On the Generation of Pseudo-Deterministic Two-Patterns Test Sequence withLFSRs

Christian DUFAZA and Yervant ZORIAN

LIRMM
UMR CNRS-UM2, 161 Rue Ada
34392 Montpellier, FRANCE
email: dufaza@lirmm.fr

Logic Vision, Inc.
101 Metro Drive
San Jose, CA, USA, 95110
zorian@lvision.com

Abstract

Many Built-In Self Test pattern generators use Linear Feedback Shift Registers (LFSR) to generate test sequences. In this paper, we address the generation of deterministic pairs of patterns for delay faults testing with LFSRs. A new synthesis procedure for a n-size LFSR is given and guarantees that a deterministic set of n precomputed test pairs is embedded in the maximal length pseudo-random test sequence of the LFSR. Sufficient and necessary conditions for the synthesis of this pseudo-deterministic LFSR are provided and show that at-speed delay faults testing becomes a reality without any additional cost for the LFSR. Moreover, since the theoretical properties of LFSRs are preserved, our method could be beneficially used in conjunction with any other technique proposed so far.

1 Introduction

Nowadays, integrated circuits with millions of transistors are common. The test of such circuits has become a challenge, since it drives the reliability of complete electronic products and impacts upon their final quality. Unfortunately, the cost of testing increases significantly with the complexity of these circuits. Hence, different test strategies must be considered in a common pool to find optimal solutions. It is now widely admitted that extensive investments in Automatic Test Equipment (ATE) are no longer sufficient to address the testing problems. Instead built-in structures are required to help solve this problem. Solutions such as Built-In Self Test (BIST) appear to be very promising for large classes of products [1–2]. BIST is well-know for its numerous advantages. For instance, BIST allows at-speed test of modules and high fault coverages using test point insertion techniques [3]. It may also provides some on-line test features [4]. BIST requires on-chip Test Pattern Generators (TPGs) as well as Output Response Analyzers (ORAs). The TPGs can be classified in two families: scan based TPGs and at-clock speed TPGs. The principle of scan based TPGs is to load test patterns serially into scan chains. Usually, the TPG is an LFSR, sometimes with reseeding or multiple polynomial capabilities [5], or bit-flipping function with XOR gates [6]. At-clock speed TPGs do not suffer from the long serial loading of scan based techniques since they provide test patterns directly at the inputs of each Circuit Under Test (CUT). Many approaches to improve fault coverage have been shown in this area: [7] presents the insertion of additional logic to switch some random patterns into deterministic ones, [8] does the same but for all the patterns, [9] gives the optimal seed of an LFSR, [10] presents a mixed scheme based on LFSR reconfigurations and [11] suggests the use of counters to generate test patterns. These are some examples.

The above approaches have two points in common: the first is that they address only the detection of stuck-at faults; the second one is that theyare based onLFSR structures with certain augmentations. So, the importance of LFSRs in BIST solutions has been widely established. For these reasons, the purpose of this paper is to analyze the capability of LFSRs and to demonstrate its generation of deterministic pairs of patterns, for the detection of faults such as delay and stuck-open.

Since the synthesis constraints for TPGs are more complex for two-patterns tests than that for single ones more sophisticated solutions have been proposed in the literature. In fact, detecting a delay fault or a stuck-open fault requires the application of two distinct but successive test patterns. The first one is called the initialization vector and isfollowed by the propagation vector; the circuit outputs are finally sampled at clock-speed for the observation of the fault [12]. Moreover, robust or nonrobust tests can be investigated. A nonrobust test detects a delay fault only if non arbitrary additional delay faults occurs at the same time in the CUT; a robust test is not sensitive to these elements. Additionally, two delay fault models are proposed in the literature. The gate delay fault model considers a single failure at a faulty gate [13] while the path delay fault model considers that the failure caused by process variations is distributed over a path [14]. Similar to stuck-

ED&TC '97 on CD-ROM

at faults, the simplest approach to detect delay faults is an exhaustive two-pattern test. For an n-input CUT, TPGs must run through all the $2^n \times (2^n-1)$ different possible pairs of patterns. For instance [15] gives an approach dedicated to datapaths that uses a counter and an accumulator for the exhaustive generation of the pairs of patterns. To circumvent the prohibitive test time, an adjacency test method, based on the switching of a single bit in the test pair, has been proposed in [16]. It requires only $n \times (2^n - 1)$ pair of patterns. In [17] and [18] a Multiple Input Shift Register (MISR) is reseeded several times with a constant parallel input vector. Using an ATPG based selection algorithm for the determination of the optimal input vector, the MISR runs through its maximal length sequence and provides a maximal robust path delay fault coverage for a substantial decrease in the test sequence length [18]. Similar to stuck-at fault testing, these approaches can be classified as exhaustive and pseudo-exhaustive TPGs for delay faults. Another important category of TPGs for delay faults is composed of pseudo-random architectures. For this purpose, the notion of transition fault coverage has been introduced in [19]. This qualifies the efficiency of pseudo-random TPGs to generate transitions. Numerous studies compare LFSR and Cellular Automata (CA) efficiencies for delay faults testing [20][21]. They lead to new proposals: XLFSR-XLCA [22], GLFSR [23] and circular self-test path [24]. For instance, in [25] necessary and sufficient conditions to ensure complete/maximal transition coverage for LFSR and CA are derived. Again, similar to stuck-at faults testing, weighted pseudo-random testing has been also proposed for delay faults [26]. The third category of "pure" deterministic TPGs for delay faults testing is still an empty set since its silicon area overhead is prohibitive. Concerning the most promising approach of pseudo-deterministic TPGs for delay faults testing, few proposals have been reported and remain in numerous cases possible extensions of the stuck-at fault case [27][10].

