]^\mu$ *has cycles* $[1(1), \mu_1\mu(k)]$, *where* $\mu k = m$.

Proof: From the cycle structure of *T* it is obvious that for any non-zero state *x*, $[T]^m$ x = x, that is $[T]^m = I$. Now, if $\mu k = m$, then $[T]^{\mu k} = [T^\mu]^k = I$, which means $[T^\mu]$ may have cycle of length *k* or factors of *k*. Let us assume that $[T^\mu]$ has cycle of length k_1, where k_1 is a factor of *k* and k_1k_2 = k.
Then for some state x_1, $[T^\mu]^{k_1}x_1 = x_1$, that is
$[T^\mu]^{k/k_2}x_1 = [T^{\mu k}]^{1/k_2}x_1 = [T^m]^{1/k_2}x_1 = x_1$
But, $[T]^m x = x$, $\forall x$. Therefore, $k_2 = 1$
Hence, $[T]^\mu$ has the cycle structure as $[1(1), \mu_1\mu(k)]$ since *T* has the cycle structure $[1(1), \mu_1(m)]$.

Lemma 1: If β is a primitive element in the extension field then $\beta^2, \beta^4, \beta^8, \cdots$ are also primitive

Proof: Without loss of generality let us assume that β is a primitive element in $GF(2^{p^q})$, *p* and *q* being the extension field parameters. β is primitive - that is, the $\beta, \beta^2, \beta^3, \cdots, \beta^{(2^{pq}-1)}$ all are unique and the cycle structure of β is [1(1), 1(m)], where $m = (2^{pq}-1)$. From *Theorem 4* it follows that β^μ has the cycle structure [1(1), μ(k)], where $\mu k = m$, as $\mu_1 = 1$. Since, *m* is odd, then 2^i will be prime to *m* for i = 1, 2, 3, ... , (pq-1). Hence, β^{2^i} has the cycle structure as [1(1), 1(m)] - that is, β^{2^i} is primitive.

3.5 Randomness Property of *HCA*

The randomness property of the *HCA* based pattern generator, formed for different values of *p*, *q*, and *n*, are studied based on the metric proposed in *DiehardC* [10].

NII-Electronic Library Service

A comparative study on randomness quality of maximal length $GF(2)$ *CA*, $GF(2^p)$ *CA* and $GF(2^{p^q})$ *CA* is presented in *Table 3.5*.

Table 3.5: Randomness Test

Randomness Test	Pattern size $N = 32$		
	$GF(2)$	$GF(2^4)$	$GF(2^{2^4})$
Overlap Sum	pass	pass	pass
Run	pass	pass	pass
3Dsphere	fail	pass	pass
Parking lot	fail	fail	pass
B'day spacing	fail	pass	pass
Count 1's	fail	pass	pass
B rank(6x8)	fail	pass	pass
B rank(31x31)	fail	fail	pass
B rank(32x32)	fail	fail	pass
Count 1's(byte)	fail	fail	pass
Bit stream	fail	pass	pass
Craps	pass	pass	pass
Minimum Dist	fail	pass	pass
Overlap 5-permut	pass	fail	pass
OPSO	fail	fail	pass
OQSO	fail	fail	pass
DNA	pass	pass	pass
Squeeze	fail	pass	pass

The 18 different tests are shown in *Column 1* of the table. Each test produces a set of 'p' values. For a pattern set with good randomness quality, the values of p's will be uniformly distributed between 0.001 and 0.999. The entry *pass*, in a column, means that the 'p' value is evenly distributed on [0,1] for at least 75% cases of total number of seeds tried for a pattern generator. The results of *Table 3.5* report that the randomness quality of the patterns generated by the *HCA* is better than that of $GF(2)$ *CA*.

4 Design of Customized *HCATPG*

The proposed scheme extracts the clustering of *primary inputs* (*PI*s) and the structural dependencies among the different *hierarchical* modules of the *CUT*. Consider the example circuit of *Fig.4* with 36 primary inputs (*PI*s). To design a conventional pseudo-random *TPG* for this circuit we can use a 36-bit length *LFSR*/$GF(2)$ *CA*. In practice, for all types of *CUT* with 36 *PI*s, the same pseudo-random pattern generator or its variations are used as the *TPG*. It can be seen that the *PI*s of *Fig.4* get grouped into four 9-bit buses (*A*, *B*, *C*, *and E*). It is logical to assume that all the 36 *PI*s of the *CUT* are not independent so far as their functionality is concerned. The 9 *PI*s of type *A* input to module M_1 are functionally similar and can be considered to form a *cluster* of *PI*s rather than 9 independent *PI* lines - each carrying a single bit. The similar consideration is valid for *B*, *C* and *E*. The guiding motivation is - instead of feeding the *PI*s of a *cluster* of 9-bit bus from 9 cells of a 36-cell $GF(2)$ *CA*, we propose to feed the *cluster* from a cell of the 4-cell $GF(2^9)$ *CA*.

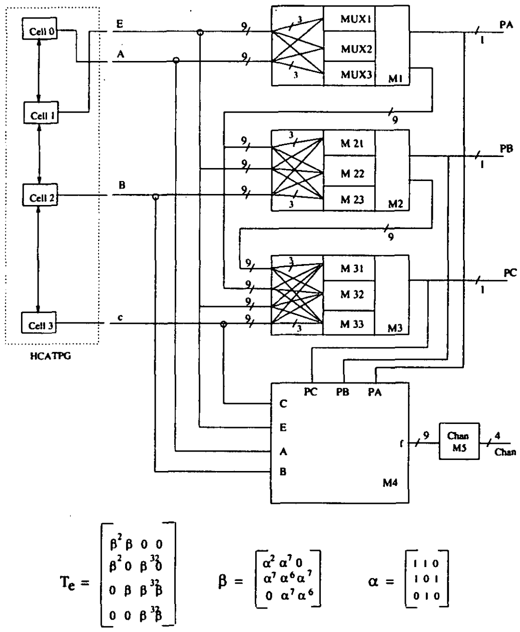

Fig. 4: High-level model for a *CUT* & the *HCATPG*

Next, on further analysis of the module M_1, it can be observed that the 9-members of *cluster A* [*E*] are divided in three groups (*sub-clusters*), each group containing 3 *PI*s and fed into three different sub-modules (MUX_1, MUX_2, and MUX_3). Similar cases can be observed for other blocks (M_2, M_3,...). It is very much logical to assume that within a *cluster*, the intra *sub-cluster PI*s are functionally more closer in comparison to the inter *sub-cluster PI*s. The functional dependencies of the *PI*s in a *sub-cluster* are to be reflected in designing the *TPG* cells. This problem can be solved if the $GF(2^9)$ *CA* cell is viewed hierarchically as a $GF(2^{3^3})$ *CA* cell. Without any loss of generality, we shall restrict to two level of hierarchy for designing *HCATPG*.

The major design steps to arrive at the desired customized *HCATPG* are as follows:
(1) Selection of p, q, and n for $GF(2^{p^q})$ *HCATPG*.
(2) Identifying the dependency matrix D.
(3) Designing the characteristic matrix T.

4.1 Selection of p, q and n

This step executes partitioning of primary inputs to form input *clusters* & *sub-clusters* based on their functional information and then identifies the cardinalities of the *clusters* & *sub-clusters*. The most frequent cardinality of the *clusters* $(c_1, c_2, \cdots, c_k)$ is chosen as the value of $(p \times q)$, whereas the most frequent cardinality of *sub-clusters* within a *cluster* is taken as the value of the parameter p. The value of n (number of *TPG* cells) is fixed as per the expression

$$n = \lceil(|c_1|/pq)\rceil + \lceil(|c_2|/pq)\rceil + \cdots + \lceil(|c_k|/pq)\rceil + \lceil(|restPIs|/pq)\rceil$$

For testing the circuit of *Fig*.4, the *HCATPG* is a 4-cell $GF(2^{3^3})$ *CA*. The cells are marked as 0, 1, 2, and 3.

NII-Electronic Library Service

4.2 Identification of Dependency matrix

If two *PI clusters* enter into the same circuit module, the structural dependencies is said to exist between them and these are referred to as *dependent clusters*. It is observed that the dependent clusters closely interact among themselves to detect the faults of the circuit module. Therefore, the *CA* cells feeding the *dependent clusters* must have dependencies among themselves. For *Fig.4*, the resulting *D* matrix is

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

where *PI* clusters *A*, *E*, *B*, and *C* are respectively fed from the Cells 0, 1, 2, and 3.

4.3 Design of Characteristic Matrix *T*

The non-zero value '1' of the *D* matrix is next replaced with a weight value (β^j) from the set W={β, β^2, β^4, $\cdots$}, to design the *T* matrix of the *HCA*. The basic algorithmic steps to arrive at the characteristic matrix *T* of the *HCATPG* are noted in [9]. In the design, to customize the *HCATPG*, the weights of *T* are tuned depending on the test requirement of the *CUT*. The tuning is performed in two steps: the *major* and *minor* tuning.

Major tuning: This step involves the fixing of the weight value (β^j) for a *HCATPG* cell. For example, to design the *T* matrix of the *HCATPG* for the *CUT* of *Fig.4*, the 1's of the D_e are replaced by the weights from the set $\{\beta, \beta^2, \beta^4, \cdots\}$. The T_e noted in *Fig.4* results in maximum fault coverage for the *CUT* and the weight values β^2, β, β^{32}, *and* β are fixed for the *HCATPG* cell 0, 1, 2, and 3 respectively.

Minor tuning: The β^j_{matrix} is a $q \times q$ matrix having its elements from the set $\{0, \alpha, \alpha^2, \cdots, \alpha^{(2^p-1)}\} \in GF(2^p)$. The minor tuning of *TPG* is done by structural modifications of the weights fixed through major tuning - that is, β^j is fine tuned by changing the relative positions of α^is within the β^j to improve the fault efficiency of the design. The final structure of the weight β for the example design is shown in *Fig.4*.

4.4 Experimental Results

The basic requirement for the proposed customized on-chip test pattern generator is the availability of the hierarchical structural description of the *CUT*. The circuits specified in *Column 1* of *Table 4.4* are designed from *ISCAS & ITC* benchmark circuits to get the hierarchical structural net lists. *Table 4.4* depicts the summary of the fault coverage shown by $GF(2)$ *CA* and $GF(2^{p^q})$ *HCA* based designs in *columns 4 and 5* respectively. A *CUT* is tested for three seeds (taken randomly) for both the designs. The fault coverage figures for a *CUT* are achieved on applying a fixed number of test vectors mentioned under the column heading *Test Vec.* Detail experimental setup is noted in [9]. *It is observed that the HCATPG resulted in higher fault coverage than that could be achieved with $GF(2)$ CA based TPG.*

Table 4.4: Test results of customized *HCATPG*

Circuit name	# PI/PO	Test Vec.	Fault coverage (%) *HCA*	Fault coverage (%) *GF*(2)
c1	36/7	400	99.18	98.57
c2	32/32	80	99.56	99.48
c3	41/32	450	98.95	98.29
c4	33/25	3000	99.12	98.76
c5	67/48	600	99.34	99.34
c6	82/64	700	98.95	98.81
c7	72/14	400	99.08	98.47
s1	24/4	2500	99.28	98.75
s2	23/24	1400	97.76	97.20

c represents combinational, s represents sequential

5 Conclusion

The paper presents an innovative concept of hierarchical cellular automata. The theory of *extension field* is utilized in designing the hierarchical *CA* (*HCA*). The *HCA* can be employed to model inherent hierarchy within a physical system. The eficiency of the *HCA* machine is tested in an engineering application - *VLSI* testing.

References

[1] J. V. Neumann, *The theory of self-reproducing Automata, A. W. Burks ed.* Univ. of Illinois Press, Urbana and London, 1966.

[2] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, pp. 601–644, July 1983.

[3] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chatterjee, *Additive Cellular Automata – Theory and Applications*, vol. 1. IEEE Computer Society Press, California, USA, ISBN 0-8186-7717-1, 1997.

[4] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in test for VLSI : Pseudo-random Techniques.* John Wiley & Sons.

[5] D. K. Pradhan and M. Chatterjee, "GLFSR–A New Test Pattern Generator for Built-in-Self-Test," *IEEE Transactions on Computer -Aided Design of Integrated Circuits and Systems*, vol. 18, no. 2, pp. 319–328, 1999.

[6] J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated synthesis of large phase shifters for built-in self-test," *International Test Conference*, pp. 1047–1056, 1998.

[7] P. D. Hortensius *et al.*, "Cellular automata based pseudo-random number generators for built-in self-test," *IEEE Trans. on CAD*, vol. 8, pp. 842–859, August 1989.

[8] T. R. N. Rao and E. Fujiwara, *Error-control Coding for Computer Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1989.

[9] B. K. Sikdar, P. Majumder, M. Mukherjee, N. Ganguly, D. K. Das, and P. P. Chaudhuri, "Hierarchical cellular automata as an on-chip test pattern generator," in *Proc. of 14th Int. Conf. on VLSI Design, India*, January 2001.

[10] "http://stat.fsu.edu/ geo/,"

NII-Electronic Library Service