BIST Test Pattern Generators For Two-Pattern Testing – Theory and Design Algorithms

Chih-Ang Chen and Sandeep K. Gupta

CENG Technical Report 93-34

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

December 1993

BIST Test Pattern Generators for Two-Pattern Testing — Theory and Design Algorithms *

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

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

Abstract

Testing for delay and CMOS stuck-open faults requires two pattern tests and test sets are usually large. Built-in self-test (BIST) schemes are attractive for such comprehensive testing. The BIST test pattern generators (TPGs) for such testing should be designed to ensure high pattern-pair coverage. In this paper, necessary and sufficient conditions to ensure complete/maximal pattern-pair coverage for linear feedback shift register (LFSR) and cellular automata (CA) have been derived. The theory developed here identifies all LFSR/CA TPGs which maximize pattern-pair coverage under any given TPG size constraints. It is shown that LFSRs with primitive feedback polynomials with large number of terms are better for two-pattern testing. Also, CA are shown to be better TPGs than LFSRs (for two-pattern testing) independent of their feedback rules. Efficient algorithms based the necessary and sufficient conditions to design optimal TPGs for two-pattern testing have been developed. Experiments on some benchmark circuits indicate the TPGs designed using the procedures outlined in this paper provide much higher delay fault coverage than other TPGs.

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

1 Introduction

Traditional focus of VLSI testing in general, and BIST in particular, has been on maximization of stuck-at fault coverage. (Furthermore, most BIST techniques design test pattern generators for each combinational logic block separately. In this paper, the circuit under test (CUT) will also be assumed to be combinational.) Some techniques, such as pseudo-exhaustive testing, go beyond single stuck-at faults and implicitly target all multiple stuck-at faults and some bridging faults. However, a large class of physical defects do not map into these fault categories. Some physical defects, such as transistor stuck-open faults in CMOS, can convert the combinational CUT into a sequential one. This is due to the fact that in presence of a stuck-open fault, the output of the faulty gate can be tri-stated and retain the charge due to the previous vector. It is necessary that the previous vector (called initialization vector $V1$) should appropriately charge (discharge) the gate output to ensure that an error is generated when the test vector (V_2) is applied. Hence, many stuck-open faults require two-pattern tests. Similarly, delay faults require two pattern tests since the delays in the circuit paths depend on the pair of input patterns applied.

There is growing interest in coverage of these faults. Firstly, increasing quality level requirements constantly require better testing methodologies. Furthermore, increasing clock rates and use of aggressive statistical timing can cause a circuit to malfunction even if each device in a fabricated chip performs within its worst case delay tolerance. Hence, chip manufacturers are beginning to augment their test methodologies to test for delay faults. Typically, expensive testers are required to apply delay tests to circuits. Also, test time can be large due to long test sequences required. Hence, BIST can effectively reduce the cost of delay testing.

One important consideration in design of BIST test pattern generators (TPG) for two-pattern testing is to ensure that adequate number of **pattern pairs** are applied to the combinational CUT. The pattern-pair coverage is measured by the metric transition coverage and its maximization is the main focus of this paper. Necessary and sufficient conditions for a TPG to achieve complete pattern-pair coverage have been derived for commonly used TPGs — LFSR and CA. Cellular automata have been shown to have large number of ways in which complete/maximal transition coverage can be achieved. A TPG design procedure that maximizes transition coverage under a given test time/TPG hardware constraint has been developed. The TPG designed by using the results derived are validated with robust path delay fault simulation on synthesized benchmark circuits.

The following is organized into seven main sections. Section 2 reviews the related research. Some preliminary definitions are given in Section 3. Necessary and sufficient conditions for designing a LFSR or CA to achieve complete and maximal transition coverage are derived and comparison of the two TPGs are discussed in Section 4. TPG design algorithms based on these conditions are described in Section 5. Robust path delay fault simulation results discussed in Section 6 show that TPG designs that meet the conditions derived in this paper are superior to other TPGs. Finally, concluding remarks and future research directions are presented in Section 7.

2 Review of Previous Results

In [19], a $2n$-stage non-linear feedback shift register with alternate stages connected to CUT inputs is proposed to generate pre-determined ordered two-pattern tests. The feedback function is a PLA-like function calculated from interlaced test pairs. Although this TPG design can generate the desired test pairs in short test time, the high complexity of the non-linear feedback network limits its applicability. In *adjacency testing* [6], only two-pattern tests that differ in single bit are generated. This scheme reduces the maximum number of test pairs to $n(2^n - 1)$. As shown in [18], such test pairs are sufficient to robustly detect all path delay faults. Design of circuits that generate such (V_1, V_2) test pairs exhaustively are presented in [6]. However, due to non-linear nature of these TPGs area overhead is high.

Input separation [15] was introduced to remove the correlation between consecutive stages in a scan chain. Inputs that fan-out to common outputs are assigned to non-adjacent stages of the scan chain. This method can also be used for BIST TPG design as shown in Section 5.1. However, the existence of XOR gates between stages of a BIST TPG allows extra freedom (not available in a scan chain design) which may lead to a more compact TPG design.

Also introduced in [15] is the notion of *AC strength*, which is defined as the ratio of the maximum number of two-pattern tests that can be generated by a TPG to the exhaustive two-pattern count $2^n(2^n - 1)$. It quantifies the two-pattern test capability of a TPG. A similar metric called *transition count* was defined in [8]. The relation between a linear TPG design and the rank of submatrices derived from its transition matrix was also developed in [8]. The transition coverage of some autonomous linear pattern generators was studied in [8]. A metric (similar to AC strength) called *transition count* was proposed. It is a measure of distinct pattern pairs that occur in the sequence generated by a TPG. Transition

count for an n-input circuit is the number of distinct two-pattern tests applied at its inputs and is less than or equal to $2^{2n} - 1$.

Further study on transition coverage has been reported in [20]. The transition counts of linear feedback shift registers (LFSRs) and linear hybrid cellular automata (LHCA) have been computed. Two new test pattern generators, XLFSRs and XLHCA, are defined. An XLFSR is an LFSR whose odd stage outputs are grouped together, followed by even stage outputs. An XLHCA is obtained similarly from an LHCA. This configuration maximizes transition coverage if n CUT inputs are connected to n contiguous outputs of the TPG. However, the TPG designs in [20] are too restrictive. There are a large number of TPGs with maximum transition count in addition to those reported in [20]. In this paper, rules of designing a TPG (LFSR/LHCA) that generates the highest transition coverage are given. XLFSRs and XLHCA can be shown to be special cases of our results.

Modifications of circular self-test path (CSTP) [10] to improve the path delay fault coverage were proposed in [14]. Three BIST circuits were described, namely, double CSTP, double-flip-flop CSTP, and simulated CSTP. A double CSTP consists of two circular self-test paths with data shifted in opposite directions. One of the paths also serves as the test response compressor for the other path. A double-flip-flop CSTP adds an extra flip-flop between two consecutive flip-flops in a conventional CSTP. The above two approaches require twice the number of flip-flops. In a stimulated CSTP, the primary input CSTP section compacts test response from a CUT (stimulated CSTP section). This requires that the number of flip-flops in the stimulated CSTP section matches the one in the primary input CSTP section. These modifications introduce extra freedom in generating two-pattern tests. Hence, the fault coverage of the simulation results approach that of a $2n$-stage LFSR.

3 Preliminaries

In this section, some preliminary notions and definitions are described. In the following, n denotes the number of CUT inputs and m denotes the number of TPG stages. Two-pattern exhaustive testing for an n-input circuit requires that all possible 2^n vectors are applied as V_1, each followed by all possible $(2^n - 1)$ V_2's ($\neq V_1$). By properly ordering the vector pairs, the number of vectors required is bounded by $2^n(2^n - 1) = 2^{2n} - 2^n$. Because $2^{2n} - 2^n > 2^{2n-1}$ (for any $n > 1$), a finite state machine with $2n$ stages is necessary to generate these test patterns. *Transition coverage* (or *two-pattern coverage*) is the number of distinct two-pattern tests applied by a TPG to CUT inputs.