Our contribution in this paper is to address this issue by proposing a synthesis method for pseudo-deterministic LFSRs with at-clock speed of delay faults testing. For a n-size LFSR, our technique allows the generation of n deterministic pairs of test patterns embedded in a classical pseudo-random test sequence. Since this value of n is low in some cases and may appear restrictive, it must be mentioned that all the basic properties of LFSR: primitive polynomial, maximal sequence, etc, are preserved in our algorithm and consequently, any other TPG technique using LFSR can be beneficially applied in conjunction with our proposal. Moreover, for our TPG no extra logic is required (except additional inputs at the XOR gates of the LFSR) and, to our best knowledge this is the first attempt to detect delay faults with a n-stages LFSR while n is the number of primary inputs of the CUT.

This paper is organized as follows: in section 2 the necessary background on LFSR is presented. In section 3, the synthesis method for pseudo-deterministic LFSR is developed. Section 4 discusses some of the limitations and possible improvements of the methodology and compares our technique with other proposals. Finally, concluding remarks and future research directions are presented in section 5.

2 Similar Matrices of LFSRs

Consider n-tuples of 0's and 1's into a field; that is all the possible combinations of 0 and 1 with n bits. Such a set of elements is called a Galois Field of characteristic 2 and of order 2^n, abbreviated GF(2^n) [28]. These n-tuples of 0's and 1's are easy to manipulate using digital circuitry and can be stored in a row of n storage elements called a register. Using also half-adder or exclusive-OR gate (XOR), we obtain an LFSR. LFSRs have been widely described [28-29] and they are basically known in their two canonical forms; the standard and the Shift Division Circuit (SDC) forms. The standard form of a LFSR of size n is schematically represented in figure 1 and its corresponding transition matrix T_0 is on the form:

$$T_0=\begin{bmatrix} 0 & 0 & 0 & \dots & c0 \\ 1 & 0 & 0 & \dots & c1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & c_{n-2} \\ 0 & 0 & 0 & \dots & c_{n-1} \end{bmatrix} \text{ with } V_{T0,t+1}=V_{T0,t}\times T_0 \text{ (Eq. 1)}$$

With $V_{T0,t}$ as the current state of an LFSR and T_0 as its transition matrix, the next state of the LFSR $V_{T0,t+1}$ is completely defined by equation 1.

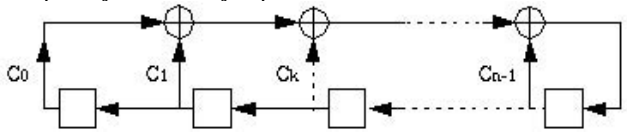

Figure 1: Standard form canonical LFSR

The second canonical form of an LFSR is represented in figure 2 and is also called a Shift Division Circuit (SDC). In this case, the corresponding transition (or companion) matrix T_c of p(x) is on the form:

$$T_c=\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \\ c0 & c1 & c2 & \dots & c_{n-1} \end{bmatrix}, \text{ with } V_{Tc,t+1}=V_{Tc,t}\times T_c \text{ (Eq. 2)}$$

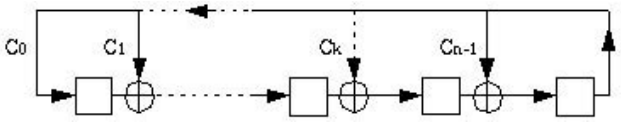

Figure 2: Shift Division Circuit, 2nd canonical form LFSR

So, we can associate an LFSR with its characteristic

polynomial $p(x) = c_0x^0 + c_1x^1 + ... + c_nx^n$. Considering a primitive polynomial p(x) for the characteristic matrix T_c, it is well known that its corresponding transition states graph is of maximal length and equal to (2^n-1) [28]. However, the matrix T_c defines only one switching network for this state graph or equivalently corresponds to one possible state assignment. In fact, for a given state graph several networks can be found. Once a states assignment is made, the state graph may be replaced by its transition matrix; up to now we have just consider T_c. So, in our case several linear switching networks can provide exactly the same properties than the SDC - LFSR except for the state assignments. Let us now consider a linear mapping over $GF(2^n)$, such that:

$\sigma : V_{Tc,t} \rightarrow V_{T,t} = V_{Tc,t} \times B$

The linear mapping is totally defined by the nonsingular n×n square matrix B. In this new base for $GF(2^n)$, we can establish as for equation 2, the next state equation for $V_{T,t}$:

$V_{T,t+1} = V_{T,t} \times T$ (Eq. 3)

Now, applying the linear mapping σ to $V_{Tc,t}$ and $V_{Tc,t+1}$, one can easily obtain: $T = B^{-1} \times T_c \times B$ or equivalently

$T_c = B \times T \times B^{-1}$ (Eq. 4)

T and T_c are called similar matrices and have been studied concerning linear algebra. Many theorems exist on similar matrices, the most important show that all the properties concerning the matrix T_c apply also to matrix T. In our case, it means for instance that T_c and T have the same characteristic polynomial p(x), then the same length for the transition state graph, etc, except that their states assignments are different. Then, the matrix T characterizes a new switching network for an LFSR with the same properties than their two canonical forms T_c and T_0. However, each j^{th} column of this matrix T provides the linear logic equation for the j^{th} register cell of the LFSR and can contain a larger number of "1" than the corresponding columns of T_c or T_0 (a "1" means an additional input for the XOR gate at this stage). So, at this point it must be mentioned that the matrix T represents rather a Linear Finite State Machine (LFSM) than one of the two canonical forms of LFSRs. However, for clarity in the following, we call it also LFSR.

3 Pseudo-Deterministic LFSRs

In this section, we develop a new synthesis technique for LFSR that guarantees a precomputed set of n pairs patterns are embedded in the maximal length sequence generated by this structure. Moreover, since the maximal length sequence of the LFSR is preserved, the test methodology assumed is a pseudo-deterministic test and then both pseudo-random and deterministic patterns are generated. The subset of the deterministic pairs of patterns addresses the remaining hard to detect faults that are resistant to the pseudo-random test. Since the LFSR is ran at-clock speed, actual delay faults can be detected by this technique.

