EBIST: A Novel Test Generator with Built-In Fault Detection Capability

Dhiraj K. Pradhan
University of Bristol, UK
pradhan@cs.bris.ac.uk

Chunsheng Liu
Duke University, USA
mliu@ee.duke.edu

Krish Chakraborty
Duke University, USA
krish@ee.duke.edu

www.cs.bris.ac.uk/~pradhan

Abstract : A novel design methodology for test pattern generation in BIST is presented. Here faults and errors in the generator itself are detected. Two different design methodologies are presented. The first one guarantees all single fault/error detection and the second methodology is capable of detecting multiple faults and errors. Furthermore the proposed LFSRs do not have additional hardware overhead. Also importantly the test patterns generated have the potential to achieve superior fault coverage.

1 Introduction

Built-in-Self-Test is widely used for both detection of manufacturing defects as well as for detection of operational faults. The basic components of the BIST scheme are illustrated in Fig 1. The test pattern generator (TPG) is usually built using a LFSR. Similarly, a test response analyser (TRA) unit computing the signature is also built using a LFSR. The standard practice is to assume that these LFSRs are hard core logic in that they cannot fail either in field or during test, or are to be tested before any test applied to circuit under test (CUT). What we propose here is a novel design methodology shown in Fig. 2 that allows for detection of errors that may be induced by either a manufacturing defect or operational fault in the LFSR. This removes a major short-coming in BIST methodology [2]. Assuming that a BIST circuit is built out of same components as CUT it is unrealistic to assume that BIST hardware is defect/fault free. It is also important to locate the fault and determine whether the TPG or the CUT is faulty for the following reasons:

(a) Often BIST technology is inserted after the design is completed. So potential design flaws that can be introduced at this stage can affect the overall test coverage. The ability to distinguish can be important.

(b) After a fault occurs in an ultra reliable system where a repair may be impossible. Our proposed design methodology can enhance reliability when the fault is located to the TPG.

(c) When a CUT inputs are parity protected (e.g., in a bus) the proposed design can be very useful in distinguishing input faults from those in CUT.

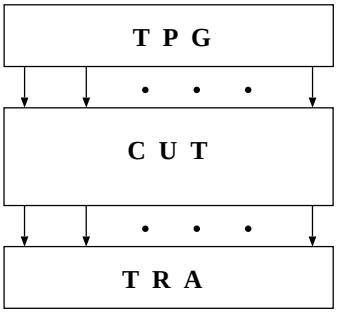

Figure 1. BIST Scheme

The hardware overhead for the proposed fault-detecting test pattern generator (TPG) is minimal. There is no extra logic for the LFSR itself. The only overhead is in the error detecting circuit which requires Ex-or gates and OR gates. As shown later, these additional gates are external to the LFSR itself and therefore do not impede the performance of BIST. This is important because BIST can often be used for at speed testing for delay faults. Also shown in this paper is that these new LFSR designs have the potential for achieving higher fault coverage than standard LFSR.

Basically, we propose two types of design. First we propose a design where all the test patterns generated have the same parity, thus allowing for all single fault/error detection. Secondly, the design allows multiple fault detection where the Test Pattern Generator (TPG) states correspond to a nonzero codewords of same cyclic code. The proposed designs are based on earlier work reported in [1].

The second design allows designing LFSR where tests generated are separated by a minimum distance of d specified by the designer. Both the designs as shown in the experimental result section has the potential to achieve higher fault-coverage than the standard LFSR based TPGs.

This paper is organised into three main sections. The second section describes the design methodology and de-

1530-1591/03 $17.00 © 2003 IEEE

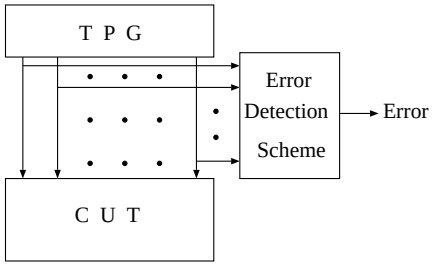

Figure 2. Enhanced BIST (EBIST)

velops the mathematical framework for parity preserving LFSR design followed by the third section describing the cyclic codewords generating LFSR. The fourth section presents experimental results. The final section presents future extension and conclusions.

