Design of An On-Chip Test Pattern Generator Without Prohibited Pattern Set (PPS)

Niloy Ganguly[1] Biplab K Sikdar[2] P P alChaudhuri[2]

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

Abstract— **This paper reports the design of a Test Pattern Generator (TPG) for $VLSI$ circuits. The on-chip TPG is so designed that it generates test patterns while avoiding generation of a given Prohibited Pattern Set (PPS). The design ensures desired pseudo-random quality of the test patterns generated. The experimental results confirm high quality of randomness while ensuring fault coverage close to the figures achieved with a typical Pseudo Random Pattern Generator ($PRPG$) designed around maximal length $LFSR/CA$. Compared to the conventional $PRPG$ it incurs no additional cost.**

I. Introduction

This paper addresses a real life problem usually encountered by the test engineers of a semiconductor company. To the best of our knowledge no published literature exists for an elegant solution of the problem reported in this paper.

The pseudo-random pattern generators ($PRPG$s) are widely used in $VLSI$ circuits [6]. The PRPG generates a large volume of patterns to test different CUTs (*Circuit Under Test*) of a $VLSI$ chip that may be accessed through a full or partial scan path. However, there are situations where some patterns are declared prohibited to a CUT. If the CUT is subjected to such a pattern of the prohibited pattern set (PPS), it may be placed to an undesirable state and even may get damaged. The manufacturers do face the problem while testing the chip equipped with on-chip TPG.

In the above context, this paper proposes the design of a test pattern generator (TPG) to generate the test patterns without the PPS specified for the CUT. Moreover, the design ensures the desired randomness qualities of the generated patterns and maintains the fault efficiency in a CUT.

The theoretical framework of Cellular Automata (CA) noted in [1] has provided the foundation of this work. The class of CA referred to as *non-maximal length group CA* are used for the design of the TPG. Compared to the conventional $PRPG$, built around maximal length $CA/LFSR$ (Linear Feedback Shift Register), the TPG proposed in this paper does not incur any additional cost.

The proposed methodology can be also implemented with $LFSR$ based TPG. However, the modular and cascadable local neighborhood structure of cellular automata suits ideally for VLSI applications.

Layout of the paper is as follows. A brief introduction to CA preliminaries (*Section II*) precedes the proposed solution methodology reported in *Section III*. The experimental results are subsequently reported in *Section IV* which clearly establish the proposed design of TPG as the most desirable solution for the real life problem addressed in this paper.

II. Cellular Automata Preliminaries

The *cellular automata (CA)* consists of a number of cells arranged in a regular manner, where the state transition of a cell depends on the present states of its neighbor. A CA cell contains memory element (FF) and can store a value from the set {0,1} $\in GF(2)$ (Galois Field (2)) - such a CA is referred to as $GF(2)$ CA. If the next state of a cell is assumed to depend only on itself and its two neighbors (left and right), then this leads to 3-neighborhood dependency. In a 3-neighborhood CA, the state of the i^{th} cell at time $(t+1)$ is denoted as

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

where q_{i-1}^t, q_i^t and q_{i+1}^t are the states of the $(i-1)^{th}$, i^{th} and $(i+1)^{th}$ cells respectively at time t; f is the next state function of the CA. The details on cellular automata and its applications are reported in [1] [5].

Characterization of cellular automata : An n-cell $GF(2)$ linear CA can be characterized by an $n \times n$ characteristic matrix T, where

$$T_{ij} = \begin{cases} 1, & \text{if the next state of the } i^{th} \text{ cell depends} \\ & \text{on the present state of the } j^{th} \text{ cell} \\ & \text{i, j = 1, 2, ..., n} \\ 0, & \text{otherwise} \end{cases}$$

If we restrict to the 3-neighborhood dependence, then T becomes a tridiagonal matrix with elements from GF(2). The n-degree polynomial of which T is a root is called the *characteristic polynomial* of the CA.

If all the states lie on some cycles, the CA is referred to as *group CA*. The TPG proposed in this paper employs *group CA*.