3.1 Overview of the synthesis method

Let us consider first the nature of the test sequence to generate. We assume that our method is not able to deal with more than n pairs of patterns; n is the size of the synthesized LFSR or the number of its registers. So, we consider no more than n pairs of test patterns that have been pre-computed with an ATPG tool for delay faults. These pairs of patterns will be noted as: P1={V1.1;V1.2}, P2={V2.1;V2.2}, ... to Pn={Vn.1;Vn.2}

For any $1 \le k \le n$, Vk.1 will correspond to the first patterns of the pair $k^{th,}$ Vk.2 will correspond to the second pattern of the pair. Since these pairs of patterns are not correlated with each other and a specific order of test application is not required between each pair, one assumes that any other dummy patterns or states of the LFSR can be inserted between each pair. Let's call Di these sub-sequences of non addressed states of the LFSR; Di are sub-sequences of different lengths and not only single pattern. Consequently, the sequence generated by the LFSR is of the form:

$$S = \begin{bmatrix} D1 \\ P1 \\ D2 \\ P2 \\ Dk \\ Pk \\ Dk+1 \\ Pn \\ Dn+1 \end{bmatrix} = \left[[x] \begin{bmatrix} V1.1 \\ V1.2 \end{bmatrix} [x] \begin{bmatrix} V2.1 \\ V2.2 \end{bmatrix} [x] \begin{bmatrix} Vk.1 \\ Vk.2 \end{bmatrix} [x] \begin{bmatrix} Vn.1 \\ Vn.n \end{bmatrix} [x] \right]^t,$$

with [X] denotes sub-sequences Di of any length in the matrix expression, and $[\ ]^t$ the transpose matrix.

Example: Let us consider the sequence :

$$S = \left[[x] \begin{bmatrix} 0101 \\ 1100 \end{bmatrix} [x] \begin{bmatrix} 1100 \\ 1110 \end{bmatrix} [x] \begin{bmatrix} 0010 \\ 1000 \end{bmatrix} [x] \begin{bmatrix} 1111 \\ 0011 \end{bmatrix} [x] \right]^t$$

Now, we have seen in section 2.1 that a LFSR satisfies equation 4: $V_{T,t+1}=V_{T,t} \times T$. If we consider the same relation complete applied to the sequence S, we obtain the following equation:

$S_{t+1}=S_t \times T$ (Eq. 5) or $\left[[x] \begin{bmatrix} V1.2 \\ x \end{bmatrix} [x] \begin{bmatrix} V2.2 \\ x \end{bmatrix} [x] \begin{bmatrix} Vk.2 \\ x \end{bmatrix} [x] \begin{bmatrix} Vn.2 \\ x \end{bmatrix} [x] \right]^t =$

$$\left[[x] \begin{bmatrix} V1.1 \\ V1.2 \end{bmatrix} [x] \begin{bmatrix} V2.1 \\ V2.2 \end{bmatrix} [x] \begin{bmatrix} Vk.1 \\ Vk.2 \end{bmatrix} [x] \begin{bmatrix} Vn.1 \\ Vn.n \end{bmatrix} [x] \right]^t \times T$$

Since we need that the successors of the Vk.1 patterns be the Vk.2 patterns, it implies that the S_{t+1} matrix has the indicated form. S_t and S_{t+1} are matrices of sizes n columns by l rows. Moreover, since the number of rows corresponds to the number of patterns; that is the pairs of test patterns plus the "dummy" patterns, it implies that l is at least equal to (2×n).

Example: From the previous example, we obtain:

$$S_{t+1}=S_t \times T = \left[[x] \begin{bmatrix} 1100 \\ x \end{bmatrix} [x] \begin{bmatrix} 1110 \\ x \end{bmatrix} [x] \begin{bmatrix} 1000 \\ x \end{bmatrix} [x] \begin{bmatrix} 0011 \\ x \end{bmatrix} [x] \right] =$$

$$\left[[x] \begin{bmatrix} 0101 \\ 1100 \end{bmatrix} [x] \begin{bmatrix} 1100 \\ 1110 \end{bmatrix} [x] \begin{bmatrix} 0010 \\ 1000 \end{bmatrix} [x] \begin{bmatrix} 1111 \\ 0011 \end{bmatrix} [x] \right]^t \times T$$

Our goal is to determine the transition matrix T of the LFSR. The following equation gives us the solution for T: $S_t^{-1} \times S_{t+1} = T$ (Eq. 6). S_t^{-1} is the left pseudo-inverse of S_t. If S_t is a (n×l) matrix, its left pseudo-inverse S_t^{-1} is a (l×n) matrix, such that $S_t^{-1} \times S_t = I_N$; I_N identity matrix of order n. More details on generalized pseudo-inverses can be found in [34]. Assuming that the matrix S_t contains at least n linearly independent vectors, the pseudo-inverse S_t^{-1} is always computable. Moreover, if we assume that these n linearly independent patterns chosen in S_t correspond to the Vk.1 first patterns of the detecting pairs, the pseudo-inverse matrix S_t^{-1} will always have the following form: $S_t^{-1} = [[x0]\ VI1.1\ [x0]\ VI2.1\ [x0]\ VIk.1\ [x0]\ VIn.1\ [x0]]$

VIk.1 are 1 column vectors of n row sizes and are the corresponding solutions of the pseudo-inverse calculation for the respective patterns Vk.1 of the initial S_t matrix. For all the other dependent rows of S_t, it can be observed that the corresponding columns of the S_t^{-1} matrix are null columns; [x0] denotes to these null columns.

Example: we obtain $S_t^{-1} = \begin{bmatrix} [x0] \begin{bmatrix}1\\1\\0\\0\end{bmatrix} [x0] \begin{bmatrix}0\\1\\0\\1\end{bmatrix} [x0] \begin{bmatrix}1\\1\\1\\1\end{bmatrix} [x0] \begin{bmatrix}1\\1\\0\\1\end{bmatrix} [x0] \end{bmatrix}$ with $S_t^{-1} \times S_t = I_4$.