2 Parity Preserving Test Pattern Generator for Single Fault and Error Detection Capability

This section develops the theory and design methodology for the proposed test pattern generator with built-in fault/error detection capability. This enhanced feature is a byproduct of our design methodology.

First we develop the theory behind the design followed by implementation details. The proposed design is no more complex than the standard LFSR, used in current technology, therefore making it an alternative attractive.

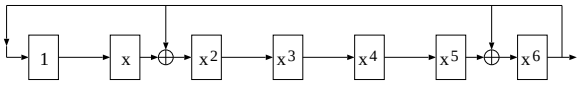

Figure 3. LFSR for distance 2

Now consider the LFSR shown in Fig. 3. This also has 7 stages. Let the initial state of the LFSR be 0000001. The successive states are shown in Table 1.

This above table illustrates the first 11 states. The total number of states is 63. It may be noted that the minimum distance between any pair of test vectors is 2. However, unlike the previous LFSR these 63 vectors do not constitute codewords in a code. But the minimum distance of 2 guarantees all single fault/error detection in the TPG as shown in Fig. 4.

Since the tests all have odd parity the parity check circuit will produce 1 for error free tests and 0 for erroneous tests. This allows for a self-checking parity circuit in that any fault within a parity circuit can be detected by the circuit shown in Fig. 5. The two output self-checking parity produces (1,0) or (0,1) in the absence of any faults in the

00	0000001
01	1010001
02	1111001
03	1101101
04	1100111
05	1100010
06	0110001
07	1001001
08	1110101
09	1101011
10	1101011
11	1100100

Table 1. The first 11 tests generated by parity preserving LFSR in Fig. 3

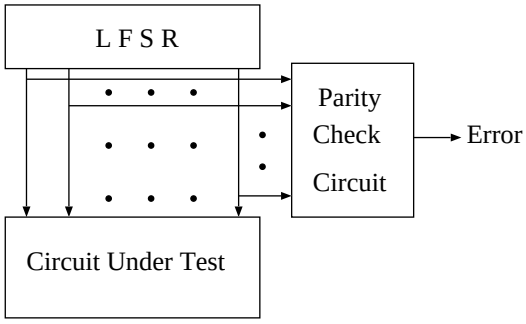

Figure 4. On-line and off-line detection in TPG

parity circuit. However if there is a single fault in the error detection circuit itself it will be detected because this will result in an (0,0) or (1,1) output for some test generated by LFSR.

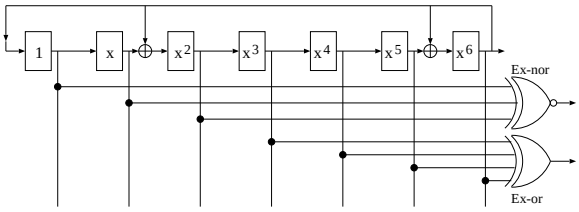

Figure 5. Self-Checking Parity Circuit for Fig. 3

The following develops the theory behind this design.

First we discuss some of the basic characteristics of LFSR and also introduce some notations. The general form of an n-stage LFSR is shown in Fig. 6. In an LFSR the in-

formation is assumed to be some representation of elements of a Galois field, and it has no external inputs except a clock used for shifting.

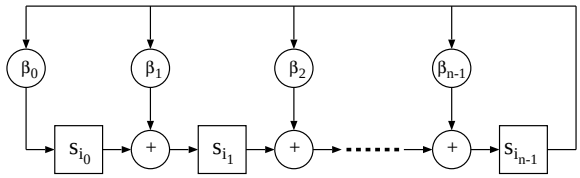

Figure 6. Autonomous linear shift register

The states S_i are symbolic representation of the contents of an LFSR during successive shifts, given the initial content as S_0. Using the theory of LFSR, knowing its initial contents and feedback connections, one can algebraically compute any state of the LFSR.