Properties of Group CA: All the states of a CA lie on some cycles iff its T matrix is nonsingular - that is, for a group CA the $det[T] \neq 0$. The group CA can be classified as maximal-length and non-maximal length CA. The maximal length CA *(Fig.1)* is the special

Proceedings of the 15th International Conference on VLSI Design (VLSID'02)
0-7695-1441-3/02 $17.00 © 2002 IEEE

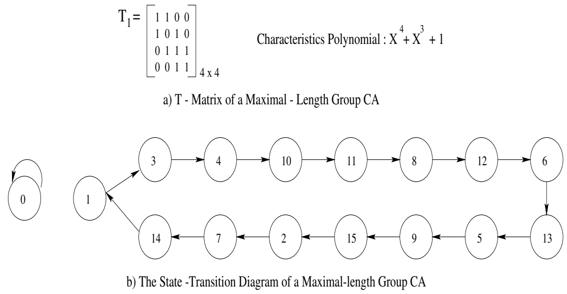

Fig. 1. A 4-cell maximal length group CA

class of group CA having a cycle of length $2^n - 1$ with all non-zero states, where n is the number of cells in the CA. Maximal length CA generates excellent pseudo-random sequence [6].

The characteristic polynomial of an n-cell maximal length CA is the n^{th}-degree primitive polynomial. For a non-maximal length CA, the characteristic polynomial $f(x)$ gets factored to invariant polynomials (elementary divisors), $f(x) = f_1(x)f_2(x)\cdots f_n(x)$. Each of the elementary divisors $f_i(x)$ forms cyclic subspace - which leads to the generation of multiple cycles. The entire state space V of a non-maximal length group CA is the direct sum $V = I_1 + I_2 + \cdots + I_n$, where I_i is the cyclic subspace generated by $f_i(x)$.

The 7-cell CA of *Fig.2* is a non-maximal length group CA with its states on 4 cycles. The primary cyclic subspaces (cycle length 7 and 15) of the state space V are generated by the elementary factors x^3+x+1 and x^4+x+1 of the characteristic polynomial $x^7+x^5+x^3+x^2+1$. The secondary cycle of length 105 is generated through combination of primary cycles.

III. Design of The TPG

This section introduces the design of the TPG which can only generate the patterns safe for the circuit - that is, free from the prohibited pattern set (PPS).

Overview of the proposed design: The patterns generated from a maximal length CA display better "randomness" quality and hence used as Pseudo Random Pattern Generator ($PRPG$) for testing $VLSI$ circuits. However, the n-cell ($n \geq 16$) non-maximal length group CA having a sufficiently large cycle length doesn't suffer in respect of randomness qualities and also can be used as the TPG for a CUT. This observation has been validated through exhaustive experimental results reported in *Section IV*.

In the proposed design, the *non-maximal* length group CA is considered as the TPG, where the CA state space is divided into multiple cycles; at least one of the cycles has large cycle length. The patterns, prohibited to the CUT, are made to fall in the smaller length cycles while one of the bigger cycles is utilized

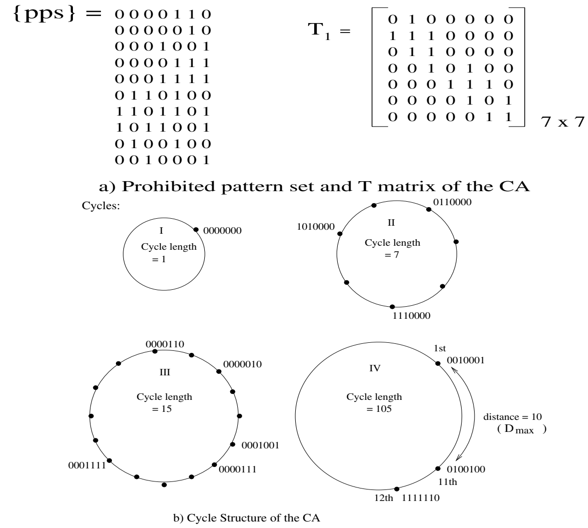

Fig. 2. $PRPG$ without the prohibited patterns

to generate the test patterns. The above discussions is next illustrated with an example.