Consequently, for the $S_t^{-1} \times S_{t+1} = T$ product, only the rows of S_{t+1} which index row k do not corresponds to a null column of S_t^{-1} will participate to the result for the matrix T. Exactly, in the S_{t+1} matrix, these rows correspond to the 2nd patterns of the detecting pairs, that is Vk.2. According to this, the previous equation 6 can be simplified by the following equation:

$S'^{-1}_t \times S'_{t+1} = T$ (Eq .7)

S'_t and S'_{t+1} are then (n×n) square. S'_t is composed only of n linear independent vectors of width n, and moreover we have assumed that these vectors are the Vk.1 vectors. Consequently, S'_t is invertible and gives S'^{-1}_t another square (n×n) matrix. Also, T is a square (n×n) matrix but its rank still depends on the content of the matrix S'_{t+1}.

Example: for $S'_t = \begin{bmatrix}0101\\1100\\0010\\1111\end{bmatrix}$, we obtain:

$$S'^{-1}_t \times S'_{t+1} = T = \begin{bmatrix}1011\\1111\\0010\\0111\end{bmatrix} \times \begin{bmatrix}1100\\1110\\1000\\0011\end{bmatrix} = \begin{bmatrix}0111\\1001\\1000\\0101\end{bmatrix}$$

The reader can easily verify that the LFSR of characteristic matrix T allows the generation of the 4th pairs of patterns, P1 to P4. However, for this value of T, the cycle of the LFSR is not unique and not of maximal length. With such a technique, we argue that the n pairs of test patterns Pi will be included in the sequence generated by the synthesized LFSR of transition matrix T given by equation 7, if the following criteria are satisfied:

Criteria 1: The set of n first patterns of the test pairs form an invertible (n×n) size matrix.

Criteria 2: The transition matrix of the synthesized LFSR has a rank equal to n.

Criteria 3: The characteristic polynomial of the synthesized LFSR is primitive.

These three criteria lead to the conclusion that the synthesized LFSR has also a primitive polynomial and consequently will generate a maximal length sequence. Then, any maximal length sequence contains also all the Vk.1 first patterns of the deterministic pairs to generate. Also, the LFSR has been synthesized to satisfy the n transitions from Vk.1 to Vk.2 and then we can conclude that it will generate the n pairs of deterministic patterns P1 to Pn. In the following, we will develop conditions and techniques that corresponds to the initial assumptions and that guarantee each of these three criteria.

3.2 Linear dependency of 1st pair patterns

Criteria 1 requires that the n first patterns of the n considered transitions form an invertible matrix S'_t. If it is not naturally the case and the linear independence of the first pair patterns cannot be satisfied, we can use the following technique to gaurantee an invertible S'_t matrix. The basis of this technique is to add log$_2$w extra bits to solve the dependencies; w is the order of the dependencies.

Example: let us consider the previous example and replace the past last vector v1.4=[1111] by the new vector v1.4=[1110]. The reader can easily verify that V1.4 is now a linear combination of the V1.2 and V3.1 vectors and that the new matrix S'_t is equals to: $S'_t = \begin{bmatrix}0101\\1100\\0010\\1110\end{bmatrix}$.

If we add an extra bit $x_{45} \in \{0,1\}$ to the V1.4 vector, the size of the LFSR will be now equals to n=5 and V1.4=[1110x_{45}]. Since V1.4 was before a linear combination of the vectors V2.1 and V3.1, its implies that the linear sum of the 5th bits added to each of these vectors (vectors from which the linear dependency comes), must be equal to $\overline{x}_{45}$. Obviously in this case, since the number of patterns from which the linear dependency comes from is odd, the 5th bits of the vectors V2.1 and V3.1 could not be equal. Consequently, two solutions exist for the new vectors V2.1 and V3.1; they are Sol1 = {V2.1=[1100x_{45}], V3.1=[0010 $\overline{x}_{45}$]} or Sol2 = {V2.1=[1100 $\overline{x}_{45}$], V3.1=[0010x_{45}]}. The new extended matrix S'_{tx} is then equal to:

$$S'_{tx1} = \begin{bmatrix}0101x15\\1100\underline{x45}\\0010x45\\1110x45\end{bmatrix} \text{ or } S'_{tx2} = \begin{bmatrix}0101\underline{x15}\\1100\overline{x45}\\0010x45\\1110x45\end{bmatrix}.$$

Now, let us show also that since the vector V1.1 was not a linear combination of the vectors V2.1, V3.1 and V4.1 with its four 1st bits, we can assume any new value for $x_{15} \in \{0,1\}$. Consequently, one can obtain four different matrices for S'_{tx} corresponding to the four possible combinations of x_{15} and x_{45}.

However, the size of the matrix S'_{tx} is now 4 rows by 5 columns, consequently we have to add also an extra row to S'_{tx} so as to obtain definitively an invertible (n×n) square matrix; n=5. But, the rank n=5 of the matrix S'_{tx} must be preserved and only a linearly independent extra row must be added to the four previous ones. The obtainment of such a solution is not so difficult since it exists exactly $(2^{n-1}-1)$ possible solutions for this choice, that is half of the space defined by n linearly independent vectors (except the null vector). Such a choice can be moreover be facilitated after having diagonalize or triangularize the first (n-1) vectors. Finally, by considering this latter property, the matrix S'_{tx} can be expressed of the form:

$$S'_{tx1} = \begin{bmatrix} 0101x15 \\ 1100\overline{x45} \\ 0010x45 \\ 1110x45 \\ x51x52x53x54x55 \end{bmatrix} \text{ or } S'_{tx2} = \begin{bmatrix} 0101\overline{x15} \\ 1100\overline{x45} \\ 0010x45 \\ 1110x45 \\ x51x52x53x54x55 \end{bmatrix}$$

Concluding this example, criteria 1 can always be satisfied by increasing the size n of the LFSR. It has been also shown that such an extension allows to select the matrix S'_{tx} inside a large set of possible solutions for the LFSR. Also, the extension process progresses at the low rate of $\log_2 w$ extra bits, not penalizing too much the initial solution; this value is low compared to the initial rank of the matrix. However, managing x_{ij} values is a little bit tedious since analytical computations are now required.

