Design of Hierarchical Cellular Automata for On-Chip Test Pattern Generator

Biplab K Sikdar, Niloy Ganguly, and P Pal Chaudhuri

***Abstract*—This paper introduces the concept of hierarchical cellular automata (HCA). The theory of *HCA* is developed over the Galois extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$, where each cell of the *CA* can store and process a symbol in the extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$. The hierarchical field structure of $GF(2^{p^{q^{r^{\cdot\cdot}}}})$ is employed for design of an HCA-based test pattern generator (HCATPG). The HCATPG is ideally suited for testing very large scale integration circuits specified in hierarchical structural description. Experimental results establish the fact that the HCATPG achieves higher fault coverage than that which could be achieved with any other test structures. The concept of percentile improvement in fault coverage is introduced to have a realistic assessment of fault coverage achieved with the proposed HCATPG.**

I. Introduction

With the growing complexity of present day very large scale integration (VLSI) circuits, the cost of testers (ATE) has become a significant part of the overall product cost. Built-in self-test (BIST) has emerged as an alternative. It reduces the test cost of VLSI designs. A typical BIST structure includes an on-chip test pattern generator (TPG) and signature analyzer (SA). Linear feedback shift registers (LFSRs) [1] and cellular automata (CA) [2] are extensively used as the test pattern generators. Wide variations of such structures have been proposed in [3]–[5]. Alternative schemes [6], [7] implemented with arithmetic pattern generators/compactors have been also reported.

The *BIST* schemes are aimed to meet the basic requirements of high fault coverage with minimum test application time and low overhead. However, desired fault coverage for any arbitrary random logic is difficult to achieve with BIST methodologies proposed so far. The GLFSR [8] and *phase-shift LFSR*-based [9] pseudorandom pattern generators are proposed to improve the fault efficiency. The basic limitation of the conventional on-chip TPG structures is that they are designed without due consideration to the structure of the given circuit under test (CUT).

The BIST schemes targeting analysis of logic circuits from their behavioral and RTL descriptions are reported in [10]–[13]. However, these schemes impose a number of design restrictions and are suitable only for a specific class of logic circuits.

In this background, we report an efficient methodology of a BIST scheme that employs hierarchical cellular automata (HCA)-based test pattern generator (HCATPG). The class of CA dealt with in [14] is defined over Galois Field $GF(2)$ which can handle the elements from the set {0, 1}. By contrast, each cell in a HCA, defined over Galois extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$, can store a symbol belonging to $\{0, 1, 2, 3, \ldots, (2^{pqr\cdot\cdot} - 1)\}$.

In the proposed test architecture, the TPG is customized for a CUT by tuning the parameter values p, q, r, … of the HCA used for the design. The value of extension field parameters and interconnection among the CA cells of HCATPG are fixed on the basis of natural clustering of primary inputs (PIs) to different hierarchical *RTL/functional* blocks of the CUT. The structural analysis of the subset of PIs input to different subcircuits has enabled HCATPG to achieve higher fault coverage for a CUT. The *TPG folding* scheme [15] can be employed to reduce the overhead of HCATPG.

Design of HCATPG for a given CUT is reported in Section IV after an introduction of extension field preliminaries in Section II and HCA in Section III. The concept of percentile improvement in fault coverage (PIFC) has been introduced in Section IV to have a realistic assessment of the proposed design.

Manuscript received June 5, 2001; revised May 22, 2002. This paper was recommended by Associate Editor S. Reddy.

B. K Sikdar and P P. Chaudhuri are with the Department of Computer Science and Technology, Bengal Engineering College (D.U), Howrah, West Bengal, 711103 India (e-mail: biplab@cs.becs.ac.in; ppc@cs.becs.ac.in).

N. Ganguly is with the Computer Centre, IISWBM, Calcutta, West Bengal, 700073 India (e-mail: n_ganguly@hotmail.com).

Digital Object Identifier 10.1109/TCAD.2002.804380

II. Extension Field Preliminaries

There exists an element α in the extension field $GF(2^p)$ that generates all the nonzero elements $(\alpha, \alpha^2, \ldots, \alpha^{2^p-1})$ of the field. α is the *generator* and the irreducible polynomial of which α is a root is called the *generator polynomial* $A(x)$ (coefficients of $A(x) \in GF(2)$) [16]. We choose primitive polynomials as the generator polynomial so that all the 2^p elements of the extension field are distinct.

The element α can be represented 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 $(j = 2, 3, \ldots, (2^p - 1))$ is given by M^j. A column vector of M^j can be used as its vector representation [15], [17]. The addition ($\oplus$) and multiplication (*) of $GF(2^p)$ elements follow the *plus* and *star* tables corresponding to the generator polynomial of the field.

Example 1: Fig. 1 illustrates the elements of $GF(2^2)$ with $x^2 + x + 1$ as the generator polynomial. The matrix representation of the generator α is given by $\alpha_{\text{matrix}} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$; $\alpha_{\text{vector}}(10) = 2$, the *last column* of α_{matrix}. The other elements of the field are computed from α

$$\alpha^2_{\text{matrix}} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad \alpha^3_{\text{matrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$\alpha^2_{\text{vector}} = (11) = 3, \quad \alpha^3_{\text{vector}} = (01) = 1.$$

The star and plus tables are shown in Table I. The first rows and the first columns of the tables represent the $GF(2^2)$ elements in decimal notation.

Extension of Extension Field: The extension field $GF(2^{pq})$ over $GF(2)$, where p and q both are the 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 $\{0, 1, 2, \ldots, (2^{pq} - 1)\}$ and isomorphic field operators. Similarly, the extension field $GF(2^{pqr\cdot\cdot})$ can be viewed as the extension field $GF(2^{p^{q^{r^{\cdot\cdot}}}})$. For example, the Galois extension field $GF(2^6)$ can be also 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 coefficients of the generator polynomial $B(x)$ belong to $GF(2^p)$. The generator β can be represented by a $q \times q$ matrix with the elements in $GF(2^p)$. α is the generator of $GF(2^p)$. An element of $GF(2^p)$ in turn can be represented by a $p \times p$ binary matrix. In effect, the elements of extension field $GF(2^{p^q})$ are hierarchically partitioned.