Let us consider the design of a TPG for a circuit with 7-primary inputs (PIs). The PPS of the CUT contains 10 prohibited patterns as shown in *Fig.2(a)*. The 7-cell group CA represented by $[T_1]_{7\times 7}$ matrix *(Fig.2(a))* can be selected as the candidate TPG. The states generated by the CA are divided into three primary cycles of length 1, 7 & 15 and only one large secondary cycle of length 105 *(Fig.2(b))*. Out of the given PPS, the cycle *II* ($length$ = 7) contains 3-prohibited patterns PPS_1={0110100, 1101101, 1011001}, whereas the prohibited patterns PPS_2 = {0000110, 0000010, 0001001, 0000111, 0001111} fall in cycle *III* ($length$ = 15). That is, out of 10, the 8 prohibited patterns (PPS_1 & PPS_2) are covered by the cycles *II* and *III*. Therefore, the patterns of the larger length cycle ($Cycle\ IV$) can be used for testing the CUT. Cycle IV is free from 80% of PPS.

The rest 2 prohibited patterns {0010001, 0100100}, covered by the cycle IV, are separated by a distance of 10 time steps - that is, $T^{10}(0010001) = (0100100)$. To avoid these two prohibited patterns, the CA is loaded with 0100100 and can run for L=94 time steps to generate test pattern sequence starting from 1111110 *(Fig.2(b))*. In effect, the group CA (T_1) with 0100100 as the seed is a desired TPG.

We introduce the following terminologies to designate the cycles of a non-maximal length group CA.

TargetCycle (TC): The cycle of largest length generated by the CA.

Redundant Cycle (RC): The cycles other than TC - these are redundant in the sense that these are not used for generation of pseudo random test patterns.

The design of the TPG for an $n - PI\ CUT$ should satisfy the following constraints:

C_1**:** The TPG is synthesized out of an n-cell non-maximal length group CA having a number of cycles. One of the cycles referred to as Target Cycle (TC) can be used for generation of pseudo-random test patterns.

C_2**:** Most of the patterns of PPS lie in the cycles referred to as redundant cycles (RCs).

C_3**:** The remaining members of PPS, if there are any, should get clustered in the TC within a smaller distance D_{max} so that most of the patterns of TC can be employed for testing the CUT in a single run.

A. Group CA Satisfying the Constraints

It has been established in [7] that the generation of CA based TPG satisfying all the constraints C_1, C_2, and C_3 is a hard problem. Further, the resulting CA should have three neighborhood since local neighborhood interconnects is desirable for on-chip implementation of the TPG. This leads to unsolvability of the problem. So we proceed to develop an efficient heuristic that generates acceptable solution for majority of the problem instances.

A.1 Design Satisfying Constraint C_1

An elegant method to synthesize a group CA in O(n) time for a given cycle structure has been developed. The synthesis algorithm accepts the cycle lengths as an input and generates the T matrices & its cyclic components, as the output.

For the current problem, the n-cell CA based TPG for a given CUT with n-PI (Primary Input), should have a TC with length greater than or equal to:

(i) 3(2^n-1)/4 for $n \leq 16$, and (ii) (2^n-1)/2 for $n \geq 16$, to ensure the desired pseudo random quality of the patterns generated by the TC. The outline of the synthesis algorithm is noted below for the simple case where RCs have three cycles - one of length 1 with all 0's state.

Input: (i) n, (ii) the length of TC (Target Cycle)
Output: (i) T matrix of the non-maximal length group CA, (ii) the resulting cycle structure
Step 1: Generate the numbers a & b such that

- a and b are mutually prime
- $a + b = n$
- $(2^a - 1) \cdot (2^b - 1)$ is close to TC

Step 2: Generate T matrices T_a and T_b corresponding to maximal length CA of size a and b respectively

Step 3: Place T_a and T_b in block diagonal form [2] to derive $T_{n\times n}$ corresponding to the desired CA

For $n = 7$ the TC should have length of 96 ($\approx 3/4 \times$ 127); a and b are assumed to be 3 and 4. The algorithm synthesizes the CA having cycle structure 1(1), 1(7), 1(15), 1(105) (that is, one cycle of length 1, 7, 15, and 105) as shown in *Fig.2*. Its Cycle IV of length 105 satisfies the constraint C_1.