3.3 Conserving the rank n for T

By equation7, the characteristic matrix T of the LFSR will be a (n×n) square matrix. However, its rank is unknown. Since the matrix S'_t is built so as to be invertible, its rank is equal to n. Consequently, the rank of the matrix T depends on the rank of the matrix S'_{t+1}. If S'_{t+1} has a rank lower than n, it implies that T will get the same rank than S'_{t+1} [34]. Consequently, so as to guarantee a rank equal to n for T, it is mandatory that the rank of S'_{t+1} been also equal to n. Let us remind that S'_{t+1} is the matrix composed of the 2nd patterns of the detecting pairs, that is all the Vk.2 patterns. Consequently, the criteria 2 will be satisfied if and only if the n second patterns of the detecting pairs form also a set of linear independent vectors. If this condition is not naturally satisfied, the same technique as the one described in the previous paragraph can be applied to S'_{t+1}.

3.4 Guarantee of a primitive polynomial for T

The last criteria to verify is guaranteeing that the characteristic polynomial of the matrix T is primitive. Since we know that a primitive polynomial for the matrix T corresponds to an LFSR that will generate a maximal length sequence, this implies that any Vk.1 test patterns will be included in this sequence. Moreover, since the matrix T has been computed to preserve the n transitions from the test patterns Vk.1 to the patterns Vk.2, it means that the LFSR of matrix T will generate the n transitions pairs of test patterns.

At this step the computation of the characteristic polynomial of the matrix T is required, see [28]. By comparing the resulting value of p(x) with a pre-computed table of primitive polynomials, we can easily decide if the sequence will be unique and of maximal length. Tables of primitive polynomials can be found in[28] or [30]. If p(x) is primitive, then the LFSR defined by the matrix T will generate the required transition pairs and corresponds to the unique solution. If p(x) is not a primitive polynomial, the only way to obtain a new polynomial p'(x) primitive is to modify the transition matrix T. Then, two different cases can be described depending on the existence of don't care values in the components of the matrix T. Let us consider successively these two cases:

"Don't care bits exist": Now, if don't care values from ATPG exist originally in the matrix T, we can select the appropriate combinations of these so as to obtain a primitive polynomial. To illustrate this procedure, let us consider again the example of S'_{tx} matrices described in paragraph 3.2.

Example: we have obtained:

$$S'_{tx1} = \begin{bmatrix} 0101x15 \\ 1100\underline{x45} \\ 0010x45 \\ 1110x45 \\ x51x52x53x54x55 \end{bmatrix} \text{ or } S'_{tx2} = \begin{bmatrix} 0101\underline{x15} \\ 1100\overline{x45} \\ 0010x45 \\ 1110x45 \\ x51x52x53x54x55 \end{bmatrix}.$$

For the values x_{5j}, we have seen that it exist exactly $(2^{n-1}-1)$ possible combinations that satisfied the requirements. Since the calculation of S'^{-1}_{txi} will be too tedious to obtain with the analytical values for x_{5j}, we choose arbitrarily one available combination for x_{5j} and express for instance the matrix S'_{tx1} only in function of x_{15} and x_{45}. Choosing for instance $[x_{51}x_{52}x_{53}x_{54}x_{55}]=[00011]$ and trying to diagonalize the matrix S'_{tx1} in order to calculate its inverse results in: $S'_{tx1(equivalent)} = \begin{bmatrix} 1000 & \overline{x15} \oplus x45 \\ 0100 & \underline{x15} \\ 0010 & x45 \\ 0001 & \underline{1} \\ 0000 & \overline{x45} \end{bmatrix}$.

In order to conserve a rank n=5, this partial result implies that $x_{45}=0$, and then for the two possible remaining values of x_{45}, we can obtain the following solutions for S'_{tx1} and S'^{-1}_{txi}:

$S'_{tx1\text{-}1}=\begin{bmatrix} 01011 \\ 11000 \\ 00101 \\ 11100 \\ 00011 \end{bmatrix}$ and $S'^{-1}_{tx1\text{-}1}=\begin{bmatrix} 11001 \\ 10001 \\ 01010 \\ 01111 \\ 01110 \end{bmatrix}$ for $[x_{51}x_{52}x_{53}x_{54}x_{55}] =$ $[00011]$, $[x_{45}] = [0]$ and $[x_{15}] = [1]$.

$S'_{tx1\text{-}2}=\begin{bmatrix} 01010 \\ 11000 \\ 00101 \\ 11100 \\ 00011 \end{bmatrix}$ and $S'^{-1}_{tx1\text{-}2}=\begin{bmatrix} 10111 \\ 11111 \\ 01010 \\ 01111 \\ 01110 \end{bmatrix}$ for $[x_{51}x_{52}x_{53}x_{54}x_{55}] =$ $[00011]$, $[x_{45}] = [0]$ and $[x_{15}] = [0]$.