Example 2: Fig. 2 illustrates the hierarchical partitioning of field elements in extension field. The structure of an element $\alpha_1(\text{generator}) \in GF(2^6)$ is shown in Fig. 2(a). The $GF(2^6)$ can be also viewed as $GF(2^{2^3})$ or $GF(2^{3^2})$. The hierarchical structural implementation of the elements belonging to these two fields are shown in Fig. 2(b) and (c), respectively. For generation of other nonzero elements of $GF(2^{2^3})$ from the generator β_1, the star and plus tables

Generator Polynomial A(x)= $\mathbf{x^2 + x + 1}$ *Generator* $\alpha \varepsilon$ GF(2^2) elements = $\{0, \alpha, \alpha^2, \alpha^3\}$

$$\alpha = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad \alpha^2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \alpha^3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad 0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Matrix representations of field elements

$$\alpha = \{1\ 0\} = 2 \quad \alpha^2 = \{1\ 1\} = 3 \quad \alpha^3 = \{0\ 1\} = 1 \quad 0 = \{0\ 0\} = 0$$

Vector and decimal representations of field elements

Fig. 1. The $\boldsymbol{GF(2^2)}$ field elements.

TABLE I
STAR_TABLE AND PLUS_TABLE FOR $\boldsymbol{GF(2^2)}$ FIELD ELEMENTS WITH $\boldsymbol{x^2 + x + 1}$ AS GENERATOR POLYNOMIAL

*	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

of $GF(2^2)$ (Table I) are to be considered. In the case of $GF(2^{3^2})$, the corresponding star and plus tables are of $GF(2^3)$ (Table II).

Definition 1: The element $\beta^i \in GF(2^{p^q})$ is called the primitive element of $GF(2^{p^q})$ if it is a root of any primitive polynomials in $GF(2^p)$ with degree q.

As per the theory of extension field, $GF(2^{pq})$ and $GF(2^{p^q})$ are isomorphic. However, for the current engineering application (dealing with hierarchical structure of a CUT), the hierarchical partitioning of field elements has distinct advantages. This property of hierarchical field structure is employed for the design of HCA introduced in the next section.

III. HIERARCHICAL CELLULAR AUTOMATA

Theory of $GF(2)$ CA has been dealt with in [14]. A cell of this class of *CA* stores and processes an element $0/1 \in GF(2)$ and hence is referred to as $GF(2)$ CA. Fig. 3 shows the general structure of an n-cell hierarchical $GF(2^{p^{q^{r^{\cdot^{\cdot}}}}})$ CA. For ease of presentation of the basic concept of HCATPG, in rest of the paper we shall deal with only two levels of hierarchy. Generalization for additional levels of hierarchy follows directly. In two levels of hierarchy, each cell of an n-cell $GF(2^{p^q})$ HCA consists of q number of subcells, each having p number of *FF*s (Fig. 4). The interconnection among the cells of the HCA are weighted (Fig. 3) in the sense that to arrive at the next state $q_i(t+1)$ of the ith cell, the present states of $(i-1)$th, ith, and $(i+1)$th cells are multiplied, respectively, with w_{i-1}, w_i, and w_{i+1} and then added. w_is $\in \{0, 1, 2, 3, \ldots, (2^{pq}-1)\}$ are the elements of extension field $GF(2^{p^q})$ and specify the weights of interconnection. The *multiplication* (*) and *addition* ($\oplus$) follow the *star* and *plus* tables defined in $GF(2^{p^q})$. The next state of the ith cell, in three-neighborhood dependence, is given by

$$q_i(t+1) = (w_{i-1}{}^* q_{i-1}) \oplus (w_i{}^* q_i) \oplus (w_{i+1}{}^* q_{i+1}).$$

Characterization: An n-cell HCA is characterized by an $n \times n$ characteristic matrix T, where

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

The T becomes a tridiagonal matrix for three-neighborhood dependence among the CA cells. The next state (pattern) X_{next} of a HCA, as illustrated in Example 3, can be derived as $X_{\text{next}} = T \times X_{\text{current}}$, where X_{next} and X_{current} are the n-symbol strings representing the states of the n-cell HCA.

Example 3: Let us consider the T matrix of a three-cell $GF(2^2)$ CA shown in Fig. 5. The $[T]_{3\times 3}$ matrix describes the interconnection among the CA cells of the HCA. Starting from seed $S = \{2\,3\,0\}$, the patterns generated out of the CA are shown in the figure.

State Transition Behavior: If all the states of an *HCA* lie on some cycles, then the HCA is referred to as a *group* CA. In a *nongroup* CA, the states form both cyclic and noncyclic structures.

Theorem 1: The hierarchical CA with characteristic matrix T is a group CA iff $\det[T] \neq 0$.

Proof: If the HCA under the operation with T forms a cyclic group, then there should exist an integer u such that

$$T^u = I \tag{1}$$

where I is the identity matrix and for any state $f(x)$

$$f_u(x) = T^u \cdot f(x) = f(x). \tag{2}$$

As per (1), the necessary condition for a cyclic group is existence of a "u" such that $T^u = I$, *that is*, $\det[T^u] = 1$.

If T is nonsingular, then $\det[T] = w_i$ ($w_i \in GF(2^{p^q})$). Now for some integer value of u, $1 \leq u \leq 2^{pq}-1$; $w_i^u = 1$, where "1" is the identity element of $GF(2^{p^q})$. Hence, if $\det[T] \neq 0$, there exists a u such that $\det[T^u] = [w_i]^u = 1$. This gives the necessary part.

The sufficient condition is proved with the following property of finite group of nonsingular square matrices. The group that we are interested in is the commutative subgroup generated by the powers of T. This is a cyclic subgroup with identity element I. The group is also Abelian. The order of this group is $n = 2^{(pq)^L}$, where L is the order of T. This is true because T is only a transformation matrix which maps one L length vector ($\in GF(2^{p^q})$) to another one of the same length and the number of such vectors can at most be $(2^{pq})^L = 2^{(pq)^L}$. ■

IV. DESIGN OF HCATPG

The hierarchical structural description of the CUT is assumed to be available for this design methodology. That is, the CUT is described as a network of modules/blocks, each of which in turn has a network of submodules/subblocks. The proposed scheme extracts the clustering of

$$\text{Generator } \alpha_1 = \begin{bmatrix} 0&1&0&0&0&0\\ 1&1&1&0&0&0\\ 0&1&0&1&0&0\\ 0&0&1&1&1&0\\ 0&0&0&1&0&1\\ 0&0&0&0&1&1 \end{bmatrix}_{6\times 6} \qquad \begin{array}{l}\text{generator polynomial}\\ x^6 + x^5 + x^3 + x^2 + 1\end{array}$$

(a)

$$\text{Generator } \beta_1 = \begin{bmatrix} 2&1&0\\ 1&3&1\\ 0&1&3 \end{bmatrix}_{3\times 3} = \begin{bmatrix} \begin{bmatrix}1&1\\1&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&0\\0&0\end{bmatrix}\\ \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&1\\1&1\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix}\\ \begin{bmatrix}0&0\\0&0\end{bmatrix} & \begin{bmatrix}1&0\\0&1\end{bmatrix} & \begin{bmatrix}0&1\\1&1\end{bmatrix} \end{bmatrix}_{6\times 6}$$

$$\begin{array}{l}\text{generator polynomial for } \beta_1\\ x^3 + 2x^2 + 2x + 2\\ \text{generator in GF}(2^2)\\ \alpha_2 = \begin{bmatrix}1&1\\1&0\end{bmatrix} = 2\\ \text{generator polynomial for } \alpha_2\\ x^2 + x + 1\end{array}$$

(b)

$$\text{Generator } \beta_2 = \begin{bmatrix} 0&6\\ 1&3 \end{bmatrix}_{2\times 2} = \begin{bmatrix} \begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} & \begin{bmatrix}0&0&1\\0&1&1\\1&1&0\end{bmatrix}\\ \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} & \begin{bmatrix}0&1&0\\1&1&1\\0&1&1\end{bmatrix} \end{bmatrix}_{6\times 6}$$

$$\begin{array}{l}\text{generator polynomial for } \beta_2\\ x^2 + 3x + 6\\ \text{generator in GF}(2^3)\\ \alpha_3 = \begin{bmatrix}0&1&0\\1&1&1\\0&1&1\end{bmatrix} = 3\\ \text{generator polynomial for } \alpha_3\\ x^3 + x + 1\end{array}$$

(c)

Fig. 2. Hierarchical structures of field elements. (a)Element of $GF(2^6)$ with $p = 6, q = 1$. (b) Structure of an element in $GF(2^{2^3})$ with $p = 2, q = 3$. (c) Structure of an element in $GF(2^{3^2})$ with $p = 3, q = 2$.

TABLE II
STAR_TABLE AND PLUS_TABLE FOR $GF(2^3)$ FIELD ELEMENTS WITH $x^3 + x + 1$ AS GENERATOR POLYNOMIAL

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	5	7	6	4	3	1
3	0	3	7	4	2	1	5	6
4	0	4	6	2	7	3	1	5
5	0	5	4	1	3	6	7	2
6	0	6	3	5	1	7	2	4
7	0	7	1	6	5	2	4	3

⊕	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

primary inputs (PIs) to different *hierarchical* modules and submodules of the CUT. We next introduce the concept of dependent PI clusters followed by the basic guiding principle of the design.

Definition 2: If two PI clusters enter into the same circuit module/submodule, the dependency is said to exist between them and they are referred to as dependent clusters.

Guiding Principle of the Design: Rather than considering PIs to the circuit at bit level, we can look at a subset of PIs as a cluster/subcluster of inputs to different *hierarchical modules* of the CUT and decide on the values of extension field parameters $p, q, r, \ldots$ of the HCA to be employed in the design of TPG. Further, the interconnection among the HCATPG cells exploits the dependencies of PI clusters to different circuit blocks. Hierarchical structure of $GF(2^{p^q})$ field elements (Fig. 2) supports the desired tuning of the *HCATPG* to realize the objective of enhanced fault coverage.

A. Tuning HCATPG for a CUT

The basic principle of the design is illustrated with the help of an example CUT of Fig. 6. It can be seen that the PIs of the CUT are grouped into four nine-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 nine PIs of type A, input to module M_1, are functionally similar and can be considered to form a *cluster* of PIs rather than nine independent PI lines, each carrying a single bit. Similar consideration is valid for B, C, and E. The guiding motivation is, instead of feeding the PIs 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 four-cell $GF(2^9)$ CA in a single level hierarchical implementation.

Next, on further analysis of the module M_1 of Fig. 6, it can be observed that the nine members of *cluster* $A[E]$ are divided into three groups (*subclusters*), each group containing three PIs and fed into three different submodules (MUX_1, MUX_2, and MUX_3). Similar cases can be observed for other blocks ($M_2,\ M_3,\ldots$). It is logical to assume that, within a *cluster*, the intrasubcluster PIs are functionally closer in comparison to the intersubcluster PIs. The functional dependencies of the PIs in a *subcluster* are to be reflected in designing the TPG cells. This problem for the example circuit of Fig. 6 can be solved if the $GF(2^9)$ CA cell is viewed hierarchically as a $GF(2^{p^q})$ HCA cell ($p = 3$ and $q = 3$ for the present example).

It is observed that the dependent PI clusters/subclusters closely interact among themselves to detect the faults of corresponding circuit block/subblock. Therefore, the CA cells feeding the dependent clusters/subclusters should have interdependence and so should be interconnected. That is, the tuning of HCATPG for a given CUT boils down to the following steps.

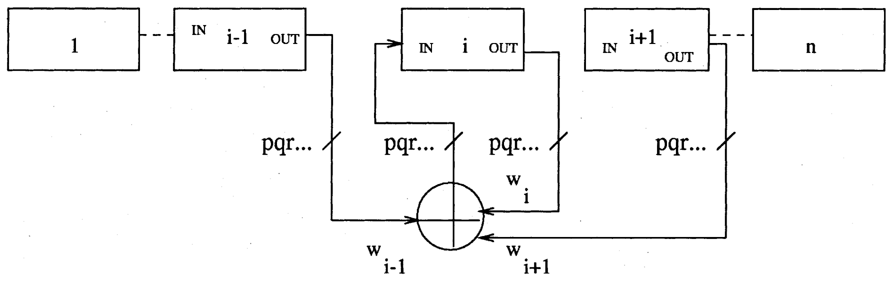
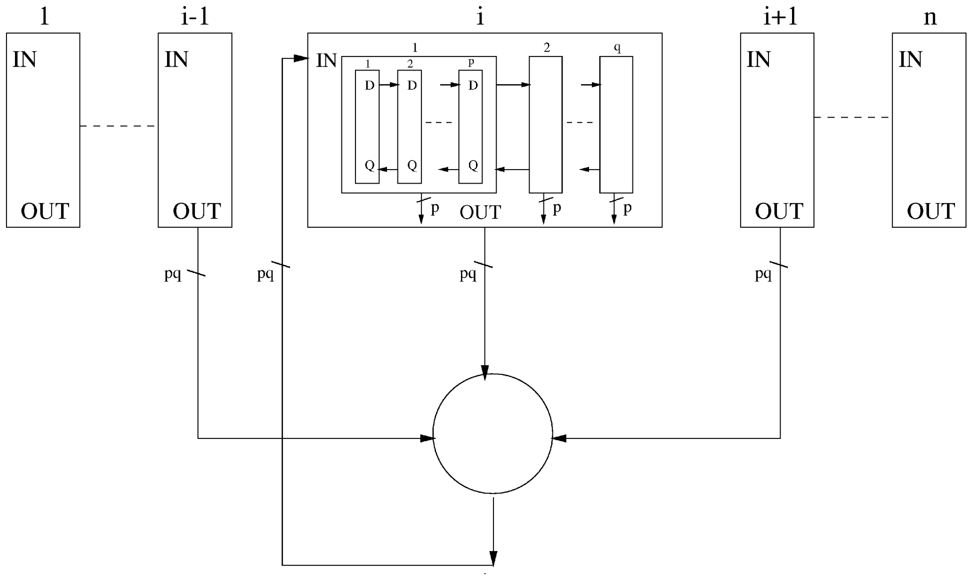
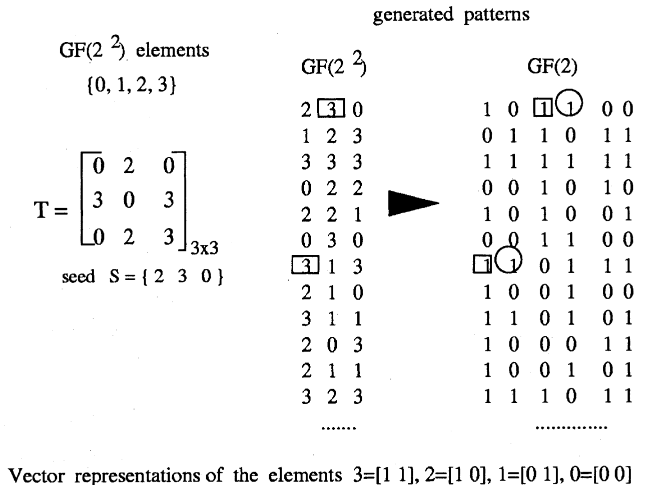


Fig. 3. General structure of an $\boldsymbol{n}$-cell HCA.



A GF (2^{p^q}) cell consists of q no. of sub-cells each having p FFs

Fig. 4. Structure of a $\boldsymbol{GF(2^{p^q})}$ HCA.


generated patterns
GF(2^2) elements
{0, 1, 2, 3}
T = [0 2 0; 3 0 3; 0 2 3]3x3
seed S = { 2 3 0 }
GF(2^2)
GF(2)
Vector representations of the elements 3=[1 1], 2=[1 0], 1=[0 1], 0=[0 0]

GF(2^2)	GF(2)
2 $\boxed{3}$ 0	1 0 $\boxed{1}$ 1 0 0
1 2 3	0 1 1 0 1 1
3 3 3	1 1 1 1 1 1
0 2 2	0 0 1 0 1 0
2 2 1	1 0 1 0 0 1
0 3 0	0 0 1 1 0 0
$\boxed{3}$ 1 3	$\boxed{1}$ 1 0 1 1 1
2 1 0	1 0 0 1 0 0
3 1 1	1 1 0 1 0 1
2 0 3	1 0 0 0 1 1
2 1 1	1 0 0 1 0 1
3 2 3	1 1 1 0 1 1
....	....

Fig. 5. The patterns generates by a three-cell $\boldsymbol{GF(2^2)}$ CA.

1) Selection of extension field parameters (p and q) and n for design of n-cell $GF(2^{p^q})$ HCA-based TPG.
2) Designing the characteristic matrix T that specifies the circuit structure of HCATPG. It involves: 1) formation of the dependency matrix (D) of the HCA that identifies the dependencies of one HCATPG cell on its neighbors and 2) specification of the weight values ($w_i s$) of the dependencies.

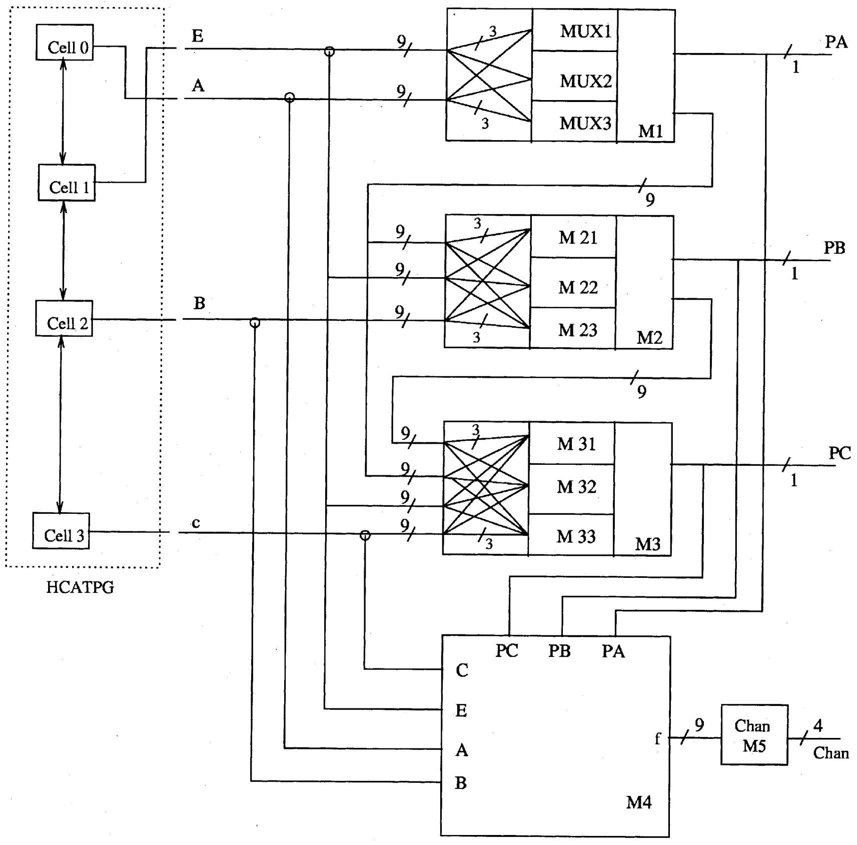
Definition 3: The dependency matrix $[D]_{n\times n}$ of an n-cell hierarchical *CA* identifies the dependencies of one cell on its neighbors, where

$$D_{ij} = \begin{cases} 1, & \text{if the next state of } i\text{th cell depends on the} \\ & \text{present state of } j\text{th cell by some weight} \\ & w_{ij} \in GF(2^{p^{q^{r\cdots}}}),\ i,\ j = 1, 2, \ldots, n \\ 0, & \text{otherwise.} \end{cases}$$

An example four-cell dependency matrix is noted below where the first cell, as per the first row, has dependency on the second cell and itself. The second cell is dependent on the first and third, and so on

$$D = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \text{ where } D[i,j] \in GF(2)\ \forall i, j.$$

The characteristic matrix T of $GF(2^{p^q})$ HCATPG is derived from D by specifying the weights of the dependencies, that is, by replacing the $1s$ of D with the nonzero weight values.



$$T=\begin{bmatrix}\beta^2 & \beta & 0 & 0\\ \beta^2 & 0 & \beta^{32} & 0\\ 0 & \beta & \beta^{32} & \beta\\ 0 & 0 & \beta^{32} & \beta\end{bmatrix}\qquad \beta=\begin{bmatrix}\alpha^2 & \alpha^7 & 0\\ \alpha^7 & \alpha^6 & \alpha^7\\ 0 & \alpha^7 & \alpha^6\end{bmatrix}\qquad \alpha=\begin{bmatrix}1 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$$

Fig. 6. High-level model for modified c432 bench circuit.

B. Selection of Field Parameters and n

This step executes partitioning of primary inputs to form *clusters* and subclusters input to different circuit blocks. The cardinalities of the clusters and subclusters are identified. The most frequent cardinality of the clusters $(c_1, c_2, \ldots, c_k)$ is chosen as the value of $(p \times q)$, whereas the most frequent cardinality of subclusters within a *cluster* is taken as the value of parameter p. The algorithm to select the extension field parameters and n for two level of hierarchy is as follows.

```
Algorithm 1: Select_parameter_values
Input: Hierarchical structural description
 of the CUT.
Output: Value of $p$, $q$ and $n$.
Step 1. Identify PI clusters $(c_1, c_2, \ldots, c_k)$
 and their cardinalities.
Step 2. Fix $p \times q =$ the most frequent cardi-
 nality of PI clusters.
Step 3. Identify subclusters and their
 cardinalities for each $c_i$.
Step 4. The value of $p$ for a cluster $c_i$ =
 the most frequent cardinality of the sub-
 clusters within $c_i$. Obtain the value of $q$
 from Step 2.
Step 5. Fix the value of $n$ as
```

$$n=\left\lceil\left(\frac{|c_1|}{pq}\right)\right\rceil+\left\lceil\left(\frac{|c_2|}{pq}\right)\right\rceil+\cdots+\left\lceil\left(\frac{|c_k|}{pq}\right)\right\rceil+\left\lceil\left(\frac{|rest\ PIs|}{pq}\right)\right\rceil.$$

```
Step 6. Return $p$, $q$, and $n$.
```

This process can be repeated for the design of HCATPG with more than two levels of hierarchy. The number of hierarchical levels will be same as that of the structural hierarchy of the CUT. The HCATPG for testing the circuit of Fig. 6 is a four-cell $GF(2^{3^3})$ HCA with two levels of hierarchy. The cells are marked as 0, 1, 2, and 3.

Once the clusters of PIs get identified, the next step is to investigate their interrelationship, that is, the formation of the dependency matrix for the HCATPG.

C. Identification of Dependency Matrix

The existence of the dependencies among the PI clusters are exploited to design the dependency matrix. The algorithmic steps to generate the dependency matrix D are given as follows.

```
Algorithm 2: Form_D_matrix
Input: Hierarchical structural description
 of the CUT, PI clusters and $n$.
Output: The dependency matrix $D$ of the
 HCATPG.
Step 1. Identify PI clusters that are
 input to same circuit module.
Step 2. The element $D[i,j] = 1$ if the $i$th
 and $j$th PI clusters are fed to the same
 RTL/functional block, otherwise $D(i,j) = 0$.
Step 3. Locally modify $D$ to make it
 tridiagonal and $\det[D] = 1$.
Step 4. Return $D$.
```

For the circuit of Fig. 6, *Algorithm 2* results in

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

where PI clusters A, E, B, and C are, respectively, fed from the *cells* 0, 1, 2, and 3. To get the dependency matrix in the three-neighborhood (*Step 3 of Algorithm 2*) we employ the graph algorithm noted in [18]. The D matrix satisfying the relation $\det[D] = 1$ generates efficient HCATPG. While the elements of D are in $GF(2)$, the elements of the T matrix are in $GF(2^{p^q})$ as discussed below.

D. Design of Characteristic Matrix T

For the proposed design, we employ the HCA having cyclic structure (group HCA). The following theorems guide the efficient design of group HCA (T matrix).

Theorem 2: A HCA having identical nonzero weight in each row (or 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 $GF(2^{p^q})$ HCA with dependency matrix D. Assume its T matrix is such that all the nonzero elements in each column are identical and in *column* i it is w_i ($w_i \in GF(2^{p^q})$). If $v_1, v_2, \ldots, v_n$ are the column vectors of T, then

$$\begin{aligned} \det[T] &= \det[w_1 v_1, w_2 v_2, \ldots, w_n v_n] \\ &= w_1 w_2 \ldots w_n \ \det[v_1, v_2, \ldots, v_n] \\ &= w_{n+1} \ \det[D], \text{ where} \\ w_{n+1} &= w_1 w_2 \ldots w_n \in GF\left(2^{p^q}\right) \text{ and } w_{n+1} \neq 0. \end{aligned}$$

Therefore, iff $\det[D] \neq 0$, the value of $\det(T) \neq 0$. Hence, as per *Theorem 1*, the HCA is a group CA. ■

For the HCA of Fig. 7 having equal nonzero weights β^2, β^4 *and* β, respectively, in the *column*s 1, 2, and 3 of its T matrix, $\det[D] = 1$ (D being a binary matrix). It is a group CA. In general, the cycle structure of a group CA is defined as $[\mu_1(k_1), \mu_2(k_2), \ldots]$, that is, it has μ_i number of cycles of length k_i, $\forall i = 1, 2, \ldots$. The cycle structure of the CA of Fig. 7 is [1(1), 1(262 143)]; it has one cycle of length 1 and one of length 262 143.

Property 1: A HCA designed with primitive weight value (*Definition 1*) while satisfying the result of *Theorem 2* generates larger length cycle of the order of 2^{npq-1}.

Experimental validation of *Property 1* is reported in Table III. The first column shows the number of cells (n) and the extension field parameter (p) of different $GF(2^p)$ CA. The HCA for a particular n and p is designed for a large number of trials with primitive and nonprimitive weight sets. In the columns 2 and 3, the cycle lengths produced by the HCA in at least 25% cases are noted. The HCAs are designed with different values of n and p.

Theorem 3: If an n-cell *HCA* with characteristic matrix T has cycle structure $[1(1), \mu_1(m)]$, where $\mu_1 m = (2^{npq} - 1)$ and p, q are the extension field parameters, then the HCA designed with $[T]^\mu$ generates cycles $[1(1), \mu_1\mu(k)]$, where $\mu k = m$.

Proof: From the cycle structure of T it is obvious that for any nonzero 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 a cycle of length k_1, where k_1 is a factor of k and $k_1 k_2 = k$.

Then for some state x_1, $[T^\mu]^{k_1} x_1 = x_1$, that is

$$[T^\mu]^{k/k_2} x_1 = \left[T^{\mu k}\right]^{1/k_2} x_1 = [T^m]^{1/k_2} x_1 = x_1.$$

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 $\beta^2, \beta^4, \beta^8, \ldots$ are also primitive.

Proof: Let us assume that β is a primitive element in $GF(2^{p^q})$. β is primitive, that is, the $\beta, \beta^2, \beta^3, \ldots, \beta^{(2^{pq}-1)}$ all are unique and the cycle structure of β is $[1(1), 1(m)]$, where $m = (2^{pq} - 1)$. From *Theorem 3* it follows that β^μ has the cycle structure $[1(1), \mu(k)]$, where $\mu k = m$, as $\mu_1 = 1$. Since, m is odd, then 2^i will be prime to m for $i = 1, 2, 3, \ldots, (pq - 1)$. Hence, β^{2^i} has the cycle structure as $[1(1), 1(m)]$, that is, β^{2^i} is primitive. ■

The results of the theorems and lemma guide the design of HCATPG generating its T matrix. The nonzero values "1" in a column of the D matrix (derived out of *Algorithm 2*) are replaced with a primitive weight value (β^j) from the set $W = \{\beta, \beta^2, \beta^4, \ldots\}$ to design the T matrix of the HCA. In a $GF(2^{p^q})$ HCA, the weight $\beta^j \in GF(2^{p^q})$. The β^j_{matrix} is a $q \times q$ matrix having its elements from the set $\{0, \alpha, \alpha^2, \ldots, \alpha^{(2^p-1)}\}$.

Limited Iteration to Identify the Structure of T: The selection of primitive weight for a column in T is performed by tuning the HCATPG for the given CUT in two steps—global and local. While in global tuning we fix up the interconnection among HCA cells, the local tuning defines the interconnection among the subcells within a cell.

a) Global tuning: This tuning involves fixing of the weight value (β^j) for a TPG cell. For example, to design the T matrix of the customized HCATPG for the CUT of Fig. 6, the $1s$ of the dependency matrix D are replaced by the weights from the set $\{\beta, \beta^2, \beta^4, \ldots\}$. The T noted in Fig. 6 results in maximum fault coverage for the CUT and the weight values $\beta^2, \beta, \beta^{32}$, *and* β are fixed for the TPG cells 0, 1, 2, and 3, respectively. This design step is referred to as the *global* tuning, while the weight values for the HCATPG cell interconnection are assigned to generate the corresponding T matrix of the HCA leading to maximum fault coverage.

b) Local tuning: The local tuning (also referred to as fine tuning) of the customized HCATPG is done by structural modifications of the weights fixed through global tuning. The final structure of weight β^j assigned for a cell is fine tuned by changing the elements $\alpha^i s$ in the $q \times q$ matrix of β^j. The structural modification of a weight β^j defines the interconnection of subcells within a HCATPG cell. Local tuning is implemented for the circuit blocks with low fault coverage identified by the HCATPG derived through global tuning.

$$\beta = \begin{bmatrix} \alpha & \alpha^3 & 0 \\ \alpha^3 & \alpha^2 & \alpha^3 \\ 0 & \alpha^3 & \alpha^2 \end{bmatrix}_{3 \times 3} \qquad \alpha = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = 2 \qquad D = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & \beta^4 & 0 \\ \beta^2 & 0 & \beta \\ 0 & \beta^4 & \beta \end{bmatrix}_{3 \times 3} = \begin{bmatrix} \begin{bmatrix} 0&0&0\\0&0&0\\0&0&0 \end{bmatrix} & \begin{bmatrix} \beta^4 \end{bmatrix} & \begin{bmatrix} 0&0&0\\0&0&0\\0&0&0 \end{bmatrix} \\ \begin{bmatrix} \beta^2 \end{bmatrix} & \begin{bmatrix} 0&0&0\\0&0&0\\0&0&0 \end{bmatrix} & \begin{bmatrix} \alpha & \alpha^3 & 0 \\ \alpha^3 & \alpha^2 & \alpha^3 \\ 0 & \alpha^3 & \alpha^2 \end{bmatrix} \\ \begin{bmatrix} 0&0&0\\0&0&0\\0&0&0 \end{bmatrix} & \begin{bmatrix} \beta^4 \end{bmatrix} & \begin{bmatrix} \beta \end{bmatrix} \end{bmatrix}_{9 \times 9}$$

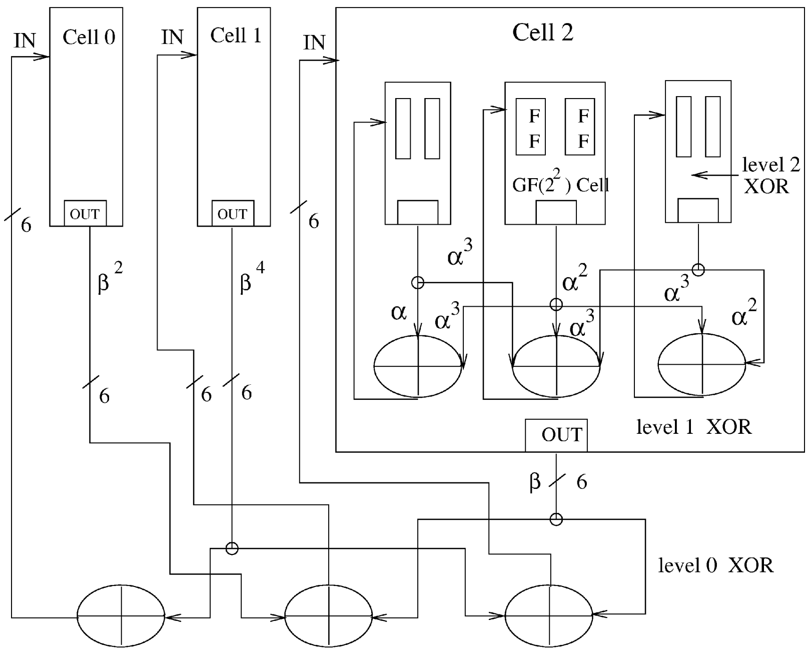


Fig. 7. Structure of a three-cell $GF(2^{2^3})$ CA.

The algorithmic steps to tune the T matrix of the *HCATPG* are reported.

```
Algorithm 3: Tune_HCATPG
Input: (1) Hierarchical structural description of the CUT; (2) Dependency matrix $D$; (3) Parameter values $p$, $q$ and $n$;
    (4) Weight set $W = \{\beta, \beta^2, \beta^4, \ldots\}$, where $\beta$ is the root of the generator polynomial
    in $GF(2^{p^q})$; (5) Seed file (SF); (6) Upper limit of test vectors ($L$); (7) Maximum
    iteration cycles for global and local tuning, $I_{\rm global}$ and $I_{\rm local}$; (8) Desired fault
    coverage (FC) for the CUT and number of undetected faults udfp in a module as a
    percentage of total number of faults in the CUT.
Output: (1) Fault coverage, (2) $T$ matrix of the HCATPG, (3) selected seed and (4)
    number of test vectors.
Step 1. Select a seed ($S$) from the SF.
Step 2. To construct $T$ matrix of the HCA
    $\forall j = 1$ to $n$ do
     {replace all 1s of column $j$ in $D$ by a
    weight $w_j \in \{W\}$ }.
Step 3. Find the cycle lengths of the HCA.
    If the minimal length cycle $< L$, go to 2.
Step 4. Fault simulation—generate test
    patterns in steps with the seed until
    $L$ is reached or fault coverage remains
    constant in two successive steps (a step
 involves generation of $L'$ number of test
 patterns defined by the test designer).
Step 5. Note the fault coverage $fc$. If
 $fc \geq$ FC, go to Step 10.
Step 6. Repeat Steps 4 and 5 with the next
 seed of SF. Select the seed $S_{\max}$ with
 maximum fault coverage.
Step 7. Perform Step 2–5 with the selected
 seed $S_{\max}$ until number of iterations ex-
 ceeds $I_{\text{global}}$.
Step 8. Find $T$ matrix with maximum fault
 coverage ($fc$) and the circuit modules
 $\{M_1, M_2, \ldots, M_k\}$ with undetected faults in
 $M_i$ ($i = 1, 2, \ldots, k$) is greater than udfp.
Step 9. Perform Local Tuning-
 (a) Identify the HCATPG cell ($j$) which
 carries test patterns to the circuit
 module $M_i$, $\forall i = 1$ to $k$, identified in
 Step 8.
 (b) Modify structure of weight $w_j$ of the
 $j$th column of $T$, $\forall j = 1$ to $k$. Repeat (b)
 until minimal length cycle of modified
 $T \geq L$.
(c) Execute fault simulation (Step 4) with
 the modified $T$ and seed $S_{\max}$. Repeat (b)
 and (c) until fault coverage is reached
 to FC or number of iterations $I_{\text{local}}$.
Step 10. Return maximum fault coverage
 achieved, corresponding $T$ matrix, the
 seed and the number of test vectors. Stop
```

Each of the algorithms reported has been coded in *C* language and a complete package named BECKIT has been developed. It is found that to complete the design of HCATPG for a CUT, more than 97% processing time of BECKIT is consumed for fault simulation. Hence, additional computation cost of the HCATPG design will have a marginal effect on total run time.

E. Experimental Results

The experimental setup for evaluation of fault coverage of HCATPG is noted as follows. 1) the PI clusters are identified from the hierarchical structural description provided in [ITC99] for benchmark circuits. 2) In order to increase the number of experimental circuits, more than one bench circuits are combined to get a hierarchical structural net list. Table IV shows the circuit characteristics. 3) Cadence fault simulator *verifault* is employed for fault simulation. 4) The value of input parameters chosen for the experimentation are:

a) upper limit of test vectors $L = 20\,000$;
b) maximum number of iterations for the global tuning $I_{\text{global}} = 5$ and local tuning $I_{\text{local}} = 10$;
c) desired FC is specified as 100% and *udfp* for a module as .05%;
d) the SF is assumed to have two seeds only.

Table V shows the fault coverage figures in *Column 3* with number of test vectors noted in Column 2 for the circuits specified in Table IV. The performance of HCATPG evaluated through BECKIT is compared with GLFSR (*Column 4*) [8] and maximal length $GF(2)$ CA-based test structures (*Column 5*). We have designed the maximal length GLFSR-based TPGs for $\delta = 2, 3, 4$, and 5 as proposed in [8]. A CUT is tested for five seeds (taken randomly) with each of four TPGs. The maximum fault coverage achieved out of these 20 runs for a CUT is noted. In order to have a fair comparison, the number of iterations with different seeds have been fixed at 20 and a benchmark circuit is tested with the same number of test vectors for all designs. The fault coverage figures are expressed as

$$\text{fault coverage} = \frac{\text{Total no. of detected faults}}{\text{Total no. of faults in the CUT}}.$$

For sequential circuits, the FFs are assumed to be initialized to 0 through hardware reset.

The figures of Table V confirm that the HCA-based design achieves better fault coverage. In absolute terms, the improvement of fault coverage figures of HCATPG over that of GLFSR and $GF(2)$ in the range

TABLE III
CYCLE STRUCTURE: PRIMITIVE VERSUS NONPRIMITIVE WEIGHTS

HCA	cyl struc with probability of occur 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

TABLE IV
CIRCUIT DESCRIPTIONS

Ckt.	#	#	#	Val. of		Designed
name	PI	PO	Faults	p	q	from
c1	36	7	490	3	3	c432
c2	32	32	7744	8	2	c6288
c3	41	32	758	2	4	c499
c4	33	25	970	2	2	c1908
c5	67	48	1216	1	8	c499+c74181
c6	82	64	1516	4	2	c499+c499
c7	72	14	980	3	3	c432+c432
s1	24	4	558	4	2	c17+s27+s208
s2	23	24	714	4	2	c17+s344+s349
s3	114	64	2531	4	2	c6+two s1
s4	102	60	2178	4	2	c6+two s2
s5	66	53	1741	2	2	c4+s1+s2
s6	36	25	1057	3	3	c432+four s298

c represents combinational, s represents sequential

TABLE V
TEST RESULTS OF CUSTOMIZED DESIGN

Ckt name	Test Vec.	Fault coverage (%)			PIFC of HCATPG over	
		HCATPG	GLFSR	GF(2)	GLFSR	GF(2)
c1	400	99.18	99.18	98.57	0	42.66
c2	80	99.56	99.56	99.48	0	15.38
c3	450	98.95	98.54	98.29	28.08	38.60
c4	3000	99.12	99.07	98.76	5.38	29.03
c5	600	99.34	98.03	99.34	66.49	0
c6	700	98.95	98.75	98.81	16	11.76
c7	400	99.08	98.98	98.47	9.80	39.87
s1	2500	99.28	98.57	98.75	49.65	42.40
s2	1400	97.76	97.06	97.20	23.81	20
s3	3500	95.46	91.70	90.75	45.30	50.92
s4	4000	97.25	95.32	94.63	41.24	48.79
s5	5000	99.20	96.09	96.27	79.54	78.55
s6	4500	91.67	87.70	86.28	32.28	39.30

of 95%–99% is not significant. However, the following points are worth highlighting at this stage.

a) Once a fault coverage figure reaches the range 95%–99%, for real life circuits any improvement will be incremental only.
b) The deterministic test pattern generators (ATPGs) incur the majority of its computing overhead while identifying the test vectors to improve fault coverage in the range of 95%–99%. In many situations ATPGs abandon the exercise due to extremely high computing load.