The synthesis algorithm generates a set S_{CA} of CA satisfying the constraint C_1. Next we identify a subset $S'_{CA} \subseteq S_{CA}$ that satisfy the constraints C_2 and C_3.

A.2 Subset Satisfying Constraints C_2 and C_3

The proposed methodology for generation of $S'_{CA} \subseteq S_{CA}$ aims to ensure that the Redundant Cycles (RCs) of the CA that satisfies the constraint C_1 cover maximum number of prohibited patterns $\in PPS$. The necessary and sufficient conditions to be satisfied to achieve this goal are next discussed after introducing a few commonly used terminologies:

Rank: The *rank* of a set is defined as the number of independent vectors in the set. For example, the rank of PPS in *Fig.2* is k=7.

Basis: A set S={ $u_1, u_2, \cdot\cdot, u_n$} of vectors is a basis of a pattern set PS, if every vector $v \in PS$ can be uniquely written as the linear combination of $u_i \in S$, $\forall i = 1, 2, \cdots, n$. The number of basis vectors in S is the *rank* of the set PS.

Vector Space: A vector space V over a field K

- is a commutative group under addition
- for any scalar $k_1,\ k_2 \in K$ and any vector $u, v \in V$,
  $k_1.(u + v) = k_1.u + k_1.v$,
  $(k_1 + k_2).u = k_1.u + k_2.u$ and
  $(k_1.k_2).u = k_1(k_2.u)$
- unit scalar 1 $\in$ K, where 1.u = u

Subspace: Let W be a subset of a vector space V over a field K. W is called a subspace of V if W is itself a vector space over K with respect to the operations of vector addition and multiplication on V.

The necessary condition: It is assumed that, the number of Redundant Cycle (RC) in the synthesized $CA \in S_{CA}$ is 3. On exclusion of trivial cycle (with all 0 state), the number of RCs is assumed to be 2 in the rest of this paper. However, its generalization (# $RC \geq 2$) can be easily implemented.

The prohibited pattern set $\{PPS\}$ is entirely covered by the two non-trivial cycles of RC implies that, the PPS should get divided to two disjoint pattern sets (PPS_1 & PPS_2) where PPS_1 falls in the subspace S_1 and PPS_2 in subspace S_2. The following theorem formalizes the necessary condition for achieving an expected solution - that is the existence of a linear opertor T with unrestricted neighborhood.

Theorem 1: If the rank of a pattern set (PPS) is $k \leq n$ and k_1 & k_2 are the ranks of two disjoint subsets PPS_1 & PPS_2, where $PPS_1 \cup PPS_2 = PPS$, then a linear operator T of rank n will generate two disjoint subspaces containing PPS_1 & PPS_2 respectively only when $k_1 + k_2 \leq n$.

Proof: Let us assume that u and v are the vector space encompassing PPS_1 and PPS_2 respectively. Hence, the dimesion of $u \leq k_1$ and the dimesion of

IEEE COMPUTER SOCIETY

$v \leq k_2$. If V is the direct sum of u and v - that is $u+v$, then the dimension of V is at least $k = k_1 + k_2$. A linear operator T of rank $\geq k$ can be constructed to generate V. Hence, to construct a T with rank n, the condition $k_1 + k_2 \leq n$ must be satisfied. ∎

In order to illustrate the result of *Theorem 1*, let us assume that the PPS (of rank $k = 7$) noted in *Fig.2(a)* be broken up into

$$PPS_1 = \begin{matrix} 0000110 \\ 0000010 \\ 0001000 \\ 0000111 \\ 0001111 \end{matrix} \quad \& \quad PPS_2 = \begin{matrix} 0110100 \\ 1101101 \\ 1011001 \\ 0100100 \\ 0010001 \end{matrix}$$

where the rank of PPS_1 and PPS_2 are $k_1 = 4$ and $k_2 = 4$ respectively. Since, $k_1 + k_2 \neq k$, we can conclude that there is no such CA which accommodates the pattern set PPS_1 and PPS_2 in its two RCs.