Now, we have to compute equation 7, $S'_{tx1\text{-}i}{}^{-1} \times S'_{(t+1)x1\text{-}i} = T$ to obtain the connectivity matrix T. For this, again many unknown variables appear in the expression of $S'_{(t+1)x1\text{-}i}$ that lead to the expression:

$$T_{x1\text{-}1} = S'_{tx1\text{-}1}{}^{-1} \times S'_{(t+1)x1\text{-}1} = \begin{bmatrix} 11001 \\ 10001 \\ 01010 \\ 01111 \\ 01110 \end{bmatrix} \times \begin{bmatrix} 1100 x'15 \\ 1110 x'25 \\ 1000 x'35 \\ 0011 x'45 \\ x'51 x'52 x'53 x'54 x'55 \end{bmatrix} \text{ or}$$

$$T_{x1\text{-}2} = S'_{tx1\text{-}2}{}^{-1} \times S'_{(t+1)x1\text{-}2} = \begin{bmatrix} 10111 \\ 11111 \\ 01010 \\ 01111 \\ 01110 \end{bmatrix} \times \begin{bmatrix} 1100 x'15 \\ 1110 x'25 \\ 1000 x'35 \\ 0011 x'45 \\ x'51 x'52 x'53 x'54 x'55 \end{bmatrix}$$

The x'_{ij} variables in the expression of $S'_{(t+1)x1\text{-}i}$ must be chosen so as to guarantee a rank of this matrix equals to 5, but also so that the characteristic polynomial of $T_{x1\text{-}i}$ be primitive. Then, again many solutions exist and wejust give below one of them for $T_{x1\text{-}1}$:

If we choose $[x'_{51}x'_{52}x'_{53}x'_{54}x'_{55}] = [00001]$, we obtain $T_{x1\text{-}1} = \begin{bmatrix} 0010 & \overline{x'15} \oplus x'25 \\ 1100 & x'15 \\ 1101 & x'25 \oplus x'45 \\ 0101 & x'25 \oplus x'35 \oplus x'45 \\ 0101 & x'25 \oplus x'35 \oplus x'45 \end{bmatrix}$. Now, trying to diagonalize $T_{x1\text{-}1}$ and then expressing it as standard form LFSR, see section 2.1, we obtain $T_{x1\text{-}1(\text{equivalent})} = \begin{bmatrix} 0000 & 1 \\ 1000 & x'35 \\ 0100 & x'15 \oplus x'35 \\ 0010 & x'15 \oplus x'25 \\ 0001 & x'15 \oplus x'25 \oplus x'45 \end{bmatrix}$

Now by choosing $p(x) = 1+x^2+x^5 = c_0x^0 + c_1x^1 + \ldots + c_kx^k + \ldots + c_nx^n$, a primitive polynomial and solving the relation $\begin{bmatrix} 1 \\ x'35 \\ x'15 \oplus x'35 \\ x'15 \oplus x'25 \\ x'15 \oplus x'25 \oplus x'45 \end{bmatrix} = \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \\ c4 \end{bmatrix}$; $\begin{bmatrix} x'15 \\ x'25 \\ x'35 \\ x'45 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $T_{x1\text{-}1} = \begin{bmatrix} 00100 \\ 11001 \\ 11011 \\ 01010 \\ 01011 \end{bmatrix}$.

The reader could easily verify that the sequence generated by the LFSR of transition matrix $T_{x1\text{-}1}$ is of maximal length and also includes the four initial pairs of deterministic patterns.

"Don't care bits does not exist": If any don't care value does not exist in the matrix Tand the characteristic polynomial of T is not primitive, the only remaining possibility is to modify at least one test vector in one pair of the initial test set. For this, consider again the previous example and calculate its characteristic polynomial.

Example: we have obtained in section 3.1: $T = \begin{bmatrix} 0111 \\ 1001 \\ 1000 \\ 0101 \end{bmatrix}$.

The characteristic polynomial of T is $p(x) = 1 + x + x^2 + x^3 + x^4$, not primitive and the corresponding LFSR describes three different cycles.

So, even if the four transition pairs P1 to P4 exist in the sub-sequences, they could not be generated without any extra mechanism to switch from a sub-sequence to another one. This solution is not acceptable and consequently we have to modify at least one pattern to obtain a new primitive transition matrix for T. Let us consider that we modify a second pattern of any test pair instead of a first pattern. This can be justified by the fact that the matrix S'_{t+1} do not need to be inverted in equation 7, $S'_t{}^{-1} \times S'_{t+1} = T$. An opposite point of view is also possible, but it will lead to a symbolic calculation for the inverse $S'_t{}^{-1}$. So, let us call Px the modified pair from the initial set such that Px={Vk.1;Vk.2=[x0,x1,...,xn-1]}; k may take any value from 1 to n. Then, using equation 7 again, we calculate a new value for T, called Tx whose contents depends on the [x0,x1,...,xn-1] coefficients. By selecting appropriate values for xi, we can ensure a primitive polynomial for Tx. If such a result is not possible for the selected value of k, we can choose a different value for k and repeat the process. If all the values of k from 1 to n lead to unsatisfactory solutions, one can decide to modify a Vk.1 vector instead of a Vk.2 vector. Also, in an extreme case, the selection of an additional Px' to Px is possible and provide much more degrees of freedom, but of course one less pair.

Example: in our example, we select k=4 for the V4.2 pattern. Then the transition P4 will be no longer generated and using equation 7, we obtain:

$$S'_t{}^{-1} \times S'_{t+1} = T_x = \begin{bmatrix} 1011 \\ 1111 \\ 0010 \\ 0111 \end{bmatrix} \times \begin{bmatrix} 1100 \\ 1110 \\ 1000 \\ X_0X_1X_2X_3 \end{bmatrix} = \begin{bmatrix} \overline{X_0} & \overline{X_1} & \overline{X_2} & X_3 \\ \overline{X_0} & X_1 & \overline{X_2} & X_3 \\ 1 & \overline{0} & \overline{0} & 0 \\ X_0 & \overline{X_1} & \overline{X_2} & X_3 \end{bmatrix}$$

In this example, the coefficients Xi must be chosen so that the rank of the matrix Tx remains equal to n=4 but also so that the new characteristic polynomial p(x) is now primitive. An exhaustive study for the 16 possible values of leads to fix x3=1 so as to conserve a rank equal to 4 for the matrix Tx. Then, depending on the values of x0, x1 and x2 it remains 8 possible solutions to obtain a primitive characteristic polynomial for Tx. Six of them lead to multiple sub-sequences or cycles described by their corresponding LFSR, and do not correspond consequently to primitive polynomials. The two remaining possible solutions with a primitive polynomial are then the following:

Example:

$T_x = \begin{bmatrix} 0101 \\ 1011 \\ 1000 \\ 0111 \end{bmatrix}$ for $[X_0X_1X_2X_3] = [0011]$ and $p(x) = 1+x^3+x^4$ or

$T_x = \begin{bmatrix} 1111 \\ 0001 \\ 1000 \\ 1101 \end{bmatrix}$ for $[X_0X_1X_2X_3] = [1011]$ and $p(x) = 1+x+x^4$

For these two different LFSRs, the corresponding state sequences are then of maximal length and include the pairs P1, P2 and P3, but not the pair P4 since this latter has been deleted to fulfill the primitive nature of the characteristic polynomial.

In conclusion, when don't care values exist from ATPG or are added with the extension process of the number of bits, this small example has shown that a large number of solutions are available to satisfy a primitive characteristic polynomial for the transition matrix T. In fact, one can

say that since the number of unknown variables is large, it is even much more difficult to enumerate all the solutions without symbolic calculations.

4 Comparison to other proposals

Now, let us experiment a comparison of our solution with various TPGs already proposed for two-patterns testing. As it has been explained in the introduction, managing the complexity of a TPG is really a trade-off between three parameters: the expected fault coverage, the test time (equivalent to the length of the test sequence) and finally the hardware cost. For instance, assuming a maximal fault coverage, a reduction in the test sequence length implies most of the time an increase of the TPG hardware cost, and vice-versa. Hereafter, in our attempt of comparison, we will give some indications on these parameters. To consider a comprehensive list of two-patterns test TPG approaches will be a cumbersome task, hence we will just analyze the following outstanding contributions in this field. We selected [31] [16] [19] [22] [24] [17] [18] [15] and [25] for the comparison. Table 1 reports our perception of these contributions. The first column contains authors with the year of contribution. The complete publications are listed in the reference list.

In table 1, Column 2 - TPG Structure tries to categorize the contributions depending on the basic architecture of the TPG: LFSR and NLFSR - Linear and Non-Linear Finite State Registers, CA - Cellular Automata, SR and MISR - Shift Register and Multiple Inputs Shift Register, or ACCUmulator. Column 3 - Category classifies it into: D - a deterministic approach, E or PE - an exhaustive or pseudo-exhaustive approach, PR - a pseudo-random approach and PD - a pseudo-deterministic approach. Then, the fault coverage (FC), the test sequence length and the hardware cost are estimated. The test length and the hardware are given in function of the number of flip-flops of an equivalent n-size LFSR but also with the addition of Control Logic (CL), BILBO (Built-In Logic Block Observer) or Accumulator and Counter. The Fault Coverage (FC) is reported depending on the fact that it can be Maximal (Max) or lower than maximal (High). For this last case, the interpretations of high is delicate in the sense that some more or less percentages for the FC must be analyzed extremely carefully in function of the circuit, the test length and many others parameters.

From table 1, it appears that solutions can be classified in two (maybe three) sets concerning the hardware. A first set requires only n flip-flops while a second set uses generally 2×n flips-flops and most often with the addition of some control logic (CL). Then, from these two sets, the test sequences become generally lower or equal to (2^n-1) and $2^n(2^n-1)$ respectively. The corresponding fault coverages are consequently maximal (Max) if the test sequences are exhaustive or pseudo-exhaustive (E or PE), still maximal in the case of Deterministic or Pseudo-Deterministic (D or PD) and lower than maximal, that is high, if the test sequences are pseudo-random sequences. Then, selecting the best approach for your application depends on the test approach (PR, PD or D), also on the possible availability of resources in your chip: SCAN, BILBO, accumulator and soon. In this context, our proposal is, to our best knowledge the first pseudo-deterministic technique for two-patterns testing and consequently offers: a maximal fault coverage, for smaller test sequences than pseudo-random ones and that, for a slight increase of the hardware cost. Certainly the most important thing is that our scheme is not in conflict with other techniques, such as [22] or [25], and consequently when merged, should lead to interesting practical solutions.

Contributions	TPG	Cat.	FC	Length	Hardware Cost	Remarks
[31] Starke - 1984	NLFSR	D	Max	Low	2n+CL	CL is excessive
[16] Craig - 1985	NLFSR	PE adj.	High	$n(2^n-1)$	2n+CL	
[19] Furuya - 1991	LFSR	PR	High	$<(2^{2n}-1)$	2n	transition FC
[22] Zhang - 1992	XLFSR	PR	High	$<(2^n-1)$	n	
[24] Pilarski - 1993	SR	PR	High	$<(2^{2n}-1)$	2n	Circular BIST
[17] Vuksic - 1994	MISR	E	Max	$2^n(2^n-1)$	n+BILBO	need BILBO
[18] Wurth - 1995	MISR	E	Max	$<2^n(2^n-1)$	n+BILBO	=[17]+Optimization
[15] Voyiatzis - 1995	ACCU	E	Max	$2^n(2^n-1)$	Accu+Counter	for Datapaths
[25] Chen - 1996	LFSR-CA	PR	High	$<(2^n-1)$	n	generalize [22], =[21]
our scheme	LFSR	PD	Max	Medium	n	1st PD technique

Table 1: Comparison of two-pattern test TPGs

4 Conclusion

Linear Feedback Shift Registers have proved in the past to be key elements for BIST architectures. In this paper, it is shown that an LFSR can be synthesized so as to embed in its sequence some deterministic pairs of patterns for delay faults testing. Key features of LFSRs have been analyzed in section 2 as well as more advanced properties

concerning transition matrices. A three criteria algorithm is described and discusses the necessary and sufficient conditions to synthesize a pseudo-deterministic LFSR. It is shown through an example, that many possibilities exist to satisfy these conditions and may also offer some optimization approaches. Finally, some limitations and possible improvements of the method are reported. It is demonstrated that this scheme can be used in conjunction with any other existing proposals.

Acknowledgments

The authors would like to thank the North Atlantic Treaty Organization (NATO), France, for providing Fellow Research Grant and the EngineeringResearch Center of Lucent Technologies Bell Labs in Princeton, New Jersey, USA, for sponsoring this research.

References

[1] T.W. Williams and K.P. Parker, "Design for Testability - A Survey", IEEE Trans. on Comp., vol.C-31, no.1, pp. 2-15, 1982.

[2] V. D. Agrawal and al., "Built-In Self Test for Digital Integrated Circuits", AT&T Technical Journal, vol.73, no.2, March-April, 1994.

[3] B. H. Seiss, P. M. Trouborst and M. H.Schulz, "Test Point Insertion for Scan-based BIST", Proc. of European Test Conf., 1991.

[4] S. K. Gupta and D. K. Pradhan, "Utilization of On-Line (Concurrent) Checkers during Built-In Self Test and Vice-Versa", IEEE Trans. on Computers, vol.45, no.1, Jan. 1996.

[5] S. Hellebrand, S. Tarnick and J. Rajski, "Generation of Vectors Patterns through Reseeding of Multiple Polynomial Linear Feedback Shift Registers", Proc. of Int'l Test Conf., pp. 120-129, 1992.

[6] H. J. Wunderlich and G. Kiefer, "Scan-based BIST with complete fault coverage and low hardware overhead", IEEE European Test Workshop, June, 1996.

[7] N.A. Touba and E.J. McCluskey, "Transformed Pseudo-Random Patterns for BIST", Proc. of VLSI Test Symp., 1995.

[8] M. Chatterjee and D. K.Pradhan, "A Novel Pattern Generator for Near-Perfect Fault-Coverage", Proc. of VLSI Test Symp., 1995.

[9] M. Lempel, S. K. Gupta and M. A. Breuer, "Test embedding with Discrete Logarithms", Proc. VLSI Test Symp., May 1994.

[10] C.Dufaza, H. Viallon and C. Chevalier, "BIST Hardware Generator for Mixed Test Scheme", Proc. of European Des. & Test Conf., 1995.

[11] D. Kagaris and S.Tragoudas, "Generating Deterministic Unordered Test Patterns with Counters", Proc. of VLSI Test Symp., pp. 374-379, May 1996.

[12] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits", IEEE Trans. Comp. Aided Design, vol.6, no.5, pp. 694-703, Sept. 1987.

[13] A. K. Pramanick and S. M. Reddy, "On the detection of delay faults", Proc. of Int'l Test Conf., pp. 845-856, Oct. 1988.

[14] G. L. Smith, "Model for Delay Faults based upon Paths", Proc. of Int'l Test Conf., pp. 342-349, Nov. 1985.

[15] I. Voyiatzis and al., "Accumulator-based BIST approach for stuck-open and delay faults testing", Proc. of European Des. & Test Conf., pp. 431-435, 1995.

[16] G. L. Craig and C. RKime, "Pseudo-Exhaustive Adjacency Testing: A BIST Approach for Stuck-Open Faults", Proc. of Int'l Test Conf., pp. 126-137, Oct. 1985.

[17] A. Vuksic and K. Fuchs, "A New BIST Approach for Delay Fault Testing", Proc. of European Des. & Test Conf., pp. 284-288, 1994.

[18] B. Wurth and K. Fuchs, "A BIST Approach to Delay Fault Testing with Reduced Test Length", Proc. of European Des. & Test Conf., pp. 418-483, 1995.

[19] K. Furuya and E. J. McCluskey, "Two-Pattern Test Capabilities of Autonomous TPG Circuits", Proc. Int'l Test Conf., pp. 704-711, Oct. 1991.

[20] K. Furuya, S. Yamazaki and M. Sato, "Evaluations of various TPG Circuits for use inTwo-pattern Testing", Proc. VLSI Test Symp., pp. 242-247, May 1994.

[21] C. Chen and S. K. Gupta, "BIST Test Pattern Generators for Stuck-Open and Delay Testing", Proc. of European Des. & Test Conf., pp. 289-296, 1994.

[22] S. Zhang, R. Byrne and D.M. Miller, "BIST Generators for Sequential Faults", Proc. ICCD, pp. 260-263, 1992.

[23] D. K. Pradhan and M. Chatterjee, "GLFSR - A new Test Pattern Generator for Built-In Self Test", Proc. Int'l Test Conf., pp. 481-490, 1994.

[24] S. Pilarski and A.Pierzynska, "BIST and Delay Fault Detection", Proc. Int'l Test Conf., pp. 236-242, Oct. 93.

[25] C. Chen and S. K. Gupta, "BIST Test Pattern Generators for Two-Pattern Testing - Theory and Design Algorithms", IEEE Trans. on Comp., vol.45, no.3, pp. 257-269, 1996.

[26] W. Wang and S. K. Gupta, "Weighted Random Robust Path Delay Testing of Synthesized Multilevel Circuits", Proc. of VLSI Test Symp., pp. 291-297, May 1994.

[27] W. Daehn and J. Mucha, "Hardware Test Pattern Generation for Built-In Testing", Proc. of Int'l Test Conf., pp. 110-113, 1981.

[28] W. W. Peterson and E. J. Weldon, "Error Correcting Codes", 2nd Ed. Cambridge, MA, M.I.T. Press, 1972.

[29] S. W. Golomb, "Shift Register Sequences", Laguna Hills, Calif.: Aegean Park Press, 1982.

[30] P.H. Bardell, W.H. McAnney and J. Savir, "Built-In Test for VLSI: pseudo-random techniques", John Wiley & Sons, New York, 1987.

[31] W. Starke, "Built-In Test for CMOS Circuits", Proc. of Int'l Test Conf., pp. 309-314, 1984.