The contents of an n-stage LFSR can be represented by an $n-1$ degree polynomial over $GF(2)$. Let $S_{i_0}, S_{i_1}, \cdots, S_{i_{n-1}}$ represent the contents of n delay elements for the ith state S_i (as shown in Fig. 6), where S_{i_k} is an element of $GF(2), 0 \leq k \leq n-1$. Then $S_i(x)$ is the polynomial representation of S_i, given as $S_i(x) = S_{i_0} + S_{i_1}x + \cdots + S_{i_{n-1}}x^{n-1}$.

The feedback connections are also represented as a polynomial $\phi(x)$ of degree n, referred to as a characteristic polynomial and given as $\phi(x) = x^n + \beta_{n-1}x^{n-1} + \cdots + \beta_0$, where β_k, an element of $GF(2)$, is the scalar multiplier used in the feedback connection of the kth stage, as shown in Fig. 6. Thus if $\beta_k = 0$, then there is no connection from the feedback path. If $\beta_k = 1$, then the feedback path is connected with the Ex-or gate. The following is a fundamental relationship between the states in a cycle: $S_i(x) = x^{i-j}S_j(x) \bmod \phi(x)$.

Furthermore, if T is the least positive integer such that $\phi(x)$ divides $x^T - 1$, then for any state $S_i(x)$; $S_i(x) = x^T S_i(x) \bmod \phi(x)$.

The integer T is called the exponent of $\phi(x)$ and the period of the LFSR. In the sequel, $p(x)$ will denote a primitive polynomial of degree k in $GF(2)$; i.e., the exponent of $p(x)$ is $2^k - 1$.

Let $S_0(x)$ be the $(n-1)$ degree polynomial representing the initial state and $\phi(x)$ is the $(n-1)$ degree feedback polynomial. The states of the LFSR after the ith shift $S_i(x)$ can be represented as $x^i S_0(x) mod\ \phi(x)$.

$S_i(x) = x^i S_0(x) mod \phi(x)$ can also be represented as $x^i S_0(x) = \phi(x)g(x) + S_i(x)$, where $S_i(x)$ is the $(n-1)$ degree polynomial obtained as the remainder dividing $x^i S_0(x)$ with $\phi(x)$.

Lemma 1 *If $\phi(x=1) = 0$, then the parity of all the states reached by the LFSR is the same as that of the initial state.*

Proof First it may be noted that $x^i s_0(x)$ has the same parity as $s_0(x)$ as long as $x^i s_0(x)$ has degree less than or equal to $(n-1)$. However when the degree of $x^i S_i(x)$ exceeds $(n-1)$ then $s_i(x) = x^i s_0(x)\ mod \phi(x)$. Now it may be noted that $x^i s_0(x) = g(x)\phi(x) + s_i(x)$ where $s_i(x)$ is the residue of dividing $x^i s_0(x)$ by $\phi(x)$. Since $\phi(1) = 0$, we have
$s_i(1) = g(1)\phi(1) + s_0(1)$
Since $\phi(1) = 0$, one has $s_i(1) = s_0(1)$.
Thus the parity of the ith state is the same as the initial state if $\phi(1) = 0$. ■

Consider for example the LFSR shown in Fig 3 and the states shown in Table 1. The parity of the constants of LFSR is odd. This guarantees all single error detection by the parity circuit shown below.

This circuit is guaranteed to detect all single errors. The following is a direct consequence of the above.

The above Lemma gives a wide variety of choices for selecting single error detecting LFSR. For example, one can design a single error detecting LFSR of cycle length much smaller than $(2^n - 1)$ where n is the size of the LFSR.

Theorem 1 *Any LFSR with an even number of Ex-or gates in the feedback connections is parity preserving.*

Proof The proof is a direct consequence of Lemma 1. ■

First it may be observed that the number of inputs n to the CUT can be very large but the number of test patterns to be applied can be significantly smaller than $(2^n - 1)$. This provides an interesting optimisation problem.

Design a parity preserving LFSR with a minimum number of Ex-or gates that will produce N tests for a n input circuit where $N \ll (2^n - 1)$. The implication of Theorem 1 says that as long as the number of feedback taps are even, one is guaranteed single fault detecting design. This requires that $\phi(x)$ the feedback polynomial has an even number of terms. Since the term x^n and 1 is always present the problem therefore reduces to having a minimum number of other x^i terms where $i \neq 0$ and $i \neq n$. Also the required $\phi(x)$ must have an even number of these x^i terms.

The following example motivates this problem.

Consider a 7 output TPG. If it is needed that the TPG is required to produce only 7 states instead of the full 127 potential states that can be achieved by the 7 bit LFSR, if we use a degree 7 primitive polynomial, then one has various choices in the design. First select a degree 3 primitive polynomial, then multiply this with $(x^4 + 1)$ to obtain the feedback polynomial. The degree 3 primitive polynomial guarantees that at least 7 states will be produced.

Let $p(x) = x^3 + x + 1$ which is a primitive polynomial. The feedback polynomial can be obtained as $\phi(x) = p(x)(1 + x^4) = x^7 + x^5 + x^4 + x^3 + x + 1$.
This LFSR, shown in Fig. 7, when initialized to 0000001, an odd parity vector, will produce 7 tests with odd parity. However a trivial example with no Ex-or gates will produce 7 tests with odd parity as shown in Fig. 8. This therefore poses an interesting optimisation problem.

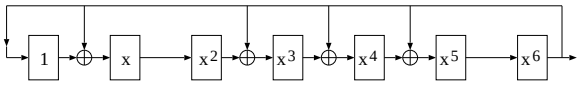

Figure 7. LFSR with feedback polynomial ($x^7 + x^5 + x^4 + x^3 + x + 1$)

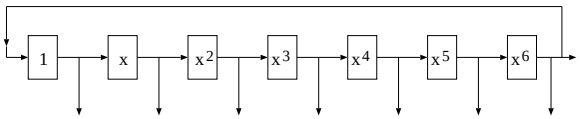

Figure 8. Optimal single parity preserving LFSR to producing 7 tests with odd parity

3 Test Pattern Generation Using Cyclic Code

To motivate this section we first present the following example. Consider the LFSR shown in Fig. 3 below. Let the initial state of the LFSR be 1101000. Table 1 illustrates the states (tests) generated by successive shifts in the LFSR. The cyclic length of this LFSR is 15. The initial state is reached again after 16 shifts. It is important to note that the vectors are separated by distance 3 or more. Thus all single and double errors in the tests can be detected. In this case the LFSR generates all non-zero codewords in a (7,4) cyclic Hamming code.

00	1101000
01	0110100
02	0011010
03	0001101
04	1011100
05	0101110
06	0010111
07	1010001
08	1110010
09	0111001
10	1000110
11	0100011
12	1001011
13	1111111
14	1100101
15	1101000

Table 2. Fifteen Nonzero Code Words of Hamming (7,4) Code

A list of code words of any cycle code can be used as test stimuli. The code structure of the cyclic code, in that case, provides error detection and/or correction over faults in the test pattern generator. In this section, it is shown that all nonzero code words of any cyclic linear code can be generated in a fixed and ordered chain by means of an autonomous linear-feedback shift register. The initial content of this shift register is chosen to be any code word. Each shifting operation in the register produces the next code word in the ordered chain as a linear function of the symbols in the previous code word.

Basically, such a pattern generator creates test patterns in a predetermined sequence of arbitrarily long length and simultaneously provides an error detection and/or correction capability of any desired level. This test pattern generator can be implemented as a software algorithm or by means of a feedback shift register constructed from binary storage and logic elements. The algorithm creates each new test pattern by simple binary operations on the binary digits of the previous test.

In each test, the last r bits are parity checks over the other k bits, and they form a legal code word of a cyclic $(k + r, k)$ code. The parity-check computations, however, are not explicit; they are only the results of the characteristic polynomial and the initial state of the register. This is the unique property of this test pattern generator which is not available with any other known test pattern generating means. An (n, k) cyclic code is a k-dimensional subspace of the n-dimensional vector space over $GF(2)$. The cyclic shift of any code word is also a code word. This code is usually characterized by means of an $r = n - k$-degree polynomial $g(x)$ over $GF(2)$, known as the generator polynomial. Any n-vector over $GF(2)$ is a code word in the code if and only if its $n - 1$-degree polynomial representation is divisible by $g(x)$. An additional property of $g(x)$ is that it always divides $x^n - 1$. Any errors in a code word can be represented by an n-vector E. An error E is detectable if an only if $E(x)$ is not divisible by $g(x)$. This condition guarantees that the corresponding error does not change a code word to another code word. Detailed treatment of cyclic codes can be found in any book on error-correcting codes.

The following, we present a technique to design LFSRs in which the initial contents when set to one of the code words will thereafter generate all the nonzero code words of a (n, k) cyclic code; i.e., $2^k - 1$ consecutive shifts of the shift register will produce $2^k - 1$ distinct nonzero words of the code in a fixed order. The following Theorem is from [1].

Theorem 2 *Let $\phi(x) = g(x) \cdot p(x)$, where $g(x)$ is a generator polynomial of a (n, k) cyclic code and $p(x)$ is a primitive polynomial of degree k. The $2^k - 1$ consecutive shifts of an LFSR characterized by polynomial $\phi(x)$ produce all $2^k - 1$ nonzero code words of the cyclic code generated by $g(x)$, provided the initial content of the LFSR is one of these nonzero code words.*

Following is an example for a circuit with 55 inputs.

Example 1 *Consider a CUT which has 55 inputs. One can use the following construction which can generate test patterns with distance 4. This will allow detection of up to 3 errors. Altogether* $2^{48} - 1$ *tests can be generated by this generator. These constitute all the codewords in a (55,48) shortened Hamming code.*
$g(x) = (1 + x)(1 + x + x^6)$
$p(x) = 1 + x + x^2 + x^4 + x^5 + x^7 + x^{48}$
$\phi(x) = p(x) \cdot g(x) = 1 + x + x^3 + x^5 + x^{10} + x^{12} + x^{13} + x^{14} + x^{48} + x^{50} + x^{54} + x^{55}$.

4 Experimental Results

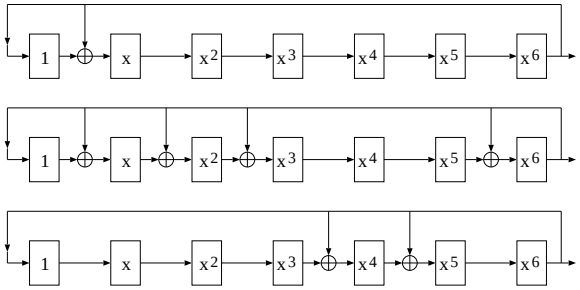

Figure 9. Different LFSR setup for the experiment

We perform experiments on some ISCAS85 benchmarks. First we compare the fault-coverage achieved by standard LFSR and parity preserving LFSR. The results demonstrate potential for higher fault-coverage for the proposed design. We have chosen odd parity for our experiment because this allows fault detection in the parity check circuit. In Table 3, there are two columns. The column 1 describes fault-coverage for standard LFSR where column 2 provides fault-coverage for parity preserving LFSR. It should be noted that we obtain improvement in fault-coverage in most cases. This when viewed with the fact that we also detect all single errors in LFSR can provide an attractive alternative to standard LFSR. Next for each benchmark, we consider six different types of LFSR pattern generators and we generate a set of test patterns using each LFSR. The test patterns are then applied to the circuit and fault simulation using FSIM is carried out to determine the fault coverage of each set of test patterns. We compare the value of fault coverage given by different sets of test patterns in Table 3. Column 2 shows the number of test patterns involved in a simulation. Column 3 gives the result given by an LFSR using a primitive polynomial. If we use $\phi_i(x)$ and $p_i(x), (i = 1, 2, 7)$ to represent the code polynomial associated with the LFSR and a primitive polynomial, respectively, then Column 3 can be characterized by $\phi_1(x) = p_1(x)$. In Column 4, we add a factor of $(1 + x)$, i.e., $\phi_2(x) = p_2(x)(1 + x)$. In Column 5 and 6, we use $\phi_3(x) = g_3(x)p_3(x)$ and $\phi_4(x) = g_4(x)p_4(x)(1 + x)$, respectively, where $g_i(x)$ is a generator polynomial whose degree is at least equal to the $\log(pi(x))$.

Circuit	# of test patterns	Fault coverage (%)	
		Non-parity Pres. Stand. LFSR	Parity Pres. LFSR
c6288	100	99.290	99.096
	1k	99.561	99.561
c3540	10k	95.916	95.799
	100k	96.004	96.004
c1908	10k	99.521	99.521
	100k	99.521	99.521
c1355	1k	98.475	95.616
	10k	99.492	99.492
c7552	100k	96.808	96.901
	1m	97.497	97.523
c2670	100k	88.387	85.948
	1m	93.557	93.265
c5315	10k	98.897	98.897
	100k	98.897	98.897
c880	10k	98.832	100
	100k	100	100
s1196	10k	98.390	97.504
	100k	100	100
s1238	10k	93.432	92.399
	100k	94.908	94.908
s1423	10k	98.548	98.614
	100k	99.076	99.076
s1488	10k	91.992	96.164
	100k	91.992	96.164
s1494	10k	91.102	95.485
	100k	91.102	95.485
s9234	100k	89.404	89.736
	1m	91.771	91.973

Table 3. Fault-coverage comparisons between standard LFSR and parity preserving LFSR with odd parity

The fault-coverage are aggregate averages over a wide variety of LFSR set ups as illustrated in Fig 9.

Example of order 7:
$g(x) = x^3 + x + 1$
$\phi_1(x) = p_1(x) = x^7 + x + 1$;
$p_2(x) = x^6 + x + 1, \phi_2(x) = p_2(x)(1 + x) = x^7 + x^6 + x^2 + 1$;
$p_3(x) = x^4 + x + 1, \phi_3(x) = g(x)p_3(x) = x^7 + x^5 + x^3 + x^2 + 1$ (this is a (7,4) hamming code);
$p_4(x) = x^3 + x + 1, \phi_4(x) = g(x)p_4(x)(1 + x) = x^7 + x^6 + x^3 + x^2 + x + 1$;

Circuit	# of test patterns	Fault coverage (%)			
		$\Phi_1(x)$	$\Phi_4(x)$	$\Phi_5(x)$	$\Phi_6(x)$
c6288	100	98.425	95.222	88.946	99.290
	500	99.561	99.535	99.561	99.561
	1k	99.561	99.561	99.561	99.561
c3540	1k	89.994	92.882	93.349	94.516
	10k	96.004	95.916	95.887	95.916
	100k	96.004	96.004	96.004	96.004
c1908	1k	95.317	96.807	94.359	96.381
	10k	99.468	99.468	99.361	99.521
	100k	99.521	99.521	99.521	99.521
c1355	500	88.945	95.680	89.708	95.299
	1k	98.094	98.920	94.473	98.475
	10k	99.492	99.492	99.492	99.492
c7552	1k	66.172	80.490	73.536	78.159
	10k	96.199	95.748	96.079	95.828
	100k	97.523	97.06	96.887	96.808
	1M	97.801	97.470	97.483	97.497
c2670	1k	52.239	57.080	46.050	66.254
	10k	84.092	84.347	74.736	84.092
	100k	88.751	85.402	84.565	88.387
	1M	94.066	93.411	93.775	93.557
c5315	1k	61.140	63.252	61.682	75.607
	10k	98.879	98.150	98.822	98.897
	100k	98.897	98.897	98.897	98.897
c880	1k	75.902	97.240	78.344	92.781
	10k	98.408	100	99.045	98.832
	100k	100	100	100	100

Table 4. Fault-coverage between standard LFSR and cyclic code generating LFSR

$p_5(x) = x^5 + x^2 + 1, \phi_5(x) = p_5(x)(1+x^2) = x^7 + x^5 + x^4 + 1;$
$p_6(x) = x^2 + x + 1, \phi_6(x) = g(x)p_6(x)(1 + x^2) = x^7 + x^6 + x^5 + x^4 + x^2 + 1.$
Since in this example, there exists an XOR between the highest two stages of $p_6(x)$ due to the low order, we omit this figure.

If we need an LFSR of degree N with no XOR gate between the highest two stages $N-1$ and N, such that an error in stage $N-1$ does not contaminate the entire feedback path before it is detected, then we can update $\phi_2(x)$ and $\phi_4(x)$ respectively with the code polynomials as follows: $\phi_2'(x) = \phi_5(x) = p_5(x)(1+x^2)$, $\phi_4'(x) = \phi_6(x) = p_6(x)g_6(x)(1+x^2)$.

The corresponding results given by these LFSRs are shown in Column 2, 3 and 4. We use one benchmark c6288 as an example to describe the polynomials associated with the LFSRs as follows (the number of inputs of c6288 is 32):
$\phi_1(x) = p_1(x) = x^{32} + x^{28} + x^{27} + x + 1;$
$p_2(x) = x^{25} + x^3 + 1, \phi_2(x) = g(x)p_2(x)(1 + x) = x^{32}+x^{31}+x^{27}+x^{25}+x^{10}+x^9+x^7+x^6+x^5+x^3+x^2+1;$
$p_3(x) = x^{30}+x^{16}+x^{15}+x+1, \phi_3(x) = p_3(x)(1+x^2) = x^{32} + x^{30} + x^{18} + x^{17} + x^{16} + x^{15} + x^3 + x^2 + x + 1;$
$p_4(x) = x^{24} + x^4 + x^3 + x + 1, \phi_4(x) = p_4(x)(1 + x + x^6)(1+x^2) = x^{32} + x^{30} + x^{27} + x^{26} + x^{25} + x^{24} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^4 + x^3 + 1.$

From the table, we observe that in most cases the use of LFSRs with larger Hamming distances (Column 2, 3 and 4) enhances the fault coverage. However, these LFSRs offer test patterns with error detecting as well as correcting ability. In addition, the fault coverage of using LFSRs with no XOR gate between the highest two stages (Column 3 and 4) is comparable with those given by other LFSRs. We further notice that the test generated can yield higher fault coverage than the original LFSR generator, especially when a relatively small number of test patterns are generated.

5 Conclusions

We have presented a new test pattern generation scheme using LFSRs with built-in fault/error detection capability. This is an integral part of the design and not obtained by augmentation as done for incorporating error detection capability. An interesting by product of our design methodology is that since the test vectors are separated by a specified minimum Hamming distance, it has the potential to achieve better fault coverage. Future research can examine other related designs [3, 4, 5, 6].

References

[1] M. Y. Hsiao, A. M. Patel and D. K. Pradhan: "Store Address Generator with On-line Fault-Detection Capability", *IEEE Transactions on Computers*, vol. c-26, no. 11, pp. 1144-1147, November 1977.

[2] G. Hetherington, T. Fryars, n. Tamarapalli, M. Kassab, A. Hassan and J. Rajski: "Logic BIST for Large Industrial Design: Real Issues and Case Studies", *Intl. Test Conference*, pp. 358-367, 1999.

[3] D. K. Pradhan and M. Chatterjee: "GLFSR - A New Pseudo-random Pattern Generator for BIST", *Intl. Test Conference*, pp. 481-490, 1994.

[4] D. K. Pradhan: "Shift Registers Designed for On-Line Fault Detection", *Proc. of FTCS 8*, Tolouse, France 1978.

[5] L. T. Wang: "Autonomous Linear Feedback Shift Register with on-line fault-detection", *Proc of FTCS 12*, pp. 311-314, 1982.

[6] N.S. Vasanthavada: "Group Parity Prediction scheme for concurrent testing", *Electronics Letters*, vol. 21, Jan 1985.