By contrast, if PPS_2 gets modified to $PPS_2^{'}$ (as noted below) and $PPS' = PPS_1 \cup PPS_2^{'}$,

$$PPS_1 = \begin{matrix} 0000110 \\ 0000010 \\ 0001000 \\ 0000111 \\ 0001111 \end{matrix} \qquad PPS_2^{'} = \begin{matrix} 0110100 \\ 1101101 \\ 1011001 \end{matrix}$$

then the rank of PPS', PPS_1 *and* $PPS_2^{'}$ are 7, 4 and 3 respectively. Now, since $k = k_1 + k_2$, the required subspaces exist and the pattern set PPS_1 & PPS_2' can fall in two separate RCs of a group CA.

The sufficient condition: If a pattern set PPS satisfies the necessary condition, it results in a linear operator T. However, the derived T should satisfy the 3-neighborhood restriction. The sufficient condition for the existence of a desired 3-neighbourhood CA based TPG is formulated in the next theorem.

Theorem 2: Let a given PPS be partitioned into disjoint subsets PPS_1 & PPS_2 with basis $b_1 = \{u_1, u_2, \cdot\cdot, u_m\}$ and $b_2 = \{u_{m+1}, u_{m+2} \cdot\cdot, u_n\}$ respectively. A CA can be synthesized with elements of PPS_1 and PPS_2 in its two different cycles if each individual basis $\{b_1, b_2\}$ is a valid basis of the subspace of the CA

A heuristic scheme is proposed to achieve a faster but approximate solution for identifying $S'_{CA} \subseteq S_{CA}$ while verifying the necessary & sufficient conditions for each member of the S_{CA} derived for the given PPS.

B. The Heuristic Solution

The problem defined in the previous *Section* is hard. However, verification of each solution that satisfies the sufficient and necessary conditions (noted in the earlier subsection) is accomplished in polynomial time. The cardinality of PPS for all practical purpose is very small (we have assumed it to be at most 25). Moreover, the set of basis which supports a valid 3-neighborhood CA is a small subset of the set of basis represented by any linear operator T. This fact drastically reduces the solution space. Further, the design does not require the strict inclusion of all the patterns ∈ PPS in the RCs; a few of these may as well be included in the TC (Target Cycle) satisfying the constraint C_3.

Acceptable criteria: After exhaustive experimentation we set the values of the following parameters in order to ensure the desired pseudo-random qualities of the patterns generated by the TC employed for the TPG. The approximate solution is acceptable only if: (i) the 75% of PPS falls in the RCs, (ii) the TC, generating the test pattern sequence, is not less than q=50% of the maximal length ($(2^n - 1)$ for an n-cell CA) with $n > 16$, and (iii) the value of D_{max} (maximum distance lost in the TC to avoid generation of any PPS element) is 10 % of the cycle length.

For a given PPS the verification algorithm performs the following basic tasks on the members of S_{CA}

1. Finding the basis of the RCs.
2. Estimation of the number of prohibited patterns that fall in the RCs.
3. Computation of the value of D_{max} in case a few members of PPS are covered by the TC.

These three tasks are elaborated with illustration.

Task 1. Finding the Basis of the CA subspaces generated by the Redundant Cycles RCs: The synthesis of CA is followed by enumeration of basis of the each individual subspace generated by the RCs of the CA. The elements of subspaces - that is, the elements lying in the RCs of length l_1 & l_2 (say) can be found out by evaluating the null space [1] of $T^{l_1} + I$ and $T^{l_2} + I$. The basis $\{a_1, a_2, \cdot\cdot, a_{l_1}\}$ and $\{b_1, b_2, \cdot\cdot, b_{l_2}\}$ for each individual set can be computed by any standard basis-finding algorithm (Row_Space Algorithm and Casting_Out Algorithm) [2].