The transfer function of an m-stage autonomous linear sequential TPG is represented by a transition matrix T given by [8]

$$T = \begin{pmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,m} \\ g_{2,1} & g_{2,2} & g_{2,3} & \cdots & g_{2,m} \\ g_{3,1} & g_{3,2} & g_{3,3} & \cdots & g_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{m,1} & g_{m,2} & g_{m,3} & \cdots & g_{m,m} \end{pmatrix}, \tag{1}$$

where $g_{i,j} \in \{0,1\}(1 \leq i,j \leq m)$. In general, the next state x' and current state x of a TPG are related by

$$x' = Tx. \tag{2}$$

For thorough two-pattern testing, the number of TPG stages (m) is normally larger than the number of CUT inputs (n). Hence, only a subset of the TPG outputs are connected to the CUT inputs. *Tapped variables* (or simply *taps*) $\mathbf{v} = \{v_1, v_2, \ldots, v_n\}$ are defined as the flip-flops (stages of the TPG) whose outputs are connected to the CUT inputs. The remaining ones $\mathbf{u} = \{u_1, u_2, \ldots, u_{m-n}\}$ are the *untapped* variables. Let $X_v = \{x_{v_1}, x_{v_2}, \ldots, x_{v_n}\}$ and $X_u = \{x_{u_1}, x_{u_2}, \ldots, x_{u_{m-n}}\}$ be the states of the TPG corresponding to the tapped and untapped variables. Then the next state of the tapped variables can be represented by [8]

$$X_v{}' = T_v X_v + T_u X_u, \tag{3}$$

where T_v and T_u are submatrices of T of sizes $n \times n$ and $n \times (m-n)$. The submatrix T_u, given by

$$T_u = \begin{pmatrix} g_{v_1,u_1} & g_{v_1,u_2} & g_{v_1,u_3} & \cdots & g_{v_1,u_{m-n}} \\ g_{v_2,u_1} & g_{v_2,u_2} & g_{v_2,u_3} & \cdots & g_{v_2,u_{m-n}} \\ g_{v_3,u_1} & g_{v_3,u_2} & g_{v_3,u_3} & \cdots & g_{v_3,u_{m-n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{v_n,u_1} & g_{v_n,u_2} & g_{v_n,u_3} & \cdots & g_{v_n,u_{m-n}} \end{pmatrix}, \tag{4}$$

is constructed from the n rows $\{v_1, v_2, \ldots, v_n\}$ of T, with the corresponding n columns removed. If r is the rank of T_u, then there are 2^r distinct transitions from each X_v state. The rank of T_u thus determines the transition coverage of the n tapped variables [8]. In the following, the transition coverage is used to evaluate the quality of a TPG. The problem of selecting $\mathbf{v}$ to maximize the transition coverage is referred as the *tap selection problem.*

Consider an m-stage TPG and an n-input CUT with $0 < n \leq m$. A tap selection is said to have *maximal transition coverage* if the TPG generates the maximum possible

number of two-pattern tests for the given TPG size. A tap selection is said to have *complete transition coverage* if all 2^{2n} two-pattern tests are applied to the CUT inputs. These definitions assumes that, if the TPG does not generate a test sequence of maximal length [1, 9], it is seeded multiple times until all possible patterns are generated. To obtain maximal transition coverage, the submatrix T_u must have full rank $r = min\{n, m-n\}$. There are 2^n possible input combinations for an n-input CUT, each with 2^r possible next states. Maximal transition coverage is thus given by 2^{n+r}. The notions of complete and maximal transition coverage coincide if $m \geq 2n$ (or equivalently, $\lfloor \frac{m}{2} \rfloor \geq n$). In this case, a TPG with $m \geq 2n$ is necessary to have complete transition coverage, but it is not sufficient. The n taps must be carefully selected to avoid dependency between consecutive stages of a TPG. If $m < 2n$ (or $\lceil \frac{m}{2} \rceil \leq n$), it is impossible to achieve complete transition coverage. Maximal transition coverage is simply $2^m - 1$. However, the taps must still be carefully selected to obtain maximal transition coverage. One main subject of this paper is the identification of necessary and sufficient conditions that an n-tap selection must satisfy to obtain complete/maximal transition coverage for an m-stage LFSR (type 1 or type 2) or CA assuming that CUT outputs depend on all n inputs. (These theoretical results are used in the design algorithms described in Section 5.2).

An output y_j of a circuit *depends* on an input x_i, denoted by $x_i \rightsquigarrow y_j$, if there exists a directed path from x_i to y_j. For an n-input CUT with a single output, the minimum number of stages required to achieve complete transition coverage is $2n$ for any linear TPG. However, in reality, a circuit normally has multiple outputs and each output depends only on a small subset of the primary inputs. Such a circuit is called a *partial dependence (PD)* circuit. This observation inspired the idea of *pseudo-exhaustive testing* [13]. Consider a circuit with n inputs $\{x_1, x_2, \ldots, x_n\}$ and p outputs $\{y_1, y_2, \ldots, y_p\}$. Let the j-th output y_j be a function of k_j inputs, i.e. $y_j = f_j(x_{i_1}, x_{i_2}, \ldots, x_{i_{k_j}})$. All gates and inputs in the transitive fan-in of y_j are called $cone_j$. An input x_i *belongs* to $cone_j$ if $x_i \rightsquigarrow y_j$. In pseudo-exhaustive testing, each cone is tested exhaustively. Assume that the largest cone depends on k inputs, i.e. $k = \max_{j=1}^{p} k_j$. Then the circuit can be tested two-pattern pseudo-exhaustively with W tests, where $2^{2k} \leq W \leq 2^{2n}$. If k is small compared to the number of inputs n, then the circuit can be tested using a shorter sequence without decreasing fault coverage. The theory developed in this paper can be applied to each cone of a partial dependence circuit. Algorithms for TPG design that exploit the theory are presented in Section 5.2.

4 Complete Conditions for Two-Pattern Coverage

A $2n$-stage TPG (LFSR or CA) with all odd or even taps is sufficient to generate the complete two-pattern tests for an n-input CUT. There exist other tap selections that also achieve complete transition coverage. The ability to identify these tap selections not only provides theoretical bounds for different TPG schemes, but also leads to more compact TPG designs for pseudo-exhaustive testing. In this section, necessary and sufficient conditions for commonly used TPG designs (namely, LFSR and CA) to achieve complete or maximal transition coverage are derived. These conditions are derived for an n-input circuit assuming a single output. The application of these results to PD circuits is described in the next section.

4.1 Linear Feedback Shift Registers

LFSRs are a class of linear sequential machines constructed from D flip-flops and modulo-2 adders (XOR gates). An m-stage LFSR is characterized by its *feedback polynomial* [1, 9] given by

$$P(x) = 1 + c_1 x + \cdots + c_{m-1} x^{m-1} + x^m \tag{5}$$

If the feedback polynomial is *primitive* [11], then the LFSR (initialized to any non-zero state) generates a sequence of length $2^m - 1$ before returning to the initial state. Such an LFSR is called a *maximal length* LFSR.

The *transition matrices* T_{LFSR1} and T_{LFSR2} of a type 1 (external XOR) and a type 2 (internal XOR) LFSR [1] are given by

$$T_{LFSR1} = \begin{pmatrix} c_{m-1} & c_{m-2} & c_{m-3} & \cdots & c_1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \qquad T_{LFSR2} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & 0 & \cdots & 0 & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_{m-2} \\ 0 & 0 & 0 & \cdots & 1 & c_{m-1} \end{pmatrix}$$

The two LFSRs with the transition matrices given above are called *dual* LFSRs. The symmetry of the two matrices introduces some duality for tap selections of the two types of LFSRs. For reasons given later, a TPG based on a type 2 LFSR normally requires fewer flip-flops than that based on a type 1 LFSR for two-pattern pseudo-exhaustive testing. In

the sequel, only type 2 LFSRs are described. The corresponding results for type 1 LFSRs can be derived from duality [4].

4.1.1 General Results

In this section, necessary and sufficient conditions for a tap selection to ensure complete and maximal transition coverage for a type 2 LFSR are derived. First, consider the case $m \geq 2n$. In this case, the $n \times (m-n)$ submatrix T_u has full rank $r = n$ if all its rows are linearly independent.

