BIST Test Pattern Generators for Two-Pattern Pseudo-Exhaustive Testing

Chih-Ang Chen and Sandeep K. Gupta

CENG Technical Report 92-25

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-2251

August 1992

BIST Test Pattern Generators for Two-Pattern Pseudo-Exhaustive Testing*

Sandeep K. Gupta
Electrical Engineering – Systems
University of Southern California
Los Angeles CA 90089-2562

Chih-Ang Chen
Electrical Engineering – Systems
University of Southern California
Los Angeles CA 90089-2562

Abstract

There is a growing interest in testing circuits for stuck-open and delay faults. Testing for these faults require two pattern tests. Built-in self-test (BIST) schemes are attractive because typically long test sequences are required. Conventional BIST test pattern generators (TPG) designed for detection of stuck-at fault provide inadequate coverage of pattern pairs. This paper develops the notion of two pattern pseudo-exhaustive testing. Such testing guarantees coverage of all single and two pattern testable faults. A number of methodologies for design of such TPGs have been presented. The main design objective is to minimize the test sequence length. Concepts particular to two pattern testing have been identified and exploited to design entirely new and efficient TPGs. It has been proven that the TPGs designed by using the new construction proposed in this paper will guarantee shorter test sequences than the previously proposed design.

Keywords: BIST, test pattern generators, pseudo-exhaustive testing, two pattern testing, delay faults, CMOS stuck-open faults.

*This research was funded by NSF Research Initiation Award no. MIP-9210871

1 Introduction

Desire for higher quality levels in the semiconductor industry is causing an increasing acceptance of fault models such as CMOS stuck-open and delay faults. With need to develop high speed circuitry, aggressive statistical clocking is being adopted by many designers. This implies that a certain fraction of the manufactured chips, which have all the devices working within their worst case delays, may not perform at the design clock rate. Hence, delay testing of chips that pass the logic tests, is necessary to identify and reject such chips.

Testing for stuck-open and delay faults require two pattern tests. The conventional BIST schemes have emphasized the design of TPGs to maximize coverage of stuck-at faults in combinational logic. Such TPGs do not guarantee high pattern pair coverage, necessary for testing these faults which require two pattern tests. Hence, the main focus of this paper is on the design of TPGs which ensure a high coverage of pattern pairs required to test the various parts of the circuit under test (CUT).

Exhaustive testing is very time consuming if the number of inputs to the circuit is large. Test time for two-pattern exhaustive testing is even higher. In [CG93] a technique for design of test pattern generator (TPG) was proposed to reduce the test time (and TPG hardware complexity) for two-pattern testing by a minimal reduction in pattern pair coverage. This technique is the only recourse if any circuit output depends on all its inputs. However, in most multi-output circuits all outputs depend only on subsets of inputs. For such circuits, it is possible to apply two-pattern exhaustive tests to *each output* and all the logic feeding it, and still reduce the total test time. The focus of this paper is to develop efficient techniques which minimize test sequence length by taking advantage of the dependence of all circuit outputs on subsets of inputs only.

Note that, major advantages of two-pattern exhaustive testing — such as complete coverage of all single and multiple stuck-at faults, complete coverage of all detectable delay and stuck-open faults, etc. — still apply for two-pattern pseudo-exhaustive testing. However, not all one and two pattern testable bridging faults may be tested. One can only guarantee the coverage of all one and two pattern testable bridging faults between nodes within any

cone.

Design of TPGs which guarantee two-pattern pseudo exhaustive testing in minimal test sequence length is the main objective of this research. A range of TPG design methodologies are presented. These represent test time vs. TPG hardware complexity tradeoff. New concepts related to two pattern pseudo-exhaustive testing have been identified and exploited to develop a new TPG design procedure. It has been proven that TPGs designed using this procedure generate shorter test sequences than other TPGs.

The paper is organized into four main sections. Section 2 covers the basic concepts in pseudo-exhaustive and two pattern testing. If also includes a review of the previous results. Section 3 discusses a number of techniques for design of efficient TPGs two pattern pseudo-exhaustive testing. Experimental results and a comparison of different TPGs are in Section 4. Finally, the conclusions and directions for future research are presented in Section 5.

2 Basics and Review

2.1 Pseudo-Exhaustive Testing — Single Pattern

Most real life circuits have multiple outputs. An output y_j of a circuit is said to *depend* on an input x_l if there exists a directed path (via gates and fan out branches) from x_l to y_j. Typically each output of a circuit depends only on a subset of its inputs. Such circuits are called partial dependence (PD) circuits. Consider an circuit with n inputs $\{x_1, x_2, \ldots, x_n\}$ and m outputs $\{y_1, y_2, \ldots, y_m\}$. Let the j-th output y_j be a function of k_j inputs, i.e. $y_j = f_j(x_{l_1}, x_{l_2}, \ldots, x_{l_{k_j}})$. Let all the gates and inputs that are in the transitive fan-in of y_j be called the $cone_j$. An input x_l is said to *belong* to $cone_j$, if y_j depends on x_l. Let $k = \max_j^m k_j$. Such a PD circuit is called an (n, k) circuit. The logic in $cone_j$ can be tested exhaustively using a maximum of 2^{k_j} test patterns. If each cone is tested exhaustively, then the circuit under test (CUT) is said to be tested pseudo-exhaustively. Let T be the test sequence applied to the CUT. The length of test sequence required for pseudo-exhaustive

testing is bounded by

$$| T | \geq \max_{i=1}^{m} 2^{k_i} = 2^k. \tag{1}$$

Note that this is the minimum number of test patterns required to test the cone with the largest number of inputs (k) exhaustively. If the cones are tested one at a time, then

$$| T | = \sum_{i=1}^{m} 2^{k_i}. \tag{2}$$

There are many problems with testing one cone at a time. Firstly, the TPG may have to be reconfigured to test each cone. This increases the hardware complexity of the TPG as well as the complexity of BIST controller. Also, longer test sequence may be required than for testing all cones simultaneously.

A number of different approaches have been presented in the literature to design TPGs which test the CUT pseudo-exhaustively. A detailed review of these methods can be found in [ABF90].

2.1.1 Test Signals (Verification Testing)

In partial dependence circuits, two inputs x_i and x_j which do not belong to *any* cone together, are said to be compatible. During testing, compatible inputs may be connected to the same TPG output. Each group of compatible inputs is called a *test signal*. Minimum number of test signals can be obtained by using the procedure outlined in [McC84]. Once p test signals, $S_1, S_2, \ldots, S_p$, are obtained, a p-stage LFSR with primitive feedback polynomial may be used as TPG.

2.1.2 Linear Sums

Given an n input circuit, the input $x_1, x_2, \ldots, x_n$ are assigned to a minimum number of test signals $S_1, S_2, \ldots, S_p$, *and their linear sums* $S_1 \oplus S_2, S_1 \oplus S_3, \ldots, S_1 \oplus S_2 \oplus \cdots \oplus S_p$. This can be viewed as a more general form of grouping than in verification testing. Hence, the test

length can be further reduced [Ake85]. The basic idea is that if $\{x_{l_1}, x_{l_2}, \ldots, x_{l_{k_j}}\}$ belong to $cone_j$, then the signals (some may be linear combinations of S_i), applied to $x_{l_1}, x_{l_2}, \ldots, x_{l_{k_j}}$ should be linearly independent. This method guarantees that if all possible 2^p pattern are applied to the p test signals, then each cone is tested exhaustively. This can be easily achieved by connecting the outputs of a p-bit LFSR to the p test signals.

2.1.3 Cyclic Code

A test set T is a universal pseudo-exhaustive test set for (n, k) CUTs if and only if

1. It is a set of n-tuples (each row is an n-bit vector), and
2. Any k columns of T have all possible 2^k k-bit patterns occurring at least once.

It is clear that such test set applies exhaustive tests to all cones in any (n, k) CUT.

Let $g(x)$ be a polynomial over GF(2). Then $g(x)$ is said to be a generator of cyclic code of block length N, minimum distance D, and number of information symbols K if

- $g(x)$ has degree $N - K$
- $g(x) \mid x^N + 1$
- any non-zero multiple of $g(x)$ of degree $\geq N - K$ and $\leq N$ has at least D non zero coefficients.

It has been shown [LC83, WM88] that if code generated by $g(x)$ has a minimum distance D, then the code generated by $h(x) = \frac{x^N+1}{g(x)}$ generates a set of vectors in which any $D-1$ columns are linearly independent. (This means that any $D - 1$ columns of the patterns generated will have all possible 2^{D-1} distinct bit-patterns occurring at least once.) Hence, given an (n, k) CUT, the coding theory methods select $g(x)$ such that it generates an (N, K, D) code where,

$$N \geq n$$

$$D \geq k+1.$$

Then $h(x)$ is computed as

$$h(x) = \frac{x^N + 1}{g(x)}. \tag{3}$$

The equation

$$f(x) = P(x)h(x) \tag{4}$$

where $P(x)$ is any primitive polynomial of degree $N - K$, is used as LFSR feedback polynomial. The LFSR is initialized with any multiple of $h(x)$ as seed. The patterns generated by this LFSR form a pseudo-exhaustive test set for any (n, k) CUT.

2.2 Review of Two-Pattern Testing

Consider an n-input m-output (n, k) CUT, where $\{x_1, x_2, \ldots, x_n\}$ are the n inputs and $\{y_1, y_2, \ldots, y_m\}$ are the m outputs. The j-th output is

$$y_j = f_j(x_{l_1}, x_{l_2}, \ldots, x_{l_{k_j}})$$

assuming that the j-th output depends on k_j inputs. Two-pattern exhaustive testing of $cone_j$ requires that all possible V_1 vectors (2^{k_j} distinct ones) be applied. Also, each V_1 should be followed by all possible V_2 vectors ($2^{k_j} - 1$ distinct ones, $V_2 \neq V_1$). Hence, the number of tests required to test $cone_j$ exhaustively with all pattern pairs is

$$|T| = 2^{k_j}(2^{k_j} - 1). \tag{5}$$

The lower bound for the (n, k) CUT (where $k = \max_{i=1}^{m} k_i$) is $|T| \geq 2^k(2^k - 1)$. Also, if one cone is tested at a time, then

$$|T| = \sum_{i=1}^{m} 2^{k_i}(2^{k_i} - 1).$$

As has been noted above, the test sequence length required to test one cone at a time may be large. Also, the TPG would have to be reconfigurable, and control complexity

may be high. Hence, in the following, the emphasis is on testing cones together in one test session.

In [Sta84], a $2n$-stage shift-register is used as the basic test pattern generator for stuck-open faults. Deterministic two-pattern tests are generated for the stuck-open faults in the circuit under test. The feedback of the shift-register is designed to ensure that all the deterministically generated two-pattern tests are embedded in the sequence generated by the resulting test pattern generator. The complexity of the TPG design is high — both in terms of design effort and hardware requirements.

In [CK85], a test pattern generator design which combines verification testing and adjacency testing concepts, is presented. In adjacency testing, instead of applying all (V_1, V_2) pairs, all (V_1, V_2) pairs *which differ in 1-bit* are applied. The test time required for this technique is lower than in two pattern pseudo-exhaustive testing. However, TPG hardware complexity is high due to the special non-linear test pattern generators required for adjacency testing. Furthermore, the test patterns generated do not guarantee coverage of all two pattern testable faults. (However, for some special circuits it guarantees coverage of all path delay faults. For example, if all paths in a CUT are robustly testable, adjacency testing guarantees 100% path delay fault coverage.)

Recently, a number of results have been reported on this topic. In [FM91] a metric called transition count is defined to measure vector pair coverage. Also, an analytical technique to compute the transition count for a given linear test pattern generator is presented. A similar measure, called AC strength, is presented in [SB91]. This paper and its other results are discussed in greater detail in the following section. In [ZBM92], LFSRs and linear CAs, with permuted outputs, have been analyzed for transition coverage. Also, in a companion paper [CG93], the authors have identified necessary and sufficient conditions for obtaining complete transition coverage for any LFSR or CA (with Rules 90 and 150 and null boundary conditions). Further, a procedure which identifies a large class of LFSR/CA TPGs which maximize transition coverage has been presented.

2.2.1 Input Separation

In [SB91] a metric called AC strength is defined. AC strength is the fraction of all possible vector pairs that may be applied to the CUT using a particular scan chain/TPG configuration.

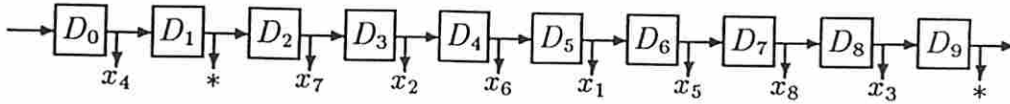
Consider an (n, k) CUT. If two inputs of the CUT, which belong to the same cone, are connected to two consecutive stages of a scan chain, then some pattern pairs can never be applied to the cone(s) which depend on both inputs. This is due to the fact that V_2 vector is obtained by one shift of vector V_1 in the chain. Hence, to guarantee a high coverage of delay and stuck-open faults in each cone, the inputs belonging to the same cone should be separated by at least one flip-flop in the scan chain.

This notion, called *input separation* has been formalized. A procedure to assign the n CUT inputs to the stages of a ν-bit shift-register, such that no two inputs belonging to any cone are assigned to adjacent shift-register stages, has been presented. For some circuits, the input separation cannot be achieve without adding extra (dummy) stages to the register (i.e. $\nu > n$). In such cases the procedure minimizes ν.

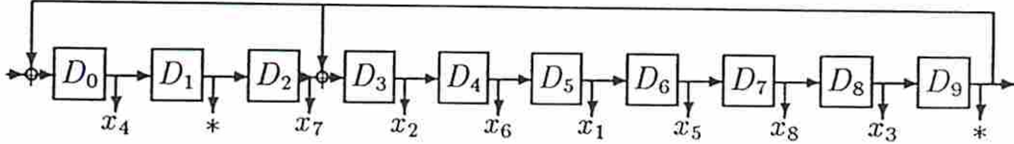
The notion of input separation can be easily applied to BIST. The following construction utilizes the input separation notion to design a BIST TPG.

Construction 1 *Given a (n, k) CUT.*

1. *Assign n CUT inputs to a ν stage shift-register stages such that*
   - *No two inputs which belong together in any cone are connected to consecutive shift-register stages, and*
   - *ν is minimized.*
2. *If the inputs of CUT assigned to the first and last stages of the shift-register are incompatible, then add an extra dummy stage at the end of the shift-register.*
3. *Convert the shift-register to an LFSR by addition of feedback connections and exclusive-or gates. Note that, primitive feedback polynomial may be used for the LFSR.*



(a) Shift Register for Input Separation



(b) TPG Using Input Separation

Figure 1: TPG Designed Using Construction 1 (Example 1)

Example 1 *Consider an 8-input 5-output function [Ake85] defined by*

$$
\begin{aligned}
y_1 &= f_1(x_1, x_2, x_3, x_4) \\
y_2 &= f_2(x_6, x_7, x_8) \\
y_3 &= f_3(x_3, x_4, x_5, x_6) \\
y_4 &= f_4(x_3, x_5, x_6, x_7) \\
y_5 &= f_5(x_1, x_4, x_7, x_8)
\end{aligned}
$$

For this CUT, no ordering with $\nu = n$ stages satisfies the input separation conditions. Only one ordering of n-inputs with one dummy stage is possible as shown in Figure 1 (a). (Dummy stages have the '' marks at their outputs.) Since, x_3 and x_4 (assigned to first and the last stages of the SR) belong to* cone_1 *(and* cone_3*), an additional dummy stage is added to the end of the shift-register. This 10-stage shift-register is converted to an LFSR by using any degree 10-primitive feedback polynomial. One such TPG is shown in Figure 1 (b). The LFSR can be initialized to any non-zero state. The TPG will apply two pattern pseudo-exhaustive tests to the CUT and the test sequence length is $2^{10} - 1$.*

If the scan chains are ordered to ensure input separation, all two-pattern tests can be applied to any cone. However, there are some considerations special to BIST.

1. In scan, deterministically generated test patterns are applied to the CUT. However, in BIST, LFSR/CA generated pseudo-random patterns are used. Hence, the test length for BIST designs may be large. Reduction of test length is a major consideration for BIST TPGs.

2. In scan, the scan chain is used to apply test patterns as well as to observe CUT response for each test vector. This constrains the scan registers to form (one or more) chain(s). However, in most BIST designs, TPG is not used simultaneously to generate patterns and observe the circuit response. This provides the flexibility of connecting the flip-flops in other configurations.

3. It can be shown that input separation is sufficient for LFSR/CA test pattern generators to apply all pattern pairs to each circuit cone. However, two inputs to a cone may be connected to consecutive CA stages and still guarantee application of all pattern pairs to the cone.

Due to these differences, more efficient TPGs (lower test length) for two-pattern pseudo-exhaustive testing can be designed and are discussed in the following.

3 Two-Pattern Pseudo-Exhaustive Testing

3.1 Basics

As has been shown above, pseudo-exhaustive testing has been well studied in the literature for single pattern testing (i.e. for testing all combinational faults in combinational circuits). The focus of this paper is on development of pseudo-exhaustive TPG techniques for two-pattern testing. In doing so, the emphasis is to take full advantage of all known results in the single pattern pseudo-exhaustive testing. The TPG design problem is defined as:
Given an (n, k) CUT, design a TPG such that

- All possible vector pairs are applied to any cone, and

- Test sequence length and TPG hardware complexity are minimized.

3.2 Test Signals

In this section it is shown that the test signals (verification testing) approach discussed above (for single pattern testing) can be easily extended to two pattern testing.

Lemma 1 *Given an (n, k) CUT and its cone information. Let p test signals be determined using method proposed in [McC84]. If all $2^p(2^p - 1)$ vector pairs are applied to the p test signals, then the CUT is two-pattern pseudo-exhaustively tested.*

Proof: Let $V_1, \ldots, V_L$ be the test pattern applied to the test signals. For a cone of size k, when these are applied to the p test signals, then by construction 2^k distinct pattern $O_1, \ldots, O_{2^k}$ appear at the inputs of $cone_j$. Note that in this case, $L \geq 2^k$. Let $V_{i_1}, V_{i_2}, \ldots, V_{i_{2^k}}$ be the 2^k patterns (where $0 \leq i_1, i_2, \ldots, i_{2^k} \leq L$) be the 2^k patterns that lead to the application of $O_1, O_2, \ldots, O_{2^k}$ to the cone being considered. If the TPG is redesigned to generate all possible pattern pairs, then there will be pairs of vectors such that each pairs (V_j, V_k), $0 \leq j, k \leq L$ will occur. If $V_j = V_{i_r}$ and $V_k = V_{i_s}$, then the application of this pair at test signals implies that (O_r, O_s) pair is applied. Since all pairs (V_j, V_k), $0 \leq j, k \leq L$ are applied, all pairs at test signals (O_r, O_s), $0 \leq r, s \leq 2^k$ are applied at the cone. Hence, the CUT is two-pattern pseudo-exhaustively tested. **Q.E.D.**

The above Lemma suggests the following construction for design of two-pattern pseudo-exhaustive test pattern generator.

Construction 2 *Given (n, k) CUT,*

1. *Determine p test signals [McC84].*
2. *Use $2p$-stage LFSR/CA (maximal length) and connect p out of its $2p$ outputs (as outlined in [CG93]) to the p-test signals.*

Note that in [CG93], it has been shown that for complete transition coverage, the p test signals (or primary inputs) can be connected to all odd (or even) stages of the $2p$-bit LFSR. (All possible ways of selecting p out of $2p$ TPG outputs to the test signals are discussed in [CG93].) If a cellular automata is used, then the p test signals should be connected to the $2p$ stages of the CA such that no two test signals are connected to any consecutive (odd, even) pair of stages.

Example 2 *Consider the (8,4) CUT in Example 1. Using the algorithm in [McC84],* $p = 5$ *tests signals are sufficient to test this CUT.*

$$\begin{aligned} S_1 &= \{x_1, x_5\} \\ S_2 &= \{x_2, x_6\} \\ S_3 &= \{x_3, x_8\} \\ S_4 &= \{x_4\} \\ S_5 &= \{x_7\} \end{aligned}$$

The number of patterns required for single pattern pseudo-exhaustive testing is $2^5 - 1 = 31$ *using 5-stage LFSR/CA with primitive feedback polynomial. Instead, if 10-stage LFSR is used and the 5 circuit inputs are connected to all the odd (even) stages of the LFSR, then* $2^{10} - 1 = 1023$ *patterns (which are a superset of the* $2^5(2^5 - 1) = 992$ *exhaustive two patterns) are applied to the CUT and the circuit is tested two-pattern pseudo-exhaustively. In this case, the test sequence length is the same as in Example 1 which uses Construction 1. Figure 2 shows a TPG design. Note that* x_1 *and* x_5 *are both connected to flip-flops which have the same input. Only one of these flip-flop's output is connected to the next stage input. Hence, in a scan design, the flip-flops will have to be reconfigured into a chain if the CUT response captured at the stage feeding* x_5 *is to be observed at the scan-out pin. A 10-stage CA may be used as well.*

It can be shown that for a large class of CUTs, Construction 2 using the notion of test signals, will provide TPG designs which will ensure that the CUT can be tested pseudo-exhaustively with fewer (or as many) tests than the TPGs designed using Construction 1.

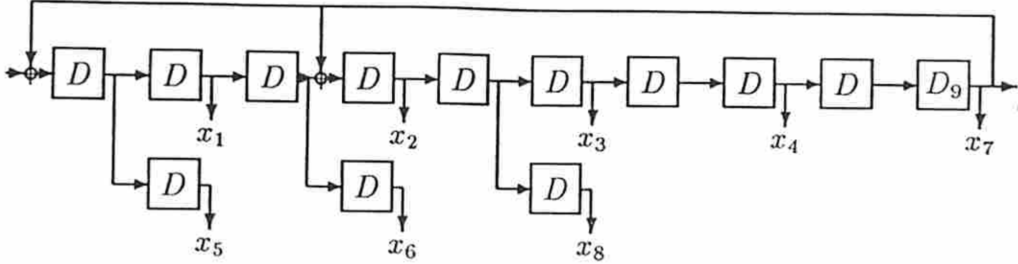


Figure 2: TPG Designed Using Construction 2 (Example 2)

The following theorem is presented without proof. (The proof of this theorem is involved and lengthy.)

Theorem 1 *Given an (n, k) CUT. Let the number of stages in the TPG1, designed using Construction 1, be ν. If*

1. *$\nu = n$ and n is even, or*

2. *$\nu > n$,*

then TPG2, obtained by using Construction 2, is guaranteed to apply two pattern pseudo-exhaustive tests to the CUT using fewer (or as many) tests than TPG1.

Hence, the concept of test signals yields TPG designs which generate shorter test sequences. For most (n, k) CUTs, the reduction in test length can be large, as illustrated by the experimental results (Section 4). This reduction in test length, for pseudo-exhaustive two-pattern testing, may be at the expense of increased TPG hardware complexity. However, small increase in TPG complexity may be acceptable in applications which require extremely high fault coverage, for substantial reduction in test sequence length.

3.3 Linear Sums

As has been discussed in Section 2.1.2, linear sum method provides a more general and hence more efficient TPG design. Hence, the next step is to study methods for two-pattern PET

using the linear sum method.

Lemma 2 *Given an (n, k) CUT. Let $S_1, S_2, \ldots, S_p$ be the p test signals determined by the linear sum method. If all possible pattern pairs are applied to the p test signals, then all cones of the CUT are two-pattern pseudo-exhaustively tested.*

Proof: Let m be the number of outputs and k_j, $1 \leq j \leq m$, be the number of inputs that feed $cone_j$. In linear sum method, the test signals and their linear sums are assigned to inputs the CUT such that no subset of these test signals, assigned to inputs to any cone, sums to zero. As the TPG generates 2^p test patterns, each of the 2^{k_j} patterns are applied to the $cone_j$ (at least once).

For two-pattern testing, every possible pattern pair (V_1, V_2) appears at the p test signals. Since no subset of inputs of each cone is dependent, exhaustive 2^{k_j} patterns for $cone_j$ occur when all 2^p V_1 patterns occur. For each V_1, there are 2^p corresponding V_2's. Hence, $cone_j$ must have exhaustive 2^{k_j} V_2 patterns for each V_1 (except for all zero V_1 which may only have $2^{k_j} - 1$ V_2, excluding all zero V_2). The number of pattern pairs appeared at inputs of $cone_j$ is $\geq 4^{k_j} - 1$, and is an exhaustive two-patterns test set for $cone_j$. **Q.E.D.**

The above result leads to the following construction:

Construction 3 *Given an (n, k) CUT,*

1. *Use linear sum method to determining $S_1, \ldots, S_p$ and their linear sums and assign them the CUT inputs [Ake85].*
2. *Use a $2p$-stage LFSR/CA and connect the p test signals to the TPG as shown in [CG93].*

Example 3 *For the circuit in Example 2, the 8 circuit inputs can be assigned to 4 test signals and their linear sums, as [Ake85]*

$$x_1 = x_5 = S_1$$

$$x_2 = x_6 = S_2$$

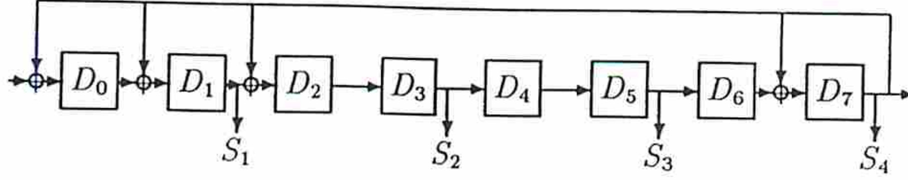


Figure 3: TPG Designed Using Construction 3 (Example 3)

$$\begin{aligned} x_3 = x_8 &= S_3 \\ x_4 &= S_4 \\ x_7 &= S_2 \oplus S_4 \end{aligned}$$

An 8-stage LFSR can be used to test the circuit two-pattern pseudo-exhaustively. The 4 test signals are connected to all the even (odd) stage outputs. Any degree-8 feedback polynomial and non zero seed may be used. Figure 3 shows the TPG. Note that the test length is $2^8 - 1$ *which is less than in Examples 1 and 2.*

Construction 3 uses the concept of linear sums to reduce the number of test signals required to test a CUT pseudo-exhaustively. Since, linear sums is a generalization of the test signal concept, the number of test patterns required by a TPG designed by the above Construction is less than (or equal to) that for the TPG designed using Construction 2. Furthermore, by Theorem 1, a TPG designed using the above construction will generate shorter test sequences than those required by the TPGs designed using Construction 1. Hence, the linear sum method can be used to design TPGs which reduce test length for two-pattern pseudo-exhaustive testing over those designed using the input separation approach.

3.4 Cyclic Codes

The test signals and linear sum methods discussed above are efficient in terms of test time since they exploit not only the parameters (n, k) of the CUT but also the exact dependencies of various cones in the CUT. However, the TPG circuitry may be large. The cyclic code method is useful because:

1. It replaces the complex optimization problems required in determining test signals by look up of the table of known codes.
2. Produces relatively area-efficient TPG design.
3. Produces universal TPGs suitable for any (n, k) CUT.

The extension of the cyclic code from single pattern testing to two-pattern testing is not as straightforward as in the cases of test signal and linear sum methods. In the following, new concepts in TPG design using cyclic codes have been developed. These new ideas have provided a procedure for design of TPGs which have been proven to require shorter test length than the previously known scheme.

3.4.1 Cyclic Codes in GF(4)

In two-pattern testing, exhaustive coverage of pattern-pairs is required. Each pattern-pair can be viewed as a pair of bits applied to each input of CUT. Two bits can take four values $(00, 01, 10, 11)$ and may be viewed as symbols over GF(4). Hence, for two-pattern pseudo-exhaustive testing, it is necessary and sufficient to apply all the 4^k possible symbol combinations at any k inputs of a CUT. A two-pattern pseudo-exhaustive test set for an (n, k) CUT is defined as

1. A set of n tuples with symbols in GF(4)
2. Such that any k columns have all the 4^k symbol combinations.

Under this framework, the TPG design problem based on the theory of cyclic codes over GF(4) is:

1. Design an n-stage GLFSR [PG91] with primitive feedback polynomial $f(x)$ over GF(4), such that test length is minimized, and
2. Find an initial seed over GF(4).

Procedures for constructing the TPG is given by

Construction 4 *Given an (n, k) CUT,*

1. *Find a (N, K, D) cyclic code over GF(4) where*
   (a) *$N \geq n$ is the code length*
   (b) *$D \geq k + 1$ is the minimum distance over GF(4)*
2. *Compute $h(x) = \frac{x^N+1}{g(x)}$*
3. *Find $P(x)$, a primitive polynomial of degree $N - deg(h(x))$ over GF(4)*
4. *Let $f(x) = P(x)h(x)$ be the feedback polynomial over GF(4)*
5. *Let any multiple of $h(x)$ be the seed*

Theorem 2 *The TPG designed using Construction 4 generates two- pattern pseudo-exhaustive tests for (n, k) CUT.*

The above construction is a direct extension of the cyclic code method from single-pattern pseudo-exhaustive testing to two-pattern pseudo-exhaustive testing. However, the major disadvantage is that it requires a table of codes over GF(4). Though GF(4) codes are a natural extensions of binary codes, tables of GF(4) codes and lists of minimal and primitive polynomials over GF(4) are not available. So, even though this may be a direct method, it is practically infeasible. Hence, the proof of theorem has been omitted. In the following, two new approaches using tables for GF(2) codes are presented.

3.4.2 Design of Two-Pattern PET TPG Over GF(2)

In this case, the TPG design problem can be stated as:

1. Design a $2n$-stage LFSR with feedback polynomial $f(x)$ over GF(2), such that all $4^k - 1$ vector pairs are applied to all possible cones of size k, in minimum number of clock cycles, and

2. Find an initial seed.

In the following, two new constructions shall be discussed which achieve these objectives.

In the following constructions, assume that a $2n$-stage LFSR is used and only the even stage outputs of the LFSR are connected to the CUT. (The CUT inputs may be connected to the odd stages of the LFSR as well. The following construction is still applicable.) It is required that any k inputs to the CUT be independent. Also, since complete two-pattern coverage is required, it is necessary that even stages $2i$, which are connected to CUT inputs, are independent of the set of odd stages adjacent to them. Hence, for two-pattern pseudo-exhaustive testing, it is sufficient to design a $2n$-stage LFSR such that any $2k$ columns of binary pattern generated are independent. This leads to the following construction:

Construction 5 *Given an (n, k) CUT,*

1. *Find a (N, K, D) binary cyclic code where*

   (a) *$N \geq 2n$ is the code length*

   (b) *$D \geq 2k + 1$ is the minimum distance*

2. *Compute $h(x) = \frac{x^N+1}{g(x)}$*

3. *Find $P(x)$, any primitive polynomial of degree $N - deg(h(x))$ over GF(2).*

4. *Let $f(x) = P(x)h(x)$ be the feedback polynomial over GF(2)*

5. *Let any multiple of $h(x)$ be the seed*

6. *Connect the n CUT inputs to the odd (or even) stages of the LFSR*

Theorem 3 *The TPG designed using Construction 5 applies two-pattern pseudo-exhaustive tests to any (n,k) CUT.*

Proof: The TPG designed for an (n,k) CUT using above construction has at least $2n$ stages and generates the codewords of the dual code of that generated by $g(x)$ [WM88]. This implies that any $2k$ stage outputs of the TPG (since the minimum distance of $g(x)$ is $2k+1$) generate all possible $2^{2k}-1$ patterns. Hence, $2^{2k}-1$ vector pairs are applied to any k even (odd) stages connected to the CUT. This implies that all cones in the CUT of size $\leq k$ are tested two-pattern exhaustively. **Q.E.D**

Example 4 *Given a $(6,3)$ CUT,*

1. *Need a code with*
   (a) *Code length* $= N \geq 2n = 12$
   (b) *Minimum distance* $= D \geq 2k+1 = 7$
2. *Code $C(15,5,7)$ has generator* $g(x) = (1+x+x^2)(1+x+x^2+x^3+x^4)(1+x+x^4)$
3. $h(x) = (x^{15}+1)/g(x) = (1+x)(1+x^3+x^4)$
4. $P(x) = 1+x^3+x^{10}$
5. $f(x) = h(x)P(x) = 1+x+x^4+x^5+x^6+x^8+x^{10}+x^{11}+x^{13}+x^{15}$
6. *Test length* $= 2^{10}-1$

This construction provides the design of a $2n$ stage LFSR which generates two-pattern pseudo-exhaustive tests of any (n,k) CUT. As shall be seen below, this procedure guarantees that the sequences generated at any $2k$ LFSR stages are independent. However, the only real requirement is that the sequences generated at any k even stages and at the k odd stages (left neighbors of the k even stages selected) be independent. TPGs designed by using above construction may require more test patterns than may be necessary. The following

construction for design of TPG accounts for the above observations. It designs a TPG such that sequences generated at

1. All odd and even stage outputs are totally independent.

2. Any k odd (or even) stage outputs are independent of each other.

Note that the above requirements are particular to the two pattern testing problem at hand. No known TPG design methodology exists which can meet these goals in an efficient way. The following construction provides a new TPG design methodology developed to exploit the special characteristics of two pattern testing.

Construction 6 *Given an (n, k) CUT,*

1. *Find a (N, K, D) binary cyclic code where*

   (a) *$N \geq n$ is the code length*

   (b) *$D \geq k + 1$ is the minimum distance*

2. *Compute $h(x) = (x^N + 1)/g(x)$*

3. *Find $P(x)$, a primitive polynomial of degree $2N - 2deg(h(x))$ over GF(2)*

4. *Let $f(x) = P(x)(h(x))^2$ be the feedback polynomial over GF(2)*

5. *Let any multiple of $(h(x))^2$ be the seed*

6. *Connect the n CUT inputs to the odd (or even) stages of the LFSR*

Two Lemmas, essential to the proof of the following theorem, are outlined first.

Lemma 3 *If $g(x)$ is a generator polynomial of an (n, k) cyclic code, then $g^2(x)$ is a generator polynomial of a $(2n, 2k)$ cyclic code.*

Proof: Since $g(x)$ is the generator of (n, k) code, the degree of $g(x)$ is $n-k$ and $g(x)$ divides $x^n + 1$ [LC83]. Hence, the degree of $g^2(x)$ is $2(n-k)$. Also, it can be shown that $g^2(x)$ divides $x^{2n} + 1 (= (x^n+1)^2)$. Hence, $g(x)$ generates a $(2n, 2k)$ cyclic code. **Q.E.D.**

Lemma 4 *If the code generated by* $g(x) = g_0 x + g_1 x + \cdots + g_{n-k} x$ *has minimum distance* D*, then the code generated* $g^2(x)$ *has the same minimum distance.*

Proof: The generator matrix G of the (n, k) cyclic code generated by $g(x)$ is given by

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 & 0 \\ 0 & 0 & g_0 & \cdots & g_{n-k-2} & g_{n-k-1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k-1} & g_{n-k} \end{pmatrix} \tag{6}$$

and the generator matrix G_2 of the $(2n, 2k)$ cyclic code generated by $g^2(x)$ is given by

$$G_2 = \begin{pmatrix} g_0 & 0 & g_1 & 0 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & g_0 & 0 & g_1 & 0 & \cdots & 0 & g_{n-k} & \cdots & 0 & 0 \\ 0 & 0 & g_0 & 0 & g_1 & \cdots & g_{n-k-1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & g_{n-k} \end{pmatrix} \tag{7}$$

Let r_i, $0 \le i \le 2n-1$, represent row vector of G_2. Then the code word v, where $v \in$ row space of G_2, is given by

$$v = v_0 r_0 + v_1 r_1 + v_2 r_2 + \cdots + v_{2n-1} r_{2n-1} \tag{8}$$

$$= \underbrace{v_0 r_0 + v_2 r_2 + \cdots + v_{2n-2} r_{2n-2}}_{d_{\text{even}}=D} + \underbrace{v_1 r_1 + v_3 r_3 + \cdots + v_{2n-1} r_{2n-1}}_{d_{\text{odd}}=D} \tag{9}$$

where $v_i \in \{0, 1\}$, $0 \le i \le 2n-1$, is the component with respects to r_i, and $d_{\text{even}}(d_{\text{odd}})$ is the minimum distance of the even (odd) subspace of G_2. Since even and odd vectors of G_2 span disjoint subspace of G_2, the minimum distance of the row space of G_2 is determined by the minimum of d_{even} and d_{odd}, which is D. **Q.E.D.**

Theorem 4 *The LFSR designed using Construction 6 applies two-pattern pseudo-exhaustive tests to any (n,k) CUT.*

Proof: Let $h(x) = h_0 + h_1 x + h_2 x^2 + \cdots + h_k x^k$. Then the generator matrix H_2 of generator polynomial $h^2(x)$ is given by

$$H_2 = \begin{pmatrix} h_0 & 0 & h_1 & 0 & h_2 & \cdots & h_k & 0 & \cdots & 0 & 0 \\ 0 & h_0 & 0 & h_1 & 0 & \cdots & 0 & h_k & \cdots & 0 & 0 \\ 0 & 0 & h_0 & 0 & h_1 & \cdots & h_{k-1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & h_k & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & h_k \end{pmatrix} \quad (10)$$

Since the minimum distance of the code generated by $g(x)$ has the value D, any $D-1$ or less columns of H must be linearly independent. By Lemma 3, $h^2(x)$ generates a $(2N, 2N-2K)$ cyclic code and by Lemma 4, any $D-1$ or less columns of H_2 are also linearly independent. Let R^w denote the column space spanned by any $D-1$ columns. Particularly, if we choose any $D-1$ odd columns or $D-1$ even column, then each of them will span R^w. As shown in Lemma 4 and Eq. (10), odd and even column vectors of H_2 spans disjoint column subspace of H_2. If $D-1$ odd columns and $D-1$ even columns are chosen, then the resulting $2(D-1)$ columns span R^{2w} space. Therefore, the CUT with inputs connected to the taps of $(D-1)$ odd or even stages is tested two-pattern pseudo-exhaustively. **Q.E.D.**

Example 5 *Given a $(6,3)$ CUT,*

1. *Need a code with*
   (a) *Code length $= N \geq n = 6$*
   (b) *Minimum distance $= D \geq k+1 = 4$*
2. *Code $C(7,3,4)$ has generator $g(x) = (1+x^2+x^3)(1+x)$*
3. *$h(x) = (x^7+1)/g(x) = 1 + x^2 + x^3$*

4. $P(x) = 1 + x + x^2 + x^7 + x^8$

5. $f(x) = h^2(x)P(x) = 1 + x + x^2 + x^4 + x^5 + x^{11} + x^{12} + x^{13} + x^{14}$

6. *Test length* $= 2^8 - 1$

This construction does not overspecify the two-pattern pseudo-exhaustively testing problem. Hence, the length of the test sequence generate by a TPG designed using this method is lower than for the previous case. In Table 1, details of TPGs designed using Construction 5 and Construction 6 have been shown for five circuits. For each circuit, row marked $C5$ shows the details of TPGs designed using Construction 5 and the row labeled $C6$ corresponds to Construction 6. These results clearly illustrate how the TPGs designed using Construction 6 require shorter test sequence than those designed using Construction 5. Also, note that in the Constructions 5 and 6, the number of stages in LFSR can be greater than $2n$. However, it can be shown that this can be reduced to a $2n$ bit LFSR in all cases.

Note that, *no cone specific information has been used*, hence the test times required are large. These TPGs are universal TPGs for the corresponding (n, k) circuit, for two pattern pseudo-exhaustive testing. In the following, a method of utilizing circuit specific information to design TPGs which generate shorter two-pattern pseudo-exhaustive test sequences shall be studied. Since, the test length for TPG designed using Construction 6 can be shown to be always lower than that for TPG designed using Construction 5, in the following discussion only the former construction shall be used along with the circuit specific test signal approach.

Note that the test signal and coding theory concepts may be combined to obtain shorter test sequences. The original circuit inputs may be combined into test signals. TPG can then be designed for the resulting circuit using Construction 6 above. Note that in some cases, this may lead to the same TPG design as given by using Construction 2. However, in general, this will reduce test length for pseudo-exhaustive testing. The following construction combines the test signal and coding theory approaches.

Construction 7 *Given* (n, k) *CUT*

Table 1: Comparison of TPGs Designed Using Constructions 5 and 6

I. (4, 2) circuit

	N	K	D	$g(x)$ or $g^2(x)$	$h(x)$ or $h^2(x)$	$P(x)$	Test length	$f(x)$
$C5.$	8	1	8	$\sum_{i=0}^{7} x_i$	$1+x$	$1+x^3+x^7$	2^7	$1+x+x^3+x^4+x^7+x^8$
$C6.$	4	1	4	$(\sum_{i=0}^{3} x_i)^2$	$(1+x)^2$	$1+x+x^6$	2^6	$1+x+x^2+x^3+x^6+x^8$

II. (8, 4) circuit

	N	K	D	$g(x)$ or $g^2(x)$	$h(x)$ or $h^2(x)$	$P(x)$	Test length	$f(x)$
$C5.$	16	1	16	$\sum_{i=0}^{15} x_i$	$1+x$	$1+x+x^{15}$	2^{15}	$1+x^2+x^{15}+x^{16}$
$C6.$	8	1	8	$(\sum_{i=0}^{7} x_i)^2$	$(1+x)^2$	$1+x^2+x^3+x^{13}+x^{14}$	2^{14}	$1+x^3+x^4+x^5+x^{13}+x^{14}+x^{15}+x^{16}$

III. (5, 3) circuit

	N	K	D	$g(x)$ or $g^2(x)$	$h(x)$ or $h^2(x)$	$P(x)$	Test length	$f(x)$
$C5.$	10	1	10	$\sum_{i=0}^{9} x_i$	$1+x$	$1+x^4+x^9$	2^9	$1+x+x^4+x^5+x^9+x^{10}$
$C6.$	5	1	5	$(\sum_{i=0}^{4} x_i)^2$	$(1+x)^2$	$1+x+x^2+x^7+x^8$	2^8	$1+x+x^3+x^4+x^7+x^8+x^9+x^{10}$

IV. (7, 4) circuit

	N	K	D	$g(x)$ or $g^2(x)$	$h(x)$ or $h^2(x)$	$P(x)$	Test length	$f(x)$
$C5.$	14	1	14	$\sum_{i=0}^{13} x_i$	$1+x$	$1+x+x^2+x^{12}+x^{13}$	2^{13}	$1+x^3+x^{12}+x^{14}$
$C6.$	7	1	7	$(\sum_{i=0}^{6} x_i)^2$	$(1+x)^2$	$1+x+x^5+x^8+x^{12}$	2^{12}	$1+x+x^2+x^3+x^5+x^7+x^8+x^{10}+x^{12}+x^{14}$

V. (6, 3) circuit

	N	K	D	$g(x)$ or $g^2(x)$	$h(x)$ or $h^2(x)$	$P(x)$	Test length	$f(x)$
$C5.$	15	5	7	$(1+x+x^2)(1+x+x^2+x^3+x^4)(1+x+x^4)$	$(1+x)(1+x^3+x^4)$	$1+x^3+x^{10}$	2^{10}	$P(x)h(x)$
$C6.$	7	3	4	$[(1+x)(1+x+x^3)]^2$	$(1+x^2+x^3)^2$	$1+x+x^2+x^7+x^8$	2^8	$P(x)h^2(x)$

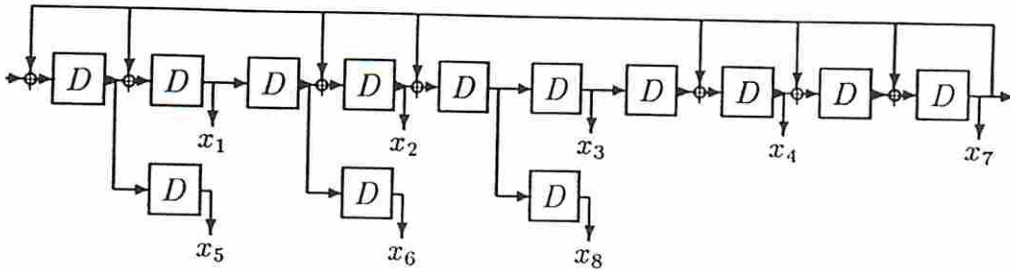


Figure 4: TPG Designed Using Construction 7 (Example 6)

1. *Combine n CUT inputs into p tests signals. This will yield a* (p, q) *circuit (where* $q = k$ *is the maximum number of test signals any circuit output depends on).*

2. *Use Construction 6 to design TPG for the* (p, q) *circuit.*

Example 6 *Consider the* $(8, 4)$ *circuit used in Examples 1, 2, and 3. The test signal approach (see Example 2) combines the inputs of the CUT into* $p = 5$ *test signals. The resulting circuit can be viewed as a* $(5, 4)$ *circuit in the Step 2 of the Construction 7. This yields a 10-stage LFSR with feedback polynomial (see Figure 4)*

$$f(x) = 1 + x + x^3 + x^4 + x^7 + x^8 + x^9 + x^{10}$$

and a test sequence length of $2^8 - 1$. *Note that this is the shortest test sequence that can be generated to test an (8,4) CUT two pattern pseudo-exhaustively.*

TPG designed using the above construction can be shown to generate shorter test sequence than TPG designed using Construction 2. The following Theorem states the result.

Theorem 5 *If* $p > q$ *in Construction 7, then TPG designed using Construction 7 will always generate a test sequence of length no more than a quarter of the length of test sequence generated by any TPG designed using Construction 2.*

Proof: If $p > q$, then a cyclic code with code length $N = p$, and minimum distance $D = p$, can be always found. Note that since $p > q$, this implies that this code can be used to design a $2p$ stage LFSR to generate test sequences which guarantee application of all vector pairs to any q out of p test signals. The generator polynomial of such a code is given by $g(x) = 1 + x + \cdots + x^{p-1}$. Hence, $h(x) = 1 + x$. This means that the feedback polynomial of the $2p$ stage LFSR is given by

$$f(x) = (1 + x)^2 P_{p-2}(x)$$

where $P_{p-2}(x)$ is any degree $p - 2$ primitive polynomial. This LFSR, with the p test signal connected to all odd (or all even) stages, generates a test sequence of length $2^{2p-2} - 1$. This is approximately $1/4$ of the length of test sequence generated if the TPG was designed using Construction 2 alone (test length in that case is $2^{2p} - 1$). **Q.E.D.**

Note that, if $p = q$ in Construction 7, then the test sequence length generated by the TPG designed using Construction 2 is *already minimum achievable.* Also note that, if $p > q + 1$, then even larger reduction in test sequence length may be possible. Using the above Theorem and Theorem 1, the following corollary can be proven.

Corollary 1 *TPG designed using the proposed Construction 7 will always generate a two pattern pseudo-exhaustive test sequence for a circuit which will, either be shorter than a test sequence generate by Construction 1, or be of minimal length.*

The details of the TPGs designed and the lengths of test sequences generated by them are discussed in the following.

4 Experimental Results

Test pattern generators were designed for a number of example circuits studied in earlier (single pattern) pseudo-exhaustive testing publications [Ake85, WM88]. Table 2 shows the (n, k) parameters of these circuit. It also shows the cone dependencies and the test signals required for the test signal (verification testing) and the linear sum approaches.

TPGs are designed for these circuit using Constructions 1-3, and Construction 7. All these constructions, unlike Constructions 5 and 6, exploit the information about the circuit cones to design more efficient TPGs. These are called $C1, C2, C3$, and $C7$, respectively in Table 3. In Table 3, N denotes the number of stages in the LFSR, and $P_i(x)$ indicates any degree-i primitive polynomial. The tap selection for the various schemes is given. The assignment of the CUT inputs to the LFSR stages is shown explicitly for $C1$ (input separation). For the rest of the methods, O/E indicates that all odd (or all even) taps should be connected to the test signals for the corresponding schemes (see Table 2). Note that the test time for all other constructions is less than or equal to the test time required by the input separation approach. It can be seen that, in all the cases the test length for C_2 is less than or equal to the test length for $C1$. Also, the test length for $C7$ is almost one-quarter the test length for $C2$, as suggested by Theorem 5. Note that for the (7,4) CUT (circuit number IV), the test length reduction for $C7$ over $C2$ is even more marked. Also, for all circuits, test length is considerably lower for $C7$ than for $C1$.

5 Conclusion

Two pattern pseudo-exhaustive testing guarantees detection of all one and two pattern testable faults. This includes a large class of faults such as single and multiple stuck-at faults, delay faults, and CMOS stuck-open faults. The main issues in pseudo-exhaustive testing are reduction of test sequence length and TPG hardware complexity.

A number of techniques for design of efficient TPGs for two pattern pseudo-exhaustive testing have been developed. It has been shown that the techniques described here will generate shorter test sequences than previously known techniques. A number of techniques have been obtained by combining the results in transition coverage [FM91], tap selection [CG93], and single pattern pseudo-exhaustive testing [Ake85, McC84]. Also, some principles unique to the two pattern testing problem have been identified and exploited to develop new techniques (Constructions 6 and 7) for the design of efficient TPGs. TPGs designed by using new constructions have been proven to require shorter test sequences for two pattern pseudo-

Table 2: Details of Circuit Dependencies

(n,k)CUT	Cone dependency	Test signal	Linear sum
$(4,2)$	$y_1=(x_1,x_3)$ $y_2=(x_1,x_2)$ $y_3=(x_2,x_3)$ $y_4=(x_3,x_4)$	$S_1=(x_1,x_4)$ $S_2=x_2$ $S_3=x_3$	$S_1=(x_1,x_4)$ $S_2=x_2$ $S_1\oplus S_2=x_1$
$(8,4)$	$y_1=(x_1,x_2,x_3,x_4)$ $y_2=(x_6,x_7,x_8)$ $y_3=(x_3,x_4,x_5,x_6)$ $y_4=(x_3,x_5,x_6,x_7)$ $y_5=(x_1,x_4,x_7,x_8)$	$S_1=(x_1,x_5)$ $S_2=(x_2,x_6)$ $S_3=(x_3,x_8)$ $S_4=x_4$ $S_5=x_7$	$S_1=(x_1,x_5)$ $S_2=(x_2,x_6)$ $S_3=(x_3,x_8)$ $S_4=x_4$ $S_2\oplus S_4=x_7$
$(5,3)$	$y_1=(x_1,x_2,x_3)$ $y_2=(x_2,x_3,x_4)$ $y_3=(x_3,x_4,x_5)$ $y_4=(x_1,x_4,x_5)$ $y_5=(x_2,x_5)$	$S_1=x_1$ $S_2=x_2$ $S_3=x_3$ $S_4=x_4$ $S_5=x_5$	$S_1=x_1$ $S_2=x_2$ $S_3=x_3$ $S_1\oplus S_3=x_4$ $S_1\oplus S_2=x_5$
$(7,4)$	$y_1=(x_1,x_2,x_3)$ $y_2=(x_1,x_3,x_4,x_7)$ $y_3=(x_2,x_5,x_6,x_7)$ $y_4=(x_1,x_3,x_5,x_6)$ $y_5=(x_4,x_5,x_6)$	$S_1=x_1$ $S_2=(x_2,x_4)$ $S_3=x_3$ $S_4=x_5$ $S_5=x_6$ $S_6=x_7$	$S_1=x_1$ $S_2=(x_2,x_4)$ $S_3=x_3$ $S_1\oplus S_2=x_5$ $S_3\oplus S_4=x_6$ $S_4=x_7$
$(6,3)$	$y_1=(x_1,x_2,x_3)$ $y_2=(x_2,x_3,x_4)$ $y_3=(x_1,x_4,x_5)$ $y_4=(x_1,x_4,x_6)$	$S_1=x_1$ $S_2=x_2$ $S_3=(x_3,x_5,x_6)$ $S_4=x_4$	$S_1=x_1$ $S_2=x_2$ $S_3=(x_3,x_5,x_6)$ $S_1\oplus S_3=x_4$

Table 3: Comparison of TPGs Designed

I. (4,2) CUT

Construction	N	Feedback polynomial	Taps	Test length	Seed
C1: Input separation	6	$P_6(x)$	3 - 1 4 2 -	$2^6 - 1$	$non - zero$
C2: Test signal	6	$P_6(x)$	O/E	$2^6 - 1$	$non - zero$
C3: Linear sum	4	$P_4(x)$	O/E	$2^4 - 1$	$non - zero$
C7: Cyclic code	6	$1 + x + x^2 + x^3 + x^4 + x^6$	O/E	$2^4 - 1$	101000

II. (8,4) CUT

Construction	N	Feedback polynomial	Taps	Test length	Seed
C1: Input separation	10	$P_{10}(x)$	4 - 7 2 6 1 5 8 3 -	$2^{10} - 1$	$non - zero$
C2: Test signal	10	$P_{10}(x)$	O/E	$2^{10} - 1$	$non - zero$
C3: Linear sum	8	$P_8(x)$	O/E	$2^8 - 1$	$non - zero$
C7: Cyclic code	10	$1 + x + x^3 + x^4 + x^7 + x^8 + x^9 + x^{10}$	O/E	$2^8 - 1$	1010000000

III. (5,3) CUT

Construction	N	Feedback polynomial	Taps	Test length	Seed
C1: Input separation	10	$P_{10}(x)$	1 - 2 - 3 - 4 - 5 -	$2^{10} - 1$	$non - zero$
C2: Test signal	10	$P_{10}(x)$	O/E	$2^{10} - 1$	$non - zero$
C3: Linear sum	6	$P_6(x)$	O/E	$2^6 - 1$	$non - zero$
C7: Cyclic code	10	$1 + x + x^3 + x^4 + x^7 + x^8 + x^9 + x^{10}$	O/E	$2^8 - 1$	1010000000

IV. (7,4) CUT

Construction	N	Feedback polynomial	Taps	Test length	Seed
C1: Input separation	13	$P_{13}(x)$	1 - 2 4 - 3 - 5 - 6 - 7 -	$2^{13} - 1$	$non - zero$
C2: Test signal	12	$P_{12}(x)$	O/E	$2^{12} - 1$	$non - zero$
C3: Linear sum	8	$P_8(x)$	O/E	$2^8 - 1$	$non - zero$
C7: Cyclic code	12	$1 + x + x^2 + x^3 + x^4 + x^6 + x^7 + x^9 + x^{11} + x^{12}$	O/E	$2^8 - 1$	101000000000

V. (6,3) CUT

Construction	N	Feedback polynomial	Taps	Test length	Seed
C1: Input separation	13	$P_{13}(x)$	1 - 2 - 4 - 3 5 6 -	$2^{10} - 1$	$non - zero$
C2: Test signal	8	$P_8(x)$	O/E	$2^8 - 1$	$non - zero$
C3: Linear sum	6	$P_6(x)$	O/E	$2^6 - 1$	$non - zero$
C7: Cyclic code	8	$1 + x + x^2 + x^3 + x^6 + x^8$	O/E	$2^6 - 1$	10100000

exhaustive testing compared to all the previously known TPG designs. Experimental results show that these techniques help design TPGs which achieve minimal test sequence length for the example circuits studied in previous (single pattern) pseudo-exhaustive testing research. The design methodology described can be easily used to design efficient TPGs for real circuits.

A number of issues are currently being investigated. Firstly, there is a need to further reduce the test sequence length for two pattern testing. There is an ongoing effort to reduce test time by the use of two-pattern pseudo-random testing using the basic framework presented here. An interesting spin-off of this work is design of efficient test pattern generators for single pattern testing. A number of such techniques are being currently investigated.

References

[ABF90] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, New York, N.Y., 1990.

[Ake85] S. B Akers. On the Use of Linear Sums in Exhaustive Testing. In *Proc. 15th Int'l. Symp. on Fault-Tolerant Computing*, pages 148–153, June 1985.

[CG93] Chih-Ang Chen and Sandeep K. Gupta. Exhaustive Testing for Stuck-Open and Delay Faults. *Submitted to the International Test Conference*, 1993.

[CK85] Gary L. Craig and Charles R. Kime. Pseudo-Exhaustive Adjacency Testing: A BIST Approach for Stuck-Open Faults. In *Proceedings IEEE International Test Conference*, pages 126–137, 1985.

[FM91] Kiyoshi Furuya and Edward J. McCluskey. Two-Pattern Test Capabilities of Autonomous TPG Circuits. In *Proceedings IEEE International Test Conference*, pages 704–711, October 1991.

[LC83] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications.* Prentice Hall, Englewood Cliffs, N.J., 1983.

[McC84] E. J. McCluskey. Verification Testing — A Pseudoexhaustive Test Technique. *IEEE Trans. on Computers*, C-33(6):541–546, June 1984.

[PG91] D. K. Pradhan and S. K. Gupta. A New Framework for Designing and Analyzing BIST Techniques and Zero Aliasing Compression. *IEEE Trans. on Computers*, C-40(6), June 1991.

[SB91] Jacob Savir and Robert Berry. At-Speed Test Is Not Necessarily an AC Test. In *Proceedings IEEE International Test Conference*, pages 722–728, 1991.

[Sta84] C. W. Starke. Built-In Test for CMOS Circuit. In *Proceedings IEEE International Test Conference*, pages 309–314, 1984.

[WM88] L. T. Wang and E. J. McCluskey. Circuits for Pseudoexhaustive Test Pattern Generation. *IEEE Trans. on CAD*, 7(10):1068–1080, October 1988.

[ZBM92] S. Zhang, R. Byrne, and D. M. Miller. BIST Generators for Sequential Faults. In *Proceedings IEEE International Conference on Computer Design*, pages 260–263, 1992.