Example 1: The basis of RCs (cycles III and II) of the CA in *Fig.2(b)* are $A = \{a_1 = 0000001$, $a_2 = 0000010$, $a_3 = 0000100$, $a_4 = 0001000\}$ & $B = \{b_1 = 0011111$, $b_2 = 0101011$, $b_3 = 1000110\}$.

Task 2. Estimation of the number of prohibited patterns in the RCs: On execution of *Task1* we have a candidate CA with (say) two subspaces S_1 and S_2 and their basis corresponding to the two RCs. A pattern ∈ PPS falls in the subspace S_1 or S_2 if it can be generated by either of the basis set. Thus the percentage of prohibited patterns covered by the RCs can be computed as illustrated in the following example.

Example 2: It is possible that the 8 patterns of PPS in *Fig.2(a)* can be represented by the basis A and B of *Example 1*. The patterns that fall in cycle III are represented by A, whereas the patterns represented from B fall in cycle II.

IEEE COMPUTER SOCIETY

Patterns represented by A	*Patterns represented by B*
$0000110 = a_2 + a_3$	$0110100 = b_1 + b_2$
$0000010 = a_2$	$1101101 = b_2 + b_3$
$0001001 = a_1 + a_4$	$1011001 = b_1 + b_3$
$0000111 = a_1 + a_2 + a_3$	
$0001111 = a_1 + a_2 + a_3 + a_4$	

The percentage of prohibited patterns covered in the two RCs = 8/10= 0.8 or 80 %.

Task 3. Computation of D_{max}: The patterns that are not covered by the RCs must fall in the TC. Let PPS'' is the set of prohibited patterns that fall in the TC. For every pattern $P_i \in PPS''$, load the CA with P_i and then run for D_i time steps to cover all the patterns in PPS''. Then the D_{max} can be computed as $D_{max} = min(D_i), \forall i$.

Let the terminal states of the entire span (denoted as L_{max}) covered in the TC be P_1 and P_2. The TPG should be loaded with a seed P_2, where $T^{D_{max}}(P_1) = P_2$. The TPG will not encounter any member of PPS till it reaches P_1. If D_{max} is within the tolerable limit, the CA is accepted. Otherwise, the search for a better CA is continued.

Example 3: The patterns $\{P_1$=001001 & P_2=0100100$\}$ are not covered by the two RCs *(Fig.2(b))*. To get D_{max}, T_1 is multiplied with P_1 until $T_1^{d_1} \cdot P_1 = P_2$. Here, D_{max}=d_1=10. So the TPG desiged with the TC can generate 105-11=94 test patterns.

The logical steps for the complete design of the TPG are given in the following algorithm.

Algorithm 1: Design_TPG
Input : Prohibited pattern set PPS, $n - PI\ CUT$
Output : *(i)* CA based TPG *(ii)* seed *(iii)* test results (fault coverage, no. of test patterns, etc) for the CUT
Iterate for a number of times {
Step 1: Randomly synthesize a non-maximal length group CA with required cycle structure - that is generate a member of S_{CA} satisfying the constraint C_1
Step 2: Identify the TC and RCs
Step 3: Find the basis of the RCs
Step 4: Partition the PPS with respect to the bases of the RCs
Step 5: Find percentage of prohibited patterns covered by RCs
Srep 6: Find D_{max} to fit the rest of the PPS covered by the TC
Step 7: Check whether the CA meet the Acceptable Criteria
If yes then select the CA as the TPG else
Iterate for the next CA }
Step 8: Find a set of valid seeds in the TC of the selected CA & run the CA for maximum of L_{max} cycles generating the test patterns
Step 9: Evaluate fault coverage of the CUT with each of the valid seeds
Step 10: Select the seed with maximum fault coverage

IV. Experimental Observation

Real life data in respect of PPS for a CUT is propitory in nature and not usually available. In the absence of real life data, the experiment is conducted for different randomly generated PPS. The number of prohibited patterns for a CUT is expected to be very small and we have set the value as 25 for a CUT. The success rate of the proposed solution will be substantially better with real life PPS data which are expected to have certain correlation rather than being random in nature.