Lemma 1 *For an m-stage type 2 LFSR, if no two consecutive stages are tapped, then the matrix T_u has full row rank.*

Proof: The condition implies that any tapped stage ν must be preceded by an untapped stage $\nu - 1$. The unique nonzero entry $g_{\nu,\nu-1}$ in column $\nu - 1$ of T_{LFSR2} is included in T_u. Hence, all rows of T_u are linearly independent. □

Lemma 2 *For an m-stage type 2 LFSR, if stage 1 and m are both untapped and there exists exactly one incidence of consecutive tapped stages $\nu - 1$ and ν with $c_{\nu-1} = 1$, then the matrix T_u has full row rank.*

Proof: Since stage 1 and m are untapped, the only nonzero entry $g_{1,m}$ in row 1 is excluded from T_u. Assume there is only one incidence of consecutive tapped stages $\nu - 1$ and ν with $c_{\nu-1} = 1$. Because all the other columns have unique nonzero entries in different columns as shown in Lemma 1, the matrix T_u must have full column rank. If there exist more than one incidence of consecutive tapped stages with nonzero coefficients, all of them map to identical rows in T_u and thus are linearly dependent. □

Theorem 1 *For an n-input CUT and an m-stage type 2 LFSR with $m \geq 2n$, a tap selection provides complete transition coverage if and only if*

1. *no two consecutive stages are tapped, or*
2. *stage 1 and m are untapped and there exists exactly one incidence of consecutive tapped stages $\nu - 1$ and ν with $c_{\nu-1} = 1$.*

Proof: Since $m \geq 2n$, the matrix T_u must have full row rank.

The sufficiency of the theorem is a direct result of Lemma 1 and 2.

For necessity, assume that two consecutive stage $\nu-1$ and ν are tapped. The nonzero entry $g_{\nu,\nu-1}$ is removed from T_u. In order for T_u to still have full row rank, stage 1 and m must be untapped, $c_{\nu-1}$ must be 1, and stage $\nu-1$ and ν must be the only consecutive tapped stages. Otherwise, the rows of T_u are linearly dependent. Next, if either one of stage 1 or m is tapped, then tapping consecutive stages $\nu-1$ and ν either introduces an all zero row (when $c_{\nu-1}=0$) or creates identical rows (when $c_{\nu-1}=1$) in T_u. In both cases, the tap selection does not obtain complete transition coverage. □

Complete transition coverage for an n-input CUT is achievable only if an m-stage TPG with $m \geq 2n$ is used. Constraints on the TPG size and test time may require that $m < 2n$. The following results identify all tap selections that maximize transition coverage under such constraints.

Lemma 3 *For an m-stage type 2 LFSR, if two consecutive stages $\mu-1$ and μ $(1 < \mu \leq m)$ are untapped, then the matrix T_u does not have full column rank.*

Proof: If stages $\mu-1$ and μ are untapped, then columns $\mu-1$ and μ with rows $\mu-1$ and μ removed are included in T_u. Since $g_{\mu,\mu-1}$ is the only non-zero entry in column $\mu-1$, the corresponding column in T_u must be zero. □

Lemma 4 *For an m-stage type 2 LFSR, if stages 1 and m are untapped and the coefficient $c_{\nu-1}$ is zero for each pair of consecutive tapped stages $\nu-1$ and ν, then the matrix T_u does not have full column rank.*

Proof: If stage ν is tapped and $c_{\nu-1}=1$, then stage $\nu-1$ must be untapped. Otherwise, the given condition is violated. With $\nu-1$ untapped and ν tapped, the unique nonzero entry $g_{\nu,\nu-1}$ is included in the corresponding row of T_u. Because stage 1 and m are untapped, the last column (column $m-n$) of T_u is given by $(c_{v_1-1}c_{v_2-1}\cdots c_{v_n-1})$. For every nonzero coefficient $c_{\nu-1}$ $(\nu \in \mathbf{v})$ in column $m-n$ of T_u, there exists a column with unique nonzero entry at $g_{\nu,\nu-1}$. Hence, column $m-n$ and the corresponding columns of T_u sum up to zero. □

Theorem 2 *For an n-input CUT and an m-stage type 2 LFSR with $m < 2n$, a tap selection achieves maximal transition coverage if and only if*

1. *no two consecutive stages are untapped, and*

2. *if stage 1 and m are untapped, there exists at least one pair of consecutive tapped stages* $\nu - 1$ *and* ν *with* $c_{\nu-1} = 1$ *for* $2 < \nu < m$.

Proof: Selecting n taps, where $m < 2n$, is the same as unselecting $m - n$ taps, where $m > 2(m - n)$. The matrix T_u must have full column rank.

The necessity of these conditions is a direct result of Lemma 3 and 4.

The sufficiency of the theorem is proven by contradiction. If the matrix T_u does not have maximal transition coverage, then either T_u has a zero column or some columns of T_u add up to zero. For the first case, let stage μ be the untapped stage corresponding a zero column in T_u. Since $g_{\mu,\mu-1}$ is the only nonzero entry in column $\mu - 1$ of T_{LFSR2}, stage $\mu - 1$ of T_{LFSR2} must also be untapped. For the second case, if some columns of T_u add up to zero, they must include the $(m - n)$-th column, because all the other columns have unique 1's in different rows. The entry $g_{1,m}$ is the only nonzero entry in row 1 and can not be included in T_u. Hence, stage 1 and m must be untapped. In addition, for every nonzero entry $c_{\nu-1}$ in the $(m - n)$-th column of T_u, the nonzero entry $g_{\nu,\nu-1}$ in column $\nu - 1$ of T_{LFSR2} must also be included in T_u. This requires that stage $\nu - 1$ be untapped and stage ν be tapped. In order that columns of T_u add up to zero, the feedback coefficient $c_{\nu-1}$ for each pair of consecutive tapped stage $\nu - 1$ and ν must be zero. Therefore, if matrix T_u does not have full rank, either consecutive stages are untapped or stage 1 and m are untapped and for each consecutive tapped stages $\nu - 1$ and ν, the feedback coefficient $c_{\nu-1}$ is zero. □

If permutations of CUT inputs are not considered, there are totally $\binom{m}{n}$ possible tap selections. They can be divided into three categories. The tap selections satisfying condition 1 of Theorem 1 (Theorem 2) obtain complete (maximal) transition coverage independent of the feedback polynomial. Such tap selections are *feedback independent*. Other tap selections satisfying the theorems are *feedback dependent*. The remaining tap selections always have less than optimal transition coverage. Determining the number of choices in each category is of practical interest. Generally, a larger number of choices indicates that the optimum tap selection can be found more easily.

Lemma 5 *Let* N_{fi}^{comp} *(*N_{fd}^{comp}*) be the number of feedback independent (dependent) tap selections for an n-input CUT and an m-stage LFSR with* $m \geq 2n$. *Then*

$$N_{fi}^{comp} = \binom{n+1}{m-n} - \binom{n-1}{m-n-2} \tag{6}$$

$$N_{fd}^{comp} = (n-1)\binom{n-2}{m-n-2} \tag{7}$$

Proof: View tapped stages as full boxes and untapped stages as empty boxes. There are n full boxes. These full boxes are separated by $n+1$ walls. Now we replace $m-n$ out of the $n+1$ walls with empty boxes. Such a replacement makes sure that no two full boxes are abutted. The number of choices is simply $\binom{n+1}{m-n}$. If stage 1 and m are tapped, the tap selection does not have complete transition coverage. The number of choices for this case, given by $\binom{n-1}{m-n-2}$, can be derived by replacing the first and the last wall with full boxes and choosing $m-n-2$ walls out of the remaining $n-1$ (to be replaced by full boxes). The number N_{fi}^{comp} is obtained by subtracting $\binom{n-1}{m-n-2}$ from $\binom{n+1}{m-n}$. For the feedback dependent ones, stage 1 and m are untapped and one incidence of consecutive tapped stages is allowed in-between. Since the first and the last box is empty, only $n-1$ walls are considered. The consecutive tapped stages correspond to a wall not replaced by an empty box. There are $n-1$ ways of selecting this wall. Of the remaining $n-2$ walls, $m-n-2$ are replaced with empty boxes. Overall, the number N_{fd}^{comp} is given by $(n-1)\binom{n-2}{m-n-2}$. □

Corollary 1 *The number of tap selections with complete transition coverage, N^{comp}, is tightly bounded by $N_{fi}^{comp} \leq N^{comp} \leq N_{fi}^{comp} + N_{fd}^{comp}$.*

Proof: The lower (upper) bound occurs when $c_i = 0$ (1) for $1 \leq i \leq m-1$. The feedback polynomial is simply $P(x) = 1 + x^m$ ($P(x) = 1 + x + x^2 + \cdots + x^m$). □

Lemma 6 *For the case $m < 2n$, let N_{fi}^{max} (N_{fd}^{max}) denote the number of feedback independent (dependent) tap selections with maximal transition coverage. Then*

$$N_{fi}^{max} = \binom{m-n+1}{n} - \binom{m-n-1}{n-2} \tag{8}$$

$$N_{fd}^{max} = \binom{m-n-1}{n-2} \tag{9}$$

Proof: For this case, we untapping $m-n$ stages, where $m > 2(m-n)$. The derivation of the number of choices for the feedback independent case is similar to Lemma 5, with the exception that n is now $m-n$ and tapped (untapped) stages are viewed as empty (full)

boxes. Also, if stage 1 and m are untapped, the transition coverage depends on the feedback coefficients (in contrast to the previous case that the transition coverage is always less than optimal). Therefore, the number N_{fd}^{max} is simply $\binom{m-n-1}{n-2}$. □

Corollary 2 *The number of tap selection with maximal transition, N^{max}, is tightly bounded by $N_{fi}^{max} \leq N^{max} \leq N_{fi}^{max} + N_{fd}^{max}$.*

Proof: The proof is similar to that for Corollary 1. □

4.1.2 Special Case $m = 2n$

For this case, a $2n$-stage type 2 LFSR with all odd/even stages connected the CUT inputs can generate a two-pattern exhaustive test set for any n-input circuit. However, these are only the two feedback independent tap selections to achieve complete transition coverage in Lemma 5. The following theorem gives all possible tap selections for the special case $m = 2n$.

Corollary 3 *For an n-input CUT and a $2n$-stage type 2 LFSR with the feedback polynomial $P(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{2n} x^{2n}$, the tap selections to achieve complete transition coverage are*

1. *Select all odd (or all even) stage outputs, or*
2. *Select stages $\underbrace{2, \ldots, 2i}_{even}, \underbrace{2i+1, \ldots, 2n-1}_{odd}$ for any i such that $c_{2i} = 1$.*

Proof: The theorem is a direct result of Theorem 1. Sine $m = 2n$ and no two consecutive stages are tapped, the taps must be all odd or all even. The exception is when stage 1 and $2n$ are untapped. For this case, we must tapped all even stages followed by all odd stages. Then there exists one incidence of tapped stages $2i$ and $2i+1$. By Theorem 1, the feedback coefficient c_{2i} must be 1. □

Corollary 4 *The number of possible ways to connect the n inputs of a CUT to a $2n$-stage type 2 LFSR for two-pattern exhaustive testing is given by $\sum_{i=0}^{n} c_{2i}$.*

Proof: Proof follows directly from the above theorem. □

The summation in Corollary 4 is maximum if $c_{2i} = 1, \forall i$. Therefore, a type 2 LFSR with many non-zero coefficients of the form c_{2i} offer more choices of tap selections for two-pattern exhaustive testing. This summation can also be obtained by replacing m with $2n$ in Lemma 5. The numbers of the feedback independent and feedback dependent tap selections that achieve complete transition coverage are 2 and $n-1$ respectively.

Example 1 *Consider a 5-input CUT and a 10-stage type 2 LFSR with generator polynomial given by* $P(x) = 1+x^2+x^3+x^4+x^8$. *Since the coefficient* c_0, c_2, c_4, *and* c_8 *are 1, the number of possible tap selections to achieve complete transition coverage is 4. They are* $(1,3,5,7)$, $(2,4,6,8)$, $(2,3,5,7)$, *and* $(2,4,5,7)$. *The first two choices (all odd or all even) are feedback independent.*

4.2 Cellular Automata

The cellular automata considered in the following are linear, rule 90/150 with null boundary conditions [2, 16]. In a 90/150 CA, the next state of stage i is a function of the present state of itself (if it is rule 150) and its neighbouring stages $(i-1)$ and $(i+1)$. The exceptions are the first and last stages which are connected to only one neighbor (null boundary condition).

The transition matrix T_{CA} of a CA with null boundary conditions is given by

$$T_{CA} = \begin{pmatrix} c_1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & c_2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & c_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & c_{m-1} & 1 \\ 0 & 0 & 0 & \cdots & 1 & c_m \end{pmatrix}, \tag{10}$$

where $c_i = 1$ (0) if stage i is of rule 150 (90).

4.2.1 General Results

In this section, necessary and sufficient conditions to ensure complete and maximal transition coverage for a CA are derived.

Lemma 7 *The transition coverage of a tap selection for a CA is independent of the rules.*

Proof: The transition coverage is determined by the rank of the matrix T_u. If stage i is tapped, then row i with column i removed is included in T_u. The element $g_{i,i}$ on the diagonal, which has different values depending on the rules used, never appears in T_u. Hence, the transition coverage of a CA tap selection is rule independent. □

The transition matrix T_{CA} is symmetric along the second diagonal but for the entries on the primary diagonal. This implies that duality rules can be derived for the two cases: $m \geq 2n$ and $m < 2n$. In this section, we only consider tap selections for the case $m \geq 2n$ in detail.

To simply the following statements, tapping (untapping) stage 1 or stage m of a CA is considered as two consecutive tapped (untapped) stages.

Lemma 8 *For an m-stage cellular automata, if three consecutive stages are tapped, then the matrix T_u does not have full row rank.*

Proof: There are at most three non-zero entries $g_{\nu-1,\nu}$, $g_{\nu,\nu}$, and $g_{\nu,\nu+1}$ in row ν of T_{CA}. If all three stages are tapped, then the row of T_u corresponding to row ν of T_{CA} must be zero. Clearly, row 1 (m) has at most two non-zero entries. If we attach a dummy stage 0 ($m+1$) adjacent to stage 1 (m) and consider it as tapped/untapped as stated earlier, then the lemma can be treated uniformly. □

Lemma 9 *For an m-stage cellular automata, if between two sets of consecutive tapped stages, only alternating tapped and untapped stages exist, then the matrix T_u does not have full row rank.*

Proof: Let $\nu - 1$ and ν be a pair of consecutive tapped stages and p be the number of pairs of alternating tapped and untapped stages before another pair of tapped stages occurs. Clearly, the tapped stages are $\{\nu-1, \nu, \nu+2, \nu+4, \ldots, \nu+2p, \nu+2p+1\}$ and the untapped stages are $\{\nu+1, \nu+3, \ldots, \nu+2p-3, \nu+2p-1\}$. The rows in T_u corresponding to the tapped stages $\{\nu, \nu+2, \nu+4, \ldots, \nu+2p\}$ sum up to zero. □

Theorem 3 *For an m-stage CA with arbitrary 90/150 rules, an n-tap selection with $m \geq 2n$ obtains complete transition coverage if and only if*

1. *no three consecutive tapped stages exist, and*

2. *between two sets of consecutive tapped stages, there exists one set of consecutive untapped stages.*

Proof: Since $m \geq 2n$, the matrix T_u must have full row rank.

The necessity is a direct result of Lemma 8 and Lemma 9.

The sufficiency is proven by contradiction. If the matrix T_u does not have full row rank, then one of the rows in T_u is zero or some rows of T_u add up to zero. For the first case, let row ν be the row in T_{CA} corresponding to the zero row in T_u. Since the entry $g_{\nu,\nu-1}$ and $g_{\nu,\nu+1}$ are 1, we must also tap row $\nu - 1$ and $\nu + 1$ in order to have a zero row in T_u. That is, three consecutive stages are tapped. For the second case, if some rows of T_u add up to zero, then the rows must be contiguous, each contains two consecutive 1's shifted by one position from the adjacent rows. For these rows to sum up to zero, there must be two pairs of tapped stages at both ends of these rows to create rows with a single non-zero entry and alternately tapped and untapped stages in-between such that the structure is continuous. □

Due to the assumption, if stage 1 (or stage m) is selected, it is considered as selecting two consecutive stages. Hence, stage 1 and 2 can never be selected together without violating the first condition of Theorem 3. This assumption should be used to interpret all results in this section.

The theorem for the case $m < 2n$ can be derived from duality. The proof is similar and hence omitted.

Theorem 4 *For an m-stage cellular automata with arbitrary 90/150 rules, an n tap selection with $m < 2n$ achieves maximal transition coverage if and only if*

1. *no three consecutive untapped stages exist, and*
2. *between two sets of consecutive untapped stages, there exists one set of consecutive tapped stages.*

In this theorem, if stage 1 is untapped, it is considered as a pair of consecutive untapped stages. This implies that, in order to achieve maximal transition coverage, at least one of stages 1 and 2 must be tapped. Similar argument holds for stage m.

Lemma 10 *For an n-input CUT and an m-stage CA with $m \geq 2n$, the number of tap selections that achieve complete transition coverage is given by*

$$\sum_{r=0}^{\lfloor \frac{n}{2} \rfloor} \binom{m-2n+r}{r} S(m,n,r), \tag{11}$$

where $S(m,n,r)$ is given by

$$S(m,n,r) = \begin{cases} 1 & \text{if } r > \frac{n-1}{2} \\ \frac{(m-n+1)!}{(m-2n+2r-1)!(n-2r)!} & \text{otherwise} \end{cases} \tag{12}$$

Proof: As in the proof of Theorem 5, untapped (tapped) stages are viewed as empty (full) boxes. There are $m-n+1$ walls that separate the $m-n$ empty boxes. Let's further assign three different colors *black*, *red*, and *white* to the walls, each to be replaced by two, one, and zero full boxes. Let r be the number of *black* boxes. Then the number of *red* and *white* walls are given by $n-2r$ and $m-2n+r+1$ respectively. From Theorem 3, between two pairs of tapped stages, there exists one pair of untapped stages, or equivalently, *black* walls must be separated by *white* walls. There are $m-2n+r$ spaces between $m-2n+r+1$ white walls (not counting the leftmost and the rightmost spaces since the null boundaries are considered tapped). We now replace r of the spaces with *black* walls. Such a replacement ensures that *black* walls are surrounded by *white* walls. There are $\binom{m-2n+r}{r}$ possible arrangements. The remaining $n-2r$ *red* walls are free to occupy the $m-2n+2r+2$ spaces (number of *black* and *white* walls + 1) between *black* and *white* walls, The problem can be modeled as displaying $n-2r$ flags with the same color on $m-2n+2r+2$ poles in a row. The solution has been shown to be $\frac{(m-n+1)!}{(m-2n+2r+1)!(n-2r)!}$ [7]. The bound of r is given by $0 \leq r \leq \lfloor \frac{n}{2} \rfloor$. Overall, the solution is obtained by taking summation of all possible values of r, or

$$\sum_{r=0}^{\lfloor \frac{n}{2} \rfloor} \binom{m-2n+r}{r} \frac{(m-n+1)!}{(m-2n+2r+1)!(n-2r)!} \tag{13}$$

The above equation is valid only when $m-2n+2r+2 \leq m-n+1$, or $r \leq \frac{n-1}{2}$. If $m = 2n = 4r$, then $r > \frac{n-1}{2}$. It is obvious that there exists one solution for this case. □

Lemma 11 *For an n-stage CUT and an m-stage CA with $m < 2n$, the number of tap selections which maximize the transition coverage is given by*

$$\sum_{r=0}^{\lfloor \frac{n}{2} \rfloor} \binom{2n-m+r}{r} S(m,n,r), \tag{14}$$

where $S(m,n,r)$ *is given by*

$$S(m,n,r) = \begin{cases} 1 & \text{if } r > \frac{m-n-1}{2} \\ \frac{(n+1)!}{(2n-m+2r+1)!(m-n-2r)!} & \text{otherwise} \end{cases} \quad (15)$$

Proof: Selecting n taps, where $m < 2n$, is the same as unselecting $m-n$ taps, where $m > 2(m-n)$. The equation is obtained by replacing n in Theorem 10 with $m-n$. □

4.2.2 Special Case $m = 2n$

Corollary 5 *Given a* $2n$*-stage rule* 90/150 *linear CA with null boundary conditions and an* n*-stage CUT, the following rules identify all tap selections achieving complete transition coverage.*

1. *Pair successive (odd, even) stages together.*
2. *Select one stage from each pair of (odd, even) stages.*

Proof: The theorem is a necessary results if we were to select n taps out of an $2n$-stage CA without violating Theorem 3. □

Corollary 6 *The* $2n$ *stage of a CA can be connected to the* n *inputs of a CUT in* 2^n *possible ways.*

Proof: Since there are two choices for each of the n pairs in a $2n$ stage CA, the number of possible connection to generate exhaustive two-pattern tests is 2^n. This number can also be derived by replacing m with $2n$ in Eq. 11 or Eq. 14. □

Example 2 *Consider a* 10 *stage CA (hybrid* 90/150*, null boundary condition). There are 32* (2^5) *choices of tap selections which achieve complete transition coverage. Some of these are* $(1,3,5,7,9)$, $(2,4,6,8,10)$, $(2,3,6,7,9)$, $(1,4,5,8,9)$, ... *(total of 32). Note that all even (or all odd) tap selection for CA achieves complete transition coverage as well.*

4.3 Discussion

Our theorems are able to identify all possible tap selections that obtain the highest achievable transition coverage for both an LFSR and a CA. In contrast, the XLFSR/XLHCA proposed in [20] are only a subset of these selections. Also, the applicability of the XLFSR/XLHCA to pseudo-exhaustive testing of partial dependency circuits was not addressed. In the next section, we will apply our theorems to the design of TPGs for pseudo-exhaustive testing of partial dependency circuits. The benefits of allowing extra flexibility in tap selections will become clear.

The feedback polynomial $P(x)$ for an LFSR need not be primitive. But if the feedback polynomial is primitive, the LFSR can generate the maximal length sequence starting from any non-zero initial state. A total of $2^{2n} - 1$ distinct two-pattern tests can be applied to the CUT without multiple seeding. In contrast, test generation for an LFSR with a non-primitive polynomial is more involved. The LFSR must be first initialized to a non-zero state and clocked until the initial state repeats (because the feedback polynomial is not primitive, the LFSR state must repeat in less than $2^{2n}-1$ clock cycles). The LFSR is then re-initialized to a non-zero state that has not yet occurred. The seeding process is repeated until all non-zero states are covered. Clearly, an LFSR with non-primitive feedback polynomial is much harder to control. Hence, an LFSR with a primitive feedback polynomial is preferred.

For the stuck-at fault model, an n-degree primitive polynomial with the minimum nonzero coefficients is normally chosen as the generator polynomial for an LSFR. For most degrees, an n-stage LFSR with a single XOR suffices to generate all $2^n - 1$ patterns. However, in our proposed two-pattern LFSR design, pre-determined locations in the generator polynomial may require to be nonzero. The resulting generator polynomial may not be primitive. To make it primitive, some other coefficients may need to be assigned nonzero values. In our experience, only one extra nonzero coefficient needs to be assigned in most cases.

The requirement of a primitive feedback polynomial does not pose a problem for a CA. Since the tap selections are independent of the exact rules used, a CA that generates the maximal length sequence with the minimum rule 150 stages can always be chosen. Tables of the maximal length 90/150 CA can be obtained from [17].

In general, a CA provides more tap selection choices with maximal transition coverage than does an LFSR. Also, an LFSR with more terms in the feedback polynomial offers more choices the the one with less terms. However, even if all feedback coefficients are zero, there

still exist (feedback independent) tap selections which attain maximal transition coverage. In pseudo-exhaustive testing, a larger number of tap selection choices normally leads to more a compact TPG design with shorter test length as shown in the following section.

5 Pseudo-Exhaustive Delay Testing

The tap selection rules in the previous section was derived assuming a single output circuit. For an n-input CUT with a single output, the minimum number of stages required to achieve complete transition coverage is $2n$ for both an LFSR and a CA, even though a CA has 2^n possible tap connections, compared to an LFSR which has at most $n+1$ such choices. Unfortunately, this is the necessary result for any linear sequential TPG design if at least one of the primary outputs depends on all the inputs. However, in a PD circuit, the tap selection conditions only need to be satisfied for each cone. Since the inputs of the cones are not disjoint, a TPG which offers larger tap selection choices may meet all these requirements simultaneously, while the one with less choices may not. The fact that a CA has higher number of tap selections to achieve maximal transition coverage makes it a better candidate for two-pattern testing than an LFSR. For some circuits, there may exists an m-stage CA which, when taps are properly assigned, can generate exhaustive two-pattern tests for each cone. But no LFSR with the same number of stages can accomplish the same objective. This can directly impact both TPG area overhead and test time. The following example illustrates this argument.

Example 3 *Consider a 5-input 4-output CUT with the following input/output dependency:*

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

The circuit can be two-pattern pseudo-exhaustively tested by an 8-stage CA with the tap assignments $\{x_1, -, x_2, -, x_4, x_5, -, x_3\}$. *However, at least a 9-stage LFSR (type 1 or type 2) is required to generate pseudo-exhaustive two-pattern tests for the CUT. In this example, a TPG design based on a CA can reduce the test time by half.*

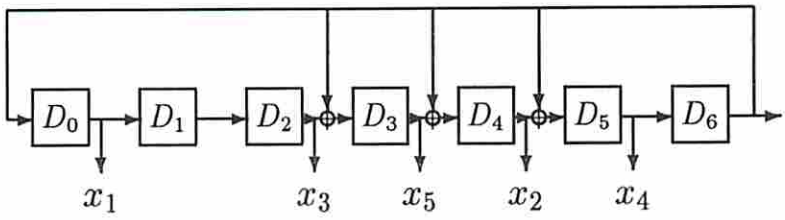

Figure 1: Type 2 LFSR Design for Example 4

5.1 Input Separation

Input separation [15] can be used as a simple tap selection scheme for BIST TPG design. Inputs belong to the same cone are assigned to non-adjacent stages of the TPG. However, in Theorem 1, one incidence of adjacent stages in each cone is allowed, as long as inputs on this cone are not assigned to stage 1 or stage n. This relaxation introduces some freedom in the tap selection and suggests potential reduction in hardware and test time.

Example 4 *Consider the following circuit.*

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

In this circuit, all inputs are incompatible. The TPG designed by the input separation procedure happens to be the worst case of a 10-stage LFSR. But using the necessary and sufficient conditions given in Theorem 1, a solution of a 7-stage LFSR exists as shown in Figure 1. The test time required for pseudo-exhaustive two-pattern testing with the two TPG designs are $2^10 - 1$ *and* $2^7 - 1$ *respectively.*

The design in Figure 1 requires three extra XOR gates at the location 3, 4, and 5. Here, we tradeoff latches with XOR gates. The savings are two folds: (1) an XOR gate requires less area than does a latch; (2) the test length is shorter (2^7 vs. 2^{10} in this example). Even though the transition coverages in the exhaustive case for the two TPG designs are

identical, we conjecture that the fault coverage for a TPG designed by Theorem 1 should rise faster with test length than a TPG designed by input separation. This property will find application in pseudo-random testing, where only a short test sequence is applied. Experimental results presented in section 6 verify our conjecture.

There exists a lower bound on the total number of latches and XOR gates required for a type 2 LFSR that generates pseudo-exhaustive two-pattern tests. Let N_d and N_x denote the number of extra LFSR stages and the number of XOR gates required. Also let N_p denote the minimum number of flip-flops required for input separation.

Theorem 5 *The sum of N_d and N_x is at least N_p, i.e. $N_d + N_x \geq N_p$.*

Proof: The theorem can be proved by contradiction. Assume that $N_d + N_x < N_p$. If the N_x flip-flops are replaced by dummy flip-flops, the configuration satisfies the input separation with only $N_d + N_x$ dummy flip-flops which are less than N_p. This violates the condition that N_p denotes the minimum number of flip-flops required for input separation. □

Input separation is a special case when $N_d = N_p$ and $N_x = 0$. The theorem only gives the lower bound on the sum of N_d and N_x, but does not specify a bound for N_d or N_x. There exist circuits such that $\min(N_d) < N_p$, where $\min(N_d)$ denotes the minimum dummy stages required, but $N_x > N_p - \min(N_d)$. For example, the following circuit has $\min(N_d) = 2$, $N_p = 6$, and $N_x = 5$.

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

5.2 An Algorithm for Delay TPG Design

In the sequel, we assume that a TPG design with complete transition coverage and minimum extra stages N_d is targeted. It is not known whether a lower bound on N_d can be formulated. Even if the minimum N_d can be predicted, the total number of tap selections is given by $\binom{m}{n} n! = \frac{m!}{(m-n)!}$ for an m-stage LFSR TPG and an n-input CUT. This number grows

exponentially even for a moderate sized circuit. Among these tap selections, only a small portion lead to TPG designs that generate complete transition coverage. Identifying these tap selections is computationally difficult for non-trivial cases.

In this section, we describe an algorithm to design a type 2 LFSR based on input separation. An algorithm that uses the necessary and sufficient conditions described above is also presented to design type 2 LFSR TPG with minimum number of stages that achieves complete transition coverage. With slight modification, the algorithm can be applied to a CA TPG design. The algorithm first reduces the search space by removing cones contained in other cones and combining primary inputs into groups. Then, by representing the tap selection rules in Theorem 1 with a state diagram, an *estimation* of the minimum number of stages required is obtained. This number can then be used to make an early decision of terminating a branch that is destined to a non-optimal solution. Unfortunately, the complexity of this algorithm grows exponentially in the worse case. A greedy version of the algorithm is then provided to trade-off optimality with run time. Two definitions are given first, followed by the reduction procedures.

Definition 1 *Input* x_{i_1} *and* x_{i_2} *are swappable if* $x_{i_1} \in cone_j \Leftrightarrow x_{i_2} \in cone_j$.

Definition 2 $g = \{x_i | x_i$ *is a CUT input*$\}$ *is a swappable group if,* $\forall x_i, x_j \in g$, x_i *and* x_j *are swappable.*

Reduction Procedure:

1. Cone containment check: remove cones contained in other cones.
2. Swappable group extraction: combine the primary inputs into swappable groups.

Cones contained in other cones do not contribute any new information to the compatibility relations among primary inputs and thus can be removed. The cone containment check not only reduce the number of cones, but also the number of swappable groups. A swappable group consists of primary inputs that always appear together in the cones. Based on the observation that permutation among members of each swappable group does not affect the transition coverage, the swappable group extraction effectively reduces the enumeration size. The order of the two procedures is important. The cone containment check must precede the swappable group extraction. The reduction procedures are clarified by the following example.

Example 5 *Consider a circuit with the following cone dependencies:*

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

The cone containment check will remove cone 1 and 5, which are contained in cone 2 and 4 respectively. The primary inputs can be combined into four swappable groups: $g_1 = \{x_1, x_2\}, g_2 = \{x_3, x_4\}, g_3 = \{x_5, x_6\}$, *and* $g_4 = \{x_7\}$. *The cone dependencies can then be re-written in a more compact form.*

$$\begin{aligned} y_2 &= f_2(g_1, g_2) \\ y_3 &= f_3(g_2, g_3) \\ y_4 &= f_4(g_3, g_4) \end{aligned}$$

Note that, if the cone containment check was not performed, only x_1 *and* x_2 *can be combined into a swappable group.*

One special case is when an output depends on all inputs. In this case, the cone dependencies are reduces to a single output with a single swappable group consists of all inputs. The algorithm will then quickly produce a $2n$-stage LFSR that satisfies Theorem 1.

The tap selection rules for an LFSR to have complete transition coverage can be presented by a state diagram as shown in Figure 2. For a PD circuit, a state variable S_j is maintained for each cone y_j. Each assignment of a new input x_i to an LFSR stage corresponds to a 1-transition (0-transition) in the state diagrams if $x_i \rightsquigarrow y_j$ ($x_i \not\rightsquigarrow y_j$). Starting from state 1, a tap at stage 1 or two consecutive taps between stage 2 and $n-1$ create a state transition to state 2. From then on, two consecutive taps or a tap at stage n leads to the illegal **stop** state. The state diagram clearly follows the semantics of Theorem 1.

Example 6 *Consider a 5-input circuit with the tap selection* $\{x_1, -, x_3, x_5, x_2, x_4, -\}$. *The state transitions for the cone* $\{x_3, x_4, x_5\}$ *are* $\{0, 0, 1, 1, 0, 1, 0\}$. *These transitions are marked in the state diagram of Figure 3. The underlined numbers denote the transition steps.*

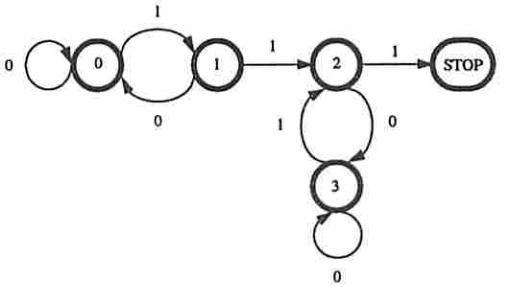

Figure 2: A State Diagram Representation for a Type 2 LFSR

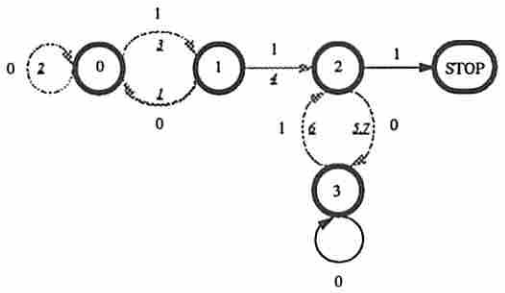

Figure 3: Example of State Transition for a Type 2 LFSR

state	minimum stage required
0	$2i-1$
1	$2i$
2	$2i+1$
3	$2i-1$

Table 1: Estimation of minimum stages required

```
DELAY-TPG(M)
    CONE-REDUCTION(M)
    for each primary output y_j do
        state[y_j] ← 1
        min_est[y_j] ← 2k_j
    PERM(0)


PERM(s)
    if all primary inputs are assigned then
        previous_best ← s
        PRINT-RESULT(tap)
        return
    fb_list ← {g_i | state[y_j] = 2 and g_i ⇝ y_j}
    for each group g_i do
        if g_i ∉ fb_list and usage_cnt[g_i] ≠ 0 then
            cell_req ← max_j {min_est[y_j]}
            if s + cell_req < previous_best then
                for each primary output y_j do
                    if g_i ⇝ y_j then TRANSITION-1(state, min_est)
                    else TRANSITION-0(state, min_est)
                usage_cnt[g_i] = usage_cnt[g_i] - 1
                tap[s] ← g_i
                PERM(s + 1)
```

Figure 4: Pseudo-code for delay TPG design

Let i be the number of unassigned inputs (not groups) in a cone. For each state in the state diagram, an estimation of the minimum TPG stages required for the cone is shown in Tables 1. Since a TPG design with minimum stages is targeted, a branch is terminated if the summation of the current number of LFSR stages and the estimated minimum stages required exceeds the previous best solution.

The pseudo-code of the algorithm to find the optimal tap selection achieving complete transition coverage for a type 2 LFSR is given in Figure 4. Procedure DELAY-TPG takes as inputs a linked list M of cone structures, each with a pointer to a linked list of inputs that feed the cone. The reduction procedures are first applied to reduce the search space. An array *usage_cnt* is kept for each swappable group to record the number of members in

State(min. est.)

	1	-	2	4	-	3	5	-
y_1 1(6)	2(5)	3(4)	2(3)	3(2)	3(2)	2 (1)	3(0)	3(0)
y_2 1(6)	0(5)	0(5)	1(4)	2(3)	3(2)	2 (1)	3(0)	3(0)
y_3 1(6)	0(5)	0(5)	0(5)	1(4)	0(3)	1 (2)	2(1)	3(0)
y_4 1(6)	2(5)	3(4)	3(4)	2(3)	3(2)	3 (2)	2(1)	3(0)
y_5 1(4)	0(3)	0(3)	1(2)	0(1)	0(1)	0 (1)	1(0)	0(-1)

Table 2: Calculation of the algorithm DELAY-TPG

State(min. est.)

	1	-	3	5	2	4	-
y_1 1(6)	2(5)	3(4)	2(3)	3(2)	2(1)	3 (0)	3(0)
y_2 1(6)	0(5)	0(5)	1(4)	0(3)	1(2)	2 (1)	3(0)
y_3 1(6)	0(5)	0(5)	1(4)	2(3)	3(2)	2 (1)	3(0)
y_4 1(6)	2(5)	3(4)	3(4)	2(3)	3(2)	2 (1)	3(0)
y_5 1(4)	0(3)	0(3)	0(3)	1(2)	2(1)	3 (0)	3(0)

Table 3: Calculation of the Greedy Algorithm

each group. A dummy stage is also treated as a swappable group with $usage\_cnt = n$. The state machine for each primary output y_j starts with the state 1 and the minimum stage estimation $2k_j$, where k_j is the number of inputs in $cone_j$. The algorithm then calls PERM to exhaustively enumerate all permutation of the inputs. The recursive routine PERM first calculates *fb_list*, the incompatible input set of the previous tap. If $state[y_j]$ equals to 2, then the next tap can not be assigned to inputs belonging to $cone_j$. Therefore, the incompatible input set *fb_list* is simply the union of the swappable groups g_i's such that $state[y_j] = 2$ and $g_i \rightsquigarrow y_j$. The updates of *state* and *min_est* follow Figure 2 and Table 1 respectively. A search branch is terminated when the sum of the number of current assigned stages and minimum stages required exceeds the previous best solution.

The complexity of this algorithm is exponential in the worst case. A greedy version can be constructed. Based on the observation that *min_est* decrements when the *state* of y_j changes, the greedy algorithm tries to assign the next stage to an input such that the highest *min_est* decrement by 1. If such an assignment does not exist or a tie occurs, then the assignment that decrements $\sum_j min\_ets[y_j]$ the most is chosen. The routine PERM, modified for the greedy version, is shown in Figure 5.

Example 7 *Consider the circuit in Example 4. The calculation to find the first solution*

```
PERM(s)
    if all primary inputs are assigned then
        previous_best ← s
        PRINT-RESULT(tap)
        return
    fb_list ← {g_i | state[y_j] = 2 and g_i ⇝ y_j}
    for each group g_i do
        if g_i ∉ fb_list and usage_cnt[g_i] ≠ 0 then
            cell_req ← max_j {min_est[y_j]}
            hc_list ← {y_j | min_est[y_j] = cell_req}
            if s + cell_req < previous_best then
                new_min_est ← UPDATE(min_est)
                Δ_i ← Σ_{y_j ∈ hc_list} (new_min_est[y_j] − min_est[y_j])
                Γ_i ← Σ_{y_j ∉ hc_list} (new_min_est[y_j] − min_est[y_j])
    Δ_max ← max_i {Δ_i}
    mx_list ← {g_i | Δ_i = Δ_max}
    Γ_max = max_{g_i ∈ mx_list} {Γ_i}
    gr_list ← {g_i ∈ mx_list | Γ_i = Γ_max}
    for each primary output y_j ∈ gr_list do
        if g_i ⇝ y_j then TRANSITION-1(state, min_est)
        else TRANSITION-0(state, min_est)
    usage_cnt[g_i] = usage_cnt[g_i] - 1
    tap[s] ← g_i
    PERM(s + 1)
```

Figure 5: A Greedy Algorithm for delay TPG design

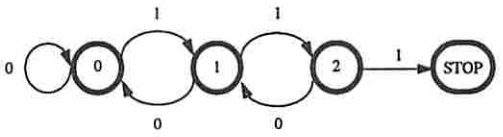

Figure 6: A State Diagram Representation for a CA

are summarized in Table 2. An entry (j, s) *in the matrix represents the state of cone* j *after TPG stages 1 to* s *have been assigned the CUT inputs on the title row. The number in the parenthesis is the minimum additional stages* min_est *required for this cone. Note that* min_est[j] *decrements when the state of cone* j *changes.*

The calculation to find the first solution for the greedy algorithm is shown in Table 3. The solution is also the optimal one with only 2 extra flip-flops. The greedy algorithm is able to find the best solution in a single pass for this example.

The algorithms can be easily modified for input separation. If the initial states for each primary output y_j is 3 instead of 1, then the algorithm calculates tap selections for input separation.

For a CA, a similar state diagram can be constructed that follows the semantic of Theorem 3. The state digram for a CA is shown in Figure 6. The same table in Table 1 is used to estimate the stages required. Because there are only state 0, 1, and 2 in Figure 6, we can simply ignore the estimate for state 3 in Table 1. The same algorithms in Figure 4 and Figure 5 can be applied to a CA without modification.

6 Experimental Results

The algorithms in the previous section had been implemented and tested on selected circuits in the ISCAS 89 [3] benchmark circuits. The input/output dependencies of the combinational part of each circuit were extracted and served as inputs to the program. The total number of flip-flops (including the extra ones) of the final TPG were generated. Statistics of these circuits and the experiment results are shown in Table 4. The TPG designs based on input separation and the necessary and sufficient conditions derived in Theorem 1 for type 2 LFSR and in Theorem 3 for CA are presented. In all case, the number of flip-flops required for TPG deigns based on a CA is no more than that based on the optimal LFSR, which in turn uses no more flop-flops than input separation.

ckt	#pi	#si	#po	k	inp sep		LFSR		CA	
					#reg	#exff	#reg	#exff	#reg	#exff
s298	3	14	6	8	21	4	19	2	18	1
s344	9	15	11	13	28	4	26	2	26	2
s349	9	15	11	13	28	4	28	4	28	4
s382	3	21	6	14	28	4	28	4	28	4
s444	3	21	6	14	28	4	28	4	28	4
s526	3	21	6	14	28	4	28	4	28	4

Table 4: Experimental Result

For a TPG design with more than, say, 20 inputs, it is impractical to apply the complete test patterns. The TPG designed by these algorithms can be used to apply a test sequence of *reasonable* length. In section 5.1, we conjectured that a TPG design based on Theorem 1 must have a sharper rise in the fault vs. test length curve. We now verify our conjecture with experiments. Three TPG designs based on input separation and our selection rules (LFSR and CA) for `s298` are generated. A random pattern generator generates the desired test patterns with tapped stage outputs stored in a different file. A robust path delay fault simulator is then used to calculate the tested paths. The result is shown in Figure 7. Also included for comparison is the fault coverage obtained by an n-stage LFSR. The curve for a CA has the sharpest rise among all three because it requires fewer number of flip-flops and has better randomness. A TPG design with fewer stages is only necessary to obtain high fault coverage, but it is not sufficient. The taps must be carefully selected to remove linear dependency inherent in consecutive TPG stages.

7 Conclusion

An important dimension which distinguishes BIST TPG design issues from DFT techniques for two-pattern testing such as scan chain ordering [5, 12, 15] is the test time and hardware tradeoff among solutions that acquire the same fault coverage. In BIST TPG, the test pattern pairs are generated randomly and it is not possible to apply carefully-selected tests. Hence, the impact of the design on test time can be tremendous. For example, for `s298`, the test sequence length for a 19-stage LFSR TPG may be close to 2^{19}, while for an 18-stage CA, 100% robust path delay fault coverage is guaranteed in 2^{18} vectors. In DFT techniques, since an ATPG program can be used to derive specific tests for the faults, which are applied via

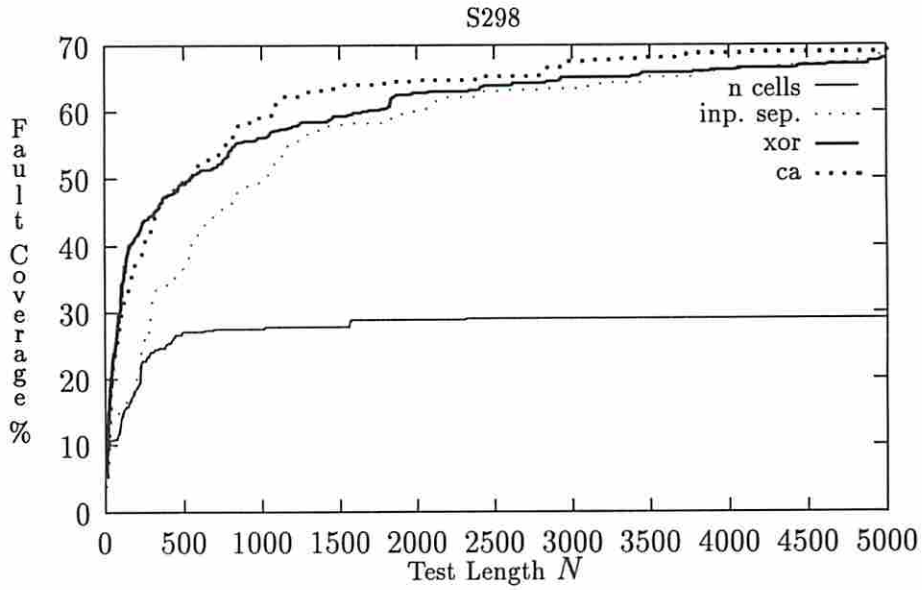

Figure 7: Fault Coverage vs. Test Length for S298

scan chain using a tester, the impact on test length may be minimal. Hence, the objective in scan design is to merely minimize hardware which can achieve a desired fault coverage. On the other hand, BIST TPG design may use extra hardware to keep test time at acceptable levels. It should also be noted that in some cases, introduction of XOR gates can help reduce the number of 'dummy' flip-flops required. Hence, CA and LFSRs have the potential to provide scan chain configurations which require lower area than simple scan chains. Such structures are currently under investigation.

Design of LFSR/CA test pattern generators suitable for two-pattern testing, is the main subject of this paper. Necessary and sufficient conditions for TPG tap selection which guarantee maximal transition coverage have been derived. LFSRs with primitive feedback polynomials, which have many even powers of x, have been shown to be more suitable as TPGs for exhaustive two-pattern testing. In general, LFSRs with greater number of XOR gates are shown to be better TPGs for two-pattern testing. Further, cellular automata are shown to be better TPGs than LFSRs for two-pattern testing, independent of their feedback rules.

The above results identify all TPGs which maximize transition coverage. Structures such as XLFSR and XLHCA [20] can be shown to a subset of our results. Experimental results on some benchmark circuits show that for a given constraint on the TPG size, the

TPGs designed by using the results derived in this paper, generate test patterns which provide much higher fault coverages than other TPGs.

These results provide the basic theory to solve many practical problems in BIST for two-pattern testing. Most real circuits have multiple outputs and, in many cases, none of the outputs depend on all the circuit inputs. In such cases, two pattern pseudo-exhaustive testing concepts can help reduce the test sequence length and TPG hardware complexity, without reducing fault coverage and test time. Algorithms to design efficient TPGs for PD circuits have been presented. The focus of this work has been on designing TPGs to maximize transition coverage. Maximization of transition coverage maximizes the number of distinct two-pattern tests applied to the CUT if all TPG states (non-zero) are traversed. However, typically high fault coverage can be obtained by using such TPGs for much shorter test lengths. The TPGs designed using the theory derived above are well suited for such efficient two-pattern testing.

There is a need to study test application methodologies essential for detection of the two-pattern testable faults (e.g. robust path delay tests require that test patterns be applied using a combination of slow and fast clocks). These and other such practical issues pertaining to test application need to be studied.

Acknowledgement

The authors wish to thank Dr. A. K. Pramanick (IBM) and Prof. S.M. Reddy (University of Iowa) for providing the path delay fault simulator.

References

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

[2] P. H. Bardell. "Analysis of Cellular Automata as Pseudorandom Pattern Generators". In *Proceedings IEEE International Test Conference*, pages 762–767, 1990.

[3] F. Brglez, D. Bryan, and K. Kozminski. "Combinational Profiles of Sequential Benchmark Circuits". In *Proceedings IEEE International Symposium on Circuits and Systems*, May 1989.

[4] C.-A. Chen and S. K. Gupta. "BIST Test Pattern Generators for Stuck-Open and Delay Testing". Technical Report CENG 93-34, University of Southern California, 1993.

[5] K.-T. Cheng, S. Devadas, and K. Keutzer. "A Partial Enhanced-Scan Approach to Robust Delay-Fault Generation for Sequential Circuits". In *Proceedings IEEE International Test Conference*, pages 403–410, 1991.

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

[7] W. Feller. *"An Introduction to Probability Theory and Its Applications"*, volume 1. John Wiley & Sons, 3rd edition, 1968.

[8] K. Furuya and E. J. McCluskey. "Two-Pattern Test Capabilities of Autonomous TPG Circuits". In *Proceedings IEEE International Test Conference*, pages 704–711, Oct. 1991.

[9] S. W. Golomb. *"Shift Register Sequences"*. Aegean Park Press, Laguna Hills, CA, 1982.

[10] A. Krasniewski and S. Pilarski. "Circular Self-Test Path: A Low-Cost BIST Technique". In *Proceedings IEEE-ACM Design Automation Conference*, pages 407–415, June 1987.

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

[12] W. Mao and M. D. Ciletti. "Arrangement of Latches in Scan-path Design to Improve Delay Fault Coverage". In *Proceedings IEEE International Test Conference*, pages 387–393, 1990.

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

[14] S. Pilarski and A. Pierzynska. "BIST and Delay Fault Detection". In *Proceedings IEEE International Test Conference*, pages 236–242, October 1993.

[15] J. Savir and R. Berry. "At-Speed Test Is Not Necessarily an AC Test". In *Proceedings IEEE International Test Conference*, pages 722–728, 1991.

[16] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller. "The Analysis of One-Dimensional Cellular Automata and Their Aliasing Properties". *IEEE Trans. on CAD*, 9(7):767–778, July 1990.

[17] T. Slater and M. Serra. "Tables of Linear Hybrid 90/150 Cellular Automata". Technical Report DCS-105-IR, Department of Computer Science, University of Vicoria, Victoria BC, Canada, 1990.

[18] G. L. Smith. "Model for Delay Faults Based on Paths". In *Proceedings IEEE International Test Conference*, pages 342–349, 1985.

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

[20] 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.