Percentile Improvement in Fault Coverage : In order to project a realistic assessment, we propose the following measurement:

Percentile Improvement in Fault Coverage PIFC

$$= \frac{X - Y}{100 - Y} \times 100$$

where X = fault coverage with the new design (say with HCATPG) and Y = fault coverage with the existing scheme (say $GF(2)$ CA/GLFSR). The PIFC with HCATPG over *GLFSR* and $GF(2)$ CA are noted in the last two columns of Table V for comparison. The results of Table V clearly establish the inherent strength of HCATPG to push up the fault coverage figures in the region of 97%–99%, achieved by the contemporary schemes.

F. Area Overhead of HCATPG

The process of hardware implementation for $GF(2)$ CA is reported in [14]. An n-cell $GF(2)$ CA (T matrix) is realized with n number of $GF(2)$ cells (*FF*s). The interconnection among the cells follows from each of the rows of T with its elements as 0 or 1. For a particular i, the outputs from cell j ($j = 1$ to n), where $T[i, j] \neq 0$, are XORed and then connected to the input of cell i. Similarly, an n-cell $GF(2^{p^q})$ hierarchical *CA* with characteristic matrix $[T]_{n\times n}$ is realized from the corresponding $npq \times npq$ binary matrix T_{bin}. To find out T_{bin}, the β^is (β, β^2 *and* β^4 in Fig. 7) of $[T]_{n\times n}$ are replaced by their corresponding $q\times q$ (3 × 3) matrix representations to get the $[T]_{nq\times nq}$ matrix (9 × 9 of Fig. 7). The elements of $[T]_{nq\times nq}$ belong to $GF(2^p)$ and in turn can be replaced by the equivalent binary $p \times p$ (2 × 2) matrix for each element (α, α^2 *and* α^3 in Fig. 7). This results in an $nqp \times nqp$ (18 × 18) T_{bin}.

The HCA structure noted in Fig. 7 is designed from its T matrix employing multilevel XOR gates to reduce hardware overhead. The $[T]_{3\times 3}$ describes the interconnection among the cells 0–2; whereas the weights β, β^2 and β^4 (belong to $GF(2^{2^3})$) define the interconnection of the subcells within each of the cells 0–2. Similarly, the α^is $\in GF(2^2)$ guide the connection among the *FF*s within a subcell. For example, the weights α, α^2 and α^3 define the connectivity of *FF*s within the subcells of cell 2.

To realize the HCA of Fig. 7, the α, α^2 and α^3 of different subcells in cell 2 are realized with level 0 XOR gates. Similarly, for the cells 0 and 1 the α^is are also realized. In level 1 of XOR realization, the weights β^2 from cell 0, β^4 from cell 1, and β from cell 2 are implemented. Once we realize the β^is, the connectivity among the cells (following the $T_{3\times 3}$ matrix) is established using level 2 of XOR gates. In the process, the number of two-input XOR gates to realize the HCA of Fig. 7 gets reduced to 38 while the number of two input XOR gates required to realize the HCA from its T_{bin} is 71. For further reduction in area overhead of HCATPG, the concept of TPG folding we have reported in [15] has been implemented.

V. CONCLUSION

This paper presents an innovative concept of HCA. The theory of the extension field is utilized in designing the HCA. We establish HCA as an efficient TPG structure for testing VLSI circuits. A scheme to customize the HCATPG structure for a CUT is noted to achieve maximal fault efficiency.

REFERENCES

[1] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudo-Random Techniques*. New York: Wiley.
[2] P. D. Hortensius *et al.*, "Cellular automata based pseudo-random number generators for built-in self-test," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 842–859, Aug. 1989.
[3] C. L. Chen, "Linear dependencies in linear feedback shift registers," *IEEE Trans. Comput.*, vol. C-35, pp. 1086–1088, Dec. 1986.
[4] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proc. IEEE Eur. Test Conf.*, 1991, pp. 237–242.
[5] T. Gheewala, H. Sucar, and P. Varma, "A global bist methodology," in *Proc. Asian Test Symp.*, 1993, pp. 154–159.
[6] A. P. Stroele, "Arithmetic pattern generators for built-in self test," in *Proc. ICAD Austin*, 1996, pp. 131–134.
[7] S. Gupta, J. Rajski, and J. Tyszer, "Arithmetic additive generators for pseudo-exhaustive test patterns," *IEEE Trans. Comput.*, pp. 939–949, 1996.
[8] D. K. Pradhan and M. Chatterjee, "GLFSR-a new test pattern generator for built-in-self-test," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 319–328, Feb. 1999.
[9] J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated synthesis of large phase shifters for built-in self-test," in *Proc. Int. Test Conf.*, 1998, pp. 1047–1056.
[10] S. S. K. Chiu and C. Papachristou, "A built-in self-testing approach for minimizing hardware overhead," in *Proc. Int. Test Conf.*, 1991, pp. 282–285.
[11] I. Parulkar, S. Gupta, and M. Breuer, "Data path allocation for synthesizing *rtl* designs with low *bist* area overhead," in *Proc. Design Automation Conf.*, 1996, pp. 395–401.
[12] A. Orailoglu and I. G. Harris, "Microarchitectural synthesis for rapid *bist* testing," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 573–586, 1997.
[13] I. Ghosh, N. K. Jha, and S. Bhaumik, "A *bist* scheme for *rtl* circuits based on symbolic testability analysis," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 111–128, 2000.

[14] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chatterjee, *Additive Cellular Automata—Theory and Applications*. New York: IEEE Computer Soc. Press, 1997, vol. 1.

[15] B. K. Sikdar, D. K. Das, V. Boppana, C. Yang, S. Mukherjee, and P. P. Chaudhuri, "$\boldsymbol{Gf(2^p)}$ cellular automata as a built in self test structure," in *Proc. ASP-DAC 2001*, Japan.

[16] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[17] 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. 14th Int. Conf. VLSI Design*, India, Jan. 2001.

[18] B. K. Sikdar, "*Theory and Application of Hierarchical Cellular Automata for VLSI Circuit Testing*,", B. E. College (Deemed University), Howrah, India, 2002.