TABLE I
SUCCESS RATE OF THE TPG DESIGN

(1) # Cell	(2) $\|PPS\|$	(3) TC	(4) RCs	(5) (%) PPS in RCs	(6) D_{max}	(7) Avg # Iter^n
9	9	465	15,31	75	48	25
14	15	14329	7,2047	80	1223	20
14	15	8191	1,8191	95	106	23
16	20	57337	7,8191	65	21259	50
16	20	32767	1,32767	97	259	17
17	25	65535	1,65535	94	1000	25
18	25	131072	1,131072	98	336	13
24	25	$2^{23} - 1$	1,(2^{23}-1)	84	18121	14
26	25	$2^{25} - 1$	1,(2^{25}-1)	78	42342	14
32	25	*	(2^{15}-1),(2^{17}-1)	89	33571	16
33	25	*	(2^{16}-1),(2^{17}-1)	95	17498	21
35	25	*	(2^{17}-1),(2^{18}-1)	97	7853	12
36	25	*	(2^{17}-1),(2^{19}-1)	95	14322	18
41	25	*	(2^{20}-1),(2^{21}-1)	82	31132	14
43	25	*	(2^{21}-1),(2^{22}-1)	93	20211	15

* indicates that the cycle length ≈ $2^n - 2^{n/2}$

Table I depicts the summary of the success rate in designing the TPG that generates good quality pseudo random patterns while avoiding generation of the given PPS. The value of n and cardinality of PPS are noted in *Column* 1 & 2. *Column 3* denotes the length of TC, while *Column 4* displays the length of the RCs. For a particular value of n, the experimentation is done for 10 different randomly generated PPS. The average percentage of prohibitive patterns which are covered by the RCs is noted in *column 5.* The value of D_{max} is given in *Column 6.* Finally, the average number of iterations taken to arrive at the solution of identifying the CA based desired TPG is noted in the last column.

Study of randomness property: The randomness property of the patterns generated by the TC, for different values of n, are studied, based on the metric proposed in [8] and *DiehardC* [9]. DiehardC 1.01 is a public domain tool which supports randomness testing of a set of patterns. It consists of 15 different tests. The results of 10 tests are noted in *Column 1 of Table III.* 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.

A comparative study on randomness quality of the patterns generated by the proposed TPGs and the corresponding maximal length CA is presented in *Tables II* & III. *Column 1* depicts the names of the 6 tests. The columns under the heading of 'Max' specify the test results for an n-cell maximal length CA, while the columns under TPG signify the result out of the patterns generated by the proposed design. Each of the tests is performed for a number of runs with different seeds. The results noted for maxlength CA and the proposed TPG are the average of the results produced with different seeds. Here 'pass' implies that the test succeeds at least for 75% cases.

TABLE II
RANDOMNESS TEST I

Random Test	$n=9$ Max	$n=9$ TPG	$n=15$ *to* 20 Max	$n=15$ *to* 20 TPG
Gap test	pass	pass	pass	pass
Run test	pass	fail	pass	pass
Serial corr test	pass	pass	pass	pass
Equidist. test	fail	fail	fail	fail
Auto-corr test	pass	pass	pass	pass
Cross-corr test	pass	fail	pass	pass

TABLE III
RANDOMNESS TEST II

Random Test	$n=24$ Max	$n=24$ TPG	$n=32$ Max	$n=32$ TPG	$n=48$ Max	$n=48$ TPG
Overlap Sum	pass	pass	pass	pass	pass	pass
Run	pass	pass	pass	pass	pass	pass
3Dsphere	pass	pass	pass	pass	fail	fail
Parking lot	fail	fail	fail	fail	fail	fail
B'day spacing	fail	fail	fail	fail	fail	fail
Craps	pass	pass	pass	pass	pass	pass
Minimum Dist	fail	fail	fail	fail	fail	fail
Overlap 5-permut	fail	fail	fail	fail	pass	pass
DNA	fail	fail	fail	fail	pass	fail
Squeeze	fail	pass	fail	fail	pass	fail

The results reported in the *Tables II*& III establish the fact that the randomness quality of the proposed TPG is as good as that of maximal length CA.

The fault coverage: It is observed that the TPG designed with the proposed scheme is as powerful as the corresponding maximal length CA based test pattern generator in respect of fault coverage and number of test patterns required to achieve the desired fault coverage. The fault simulation is done for a large number of *ISCAS benchmark* circuits in the framework of *Cadence* fault simulator *verifault*. *Table IV* compares the fault coverage shown by maximal length CA and the TC of the proposed TPG in *Column* 4 and 5 respectively. The fault coverage figures are expressed in terms of

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

while the FFs of the sequential circuits are assumed to be initialized to 0. A benchmark circuit is tested with the same number of test vectors, noted in *Column 3*, for both the designs.

Table IV reports the test results of 21 combinational and sequential circuits. It can be observed that out of 21 cases, the fault coverage of the proposed TPG:

(i) is same or better for 8 cases (marked with *), and

(ii) worse for 13 cases

than the result obtained with maximal length CA.

The difference in fault coverage between the two schemes is marginal and can be reduced by refining the heuristics employed. Hence, the proposed TPG achieves the goals of generating good quality patterns without generating the given PPS for the CUT.

TABLE IV
COMPARISON OF TEST RESULTS

Circuit Name	# PI	# Test Vector	Fault Coverage (%) Max Len	Fault Coverage (%) TPG
s349	9	400	84.00	84.00 *
s344	9	400	84.21	84.21 *
s1196	14	12000	94.85	94.04
s1238	14	10000	89.67	89.08
s967	16	9000	98.22	98.12
s1423	17	15000	56.50	53.60
s1269	18	1200	99.18	99.48 *
s3271	26	10000	98.99	98.99 *
c6288	32	60	99.51	99.43
c1908	33	4000	99.41	99.41 *
s5378	35	8000	67.63	67.72 *
s641	35	2000	85.63	85.08
s713	35	2000	81.41	80.72
s35932	35	14000	61.91	59.82
c432	36	400	98.67	99.24 *
c432m	36	4000	83.57	83.96 *
c499	41	600	98.95	98.68
c499m	41	2000	97.78	97.22
c1355	41	1500	98.98	98.11
c1355m	41	12000	92.23	92.17
s3384	43	8000	91.78	91.60

V. Conclusion

The paper presents an elegant solution for the problem of designing a TPG that generates good quality pseudo random test patterns while avoiding generation of Prohibited Pattern Set (PPS) for a given CUT. The reported solution does not incur any extra area overhead than the conventional $CA/LFSR$ based TPG. Exhaustive experimentation confirms that the the TPG maintains the fault efficiency in a CUT that could be achieved through a maximal length $CA/LFSR$ based design.

References

[1] P Pal Chaudhuri et. al. *'Additive Cellular Automata Theory and Applications'*, IEEE Computer Society Press, California, USA, 1997.

[2] I.N. Herstein *'Topics in Algebra'*, John Wiley & Sons Inc, 1987.

[3] J. Von Neumann *'The Theory of Self-Reproducing Automata'*, A. W. Burks, ed., Univ. of Illinois Press, Urbana and London, 1966.

[4] A. W. Burks *'Essays on Cellular Automata'*, Tech report, Univ. of Illinois, Urbana, 1970.

[5] Stephen Wolfram *'Theory and Applications of Cellular Automata'*, World Scientific ,1986.

[6] P. D. Hortensius et. al. *'Cellular Automata Based Pseudo-random Number Generators for Built-in Self-test'*, IEEE Trans. on CAD, 1989.

[7] N. Ganguly *'Cellular Automata Evolution: Theory & Applications'*, Phd Thesis proposal, B E College (D U), 2002.

[8] D. E. Knuth *'The Art of Computer Programming - Seminumerical Algorithms'*, Addison-Wesley, Reading, Mass, 1981.

[9] DiehardC : http://stat.fsu.edu/ geo.

0-7695-1441-3/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY