Finding Characteristic Polynomials for LFSMs Used as Test Pattern Generators

Dimitri Kagaris
Electrical and Computer Engineering Department
Southern Illinois University
Carbondale, IL 62901
USA

Abstract: – A built–in Test Pattern Generation (TPG) mechanism that is very popular for pseudorandom or pseudo–exhaustive TPG is a shift register whose initial portion is configured as a Linear Feedback Shift Register (LFSR), or some other kind of a Linear Finite State Machine (LFSM). A fundamental problem that exists in this mechanism is the presence of linear dependencies among the bit sequences produced by different stages. A formula that relates the linear dependencies with the characteristic polynomial of any kind of LFSM has been recently found. In this work we show how the computation of this formula can be made faster. This allows the faster computation of an appropriate characteristic polynomial for the target LFSM so that linear dependecies are minimized.

Keywords: – Built–in Self–Test, Test Pattern Generation, Linear Finite State Machines.

1 Introduction

In built–in self–test (BIST) test pattern generation (TPG), test stimuli that are needed to test a chip are produced by hardware that is added on the same chip. The test patterns can be produced in a deterministic, pseudorandom, or pseudoexhaustive manner [1]. For pseudoexhaustive or pseudorandom TPG, an established approach is to use a combination of a Type–1 LFSR with a simple shift register (SR), known as LFSR/SR or Extended LFSR (Fig. 1). The shift register part is in most cases identified with the scan chain. When the LFSR part of the LFSR/SR is initialized by a non–zero vector (seed), it cycles through p distinct states, where p is the period of the characteristic polynomial of the LFSR. As the successive states of the LFSR are shifted through the SR part, they provide test patterns for the test–phase inputs. Each test–phase input of the circuit under test is identified by its relative position in the LFSR/SR. The set of test–phase inputs upon which each test–phase output depends will be referred to as *Aset* (*Affector set*). The maximum Aset size will be denoted by w. The maximum Aset size can be optionally reduced by introducing extra test–phase inputs (see [1, 7, 16, 9, 13]). If w is relatively small, then pseudoexhaustive testing could be applied to test the circuit in 2^w cycles (or another power close to 2^w). Otherwise, pseudorandom testing is done for a prescribed number of R cycles, where R is determined by the level of fault coverage desired and is in general much smaller than 2^w.

The problem with the generation of patterns by such a mechanism is that a subset of stages may take on values that are linearly dependent, so that if this subset is needed in some Aset A, the corresponding test–phase output is never going to have all of its $2^{|A|}$ possible input patterns for pseudoexhaustive TPG. The problem with such linear dependencies is also present in pseudorandom TPG, in the sense that the more linear dependencies occur, the less random the generated patterns are [5]. In other words, even if pseudorandom TPG will not apply all possible test patterns, one does not want to exclude beforehand that certain patterns will never be applied. (This has been called the *principle of possible exhaustion* [2].) The issue of linear dependencies has been studied by various researchers (see, e.g., [5, 3, 10, 12]).

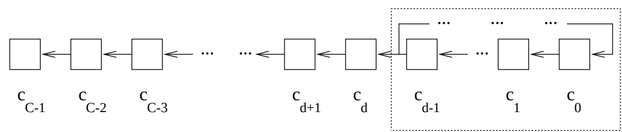


Fig. 1. Block diagram of an LFSR/SR with C stages (first d stages are configured in an LFSR).

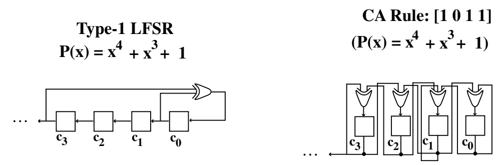
A relation between linear dependencies among the stages of a Type–1 LFSR/SR and the characteristic polynomial $P(x)$ of the LFSR has been established in [4, 11, 15]. This formula has been generalized for any kind of LFSM in [8]. Both of these formulas are analyzed in Section 3. In this work, we show how the generalized formula can be computed faster. This allows for the faster identification of an appropriate characteristic polynomial, so that the linear dependencies are minimized.

2 Preliminaries

Built–in TPG mechanims use some type of a Linear Finite State Machine (LFSM), such as Type–1 LFSRs, Type–2 LFSRs and Cellular Automata (CA), among others. Different LFSMs have different hardware overheads and diferrent behavior in terms of the sequence of states that they generate. We start with the formal definition of an LFSM.

A finite state machine with n inputs, m outputs, and f binary memory stages (flip–flops) is called a *Linear Finite State Machine* if for the current state vector s, input vector x, and output vector y, the next state vector s_{new} and the present output vector y can be expressed as $s_{\text{new}} = As + Bx$ and $y = Cs + Dx$, where A, B, C, D are matrices of dimension $f \times w$, $f \times n$, $m \times w$, $m \times n$, respectively, and the addition is done mod 2 (XOR operation).

When used as test pattern generators the LFSMs, have no external inputs (they are *autonomous*), that is, matrices B and D above are 0. Such an LFSM L will be denoted using the notation $L = (A, C)$. In addition, for most test pattern generators , matrix C is is in fact equal to the identity matrix I, that is, the output vector is just the state vector, but there are also cases where this is not so.



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Cyc	Stages 9 8 7 6 5 4	3 2 1 0
0	0 0 0 0 0 0	0 0 0 1
1	0 0 0 0 0 0	0 0 1 1
2	0 0 0 0 0 0	0 1 1 1
3	0 0 0 0 0 0	1 1 1 1
4	0 0 0 0 0 1	1 1 1 0
5	0 0 0 0 1 1	1 1 0 1
6	0 0 0 1 1 1	1 0 1 0
7	0 0 1 1 1 1	0 1 0 1
8	0 1 1 1 1 0	1 0 1 1
9	1 1 1 1 0 1	0 1 1 0
10	1 1 1 0 1 0	1 1 0 0
11	1 1 0 1 0 1	1 0 0 1
12	1 0 1 0 1 1	0 0 1 0
13	0 1 0 1 1 0	0 1 0 0
14	1 0 1 1 0 0	1 0 0 0
15	0 1 1 0 0 1	0 0 0 1
16	1 1 0 0 1 0	0 0 1 1
17	1 0 0 1 0 0	0 1 1 1
18	0 0 1 0 0 0	1 1 1 1
19	0 1 0 0 0 1	1 1 1 0
20	1 0 0 0 0 1	1 1 0 1
21	0 0 0 1 1 1	1 0 1 0
22	0 0 1 1 1 1	0 1 0 1
23	0 1 1 1 1 0	1 0 1 1

(a)

Cyc	Stages 9 8 7 6 5 4	3 2 1 0
0	0 0 0 0 0 0	0 0 0 1
1	0 0 0 0 0 0	0 0 1 1
2	0 0 0 0 0 0	0 1 0 0
3	0 0 0 0 0 0	1 0 1 0
4	0 0 0 0 0 1	1 0 1 1
5	0 0 0 0 1 1	1 0 0 0
6	0 0 0 1 1 1	1 1 0 0
7	0 0 1 1 1 1	0 1 1 0
8	0 1 1 1 1 0	1 1 0 1
9	1 1 1 1 0 1	0 1 0 1
10	1 1 1 0 1 0	1 0 0 1
11	1 1 0 1 0 1	1 1 1 1
12	1 0 1 0 1 1	0 0 1 0
13	0 1 0 1 1 0	0 1 1 1
14	1 0 1 1 0 0	1 1 1 0
15	0 1 1 0 0 1	0 0 0 1
16	1 1 0 0 1 0	0 0 1 1
17	1 0 0 1 0 0	0 1 0 0
18	0 0 1 0 0 0	1 0 1 0
19	0 1 0 0 0 1	1 0 1 1
20	1 0 0 0 0 1	1 0 0 0
21	0 0 0 1 1 1	1 1 0 0
22	0 0 1 1 1 1	0 1 1 0
23	0 1 1 1 1 0	1 1 0 1

(b)

Fig. 2: LFSM/SRs and observation matrices: (a) Type–1 LFSR. (b) CA.

Fig. 2(a) shows an LFSM/SR based on a Type–1 LFSR and Fig. 2(b) shows an LFSM/SR based on a Cellular Automaton (CA). In both cases the LFSM/SR has 10 stages ($c_0 \ldots c_9$), where the LFSM part comprises the first four stages (c_0, c_1, c_2, c_3). Matrix A is called the state transition matrix of the LFSM. The characteristic polynomial $P(x)$ of this matrix, defined as the determinant of matrix $(xI - A)$, where x denotes the polynomial variable, is called the characteristic polynomial of the LFSM. If the characteristic polynomial of an LFSM is *primitive* (see, e.g., [14]), then the LFSM goes through the maximum possible number of states before returning to its initial state. The sequence of states that each stage goes through is a periodic one starting from the time the stage receives its first "1". The bit sequence of the most significant stage (stage c_3 in this case) is known as the *characteristic sequence*. The length of the characteristic sequence is $2^4 - 1 = 15$. For each LFSM, the figure shows also the observation matrix, which gives the overall states of the LFSM/SR. In the examples shown, after an initialization phase of 10 cycles (0 through 9), each stage will repeat its sequence of length 15. Both LFSMs have the same characteristic sequence (as they have the same characteristic polynomial) but their sequence of overall states differs.

3 The formulas

We present below the formulas that relate the linear dependencies with the characteristic polynomial.

Theorem 3.1 [4, 15, 11]: *Given a Type–1 LFSR with characteristic polynomial $P(x)$ of degree d and its associated LFSR/SR structure with a total number of C stages, a subset A of k LFSR/SR stages $A = \{a_1, a_2, ..., a_k\}$, $0 \leq a_1, a_2, ..., a_k \leq C-1$, has no linear dependencies if and only if the k polynomials $P_i = x^{a_i} \bmod P'(x)$, $1 \leq i \leq k$, are linearly independent (or equivalently, there is no subset $A' \subseteq A$ such that $\sum_{a' \in A'}^{k} x^{a'} \neq 0 \bmod P'(x)$), where $P'(x)$ depends on the numbering of the LFSR/SR stages and is equal to*
(i) $P'(x) = P(x)$, if the LFSR/SR stages are numbered so that the LFSR/SR stage with index $C-1$ coincides with the 0th stage of the LFSR [4, 11]
(ii) $P'(x) = \tilde{P}(x)$, if the LFSR/SR stages are numbered so that the LFSR/SR stage with index 0 coincides with the 0th stage of the LFSR [1] [15].

Such a formula is important because the check for linear dependencies among a given set of stages A requires exponential time if done explicitly (generation of 2^k states, $k = |A|$), whereas with the above relation, the check can be done in very fast polynomial time by checking through Gaussian elimination whether the $k \times k$ matrix formed by the k polynomials $P_i(x) = x^{a_i} \bmod P'(x)$, seen as binary vectors, has rank k. As an illustration, assume that an output Y_1 of a circuit under test depends on LFSM/SR stages c_8, c_6, c_3, c_0. If the LFSM/SR is based on a Type–1 LFSR with characteristic polynomial $P(x) = x^4 + x^3 + 1$, then by examining the corresponding observation matrix (Fig. 2(a)) and starting from cycle 8, we see that a linear dependency occurs among the bit sequences in stages c_8, c_6, c_3, c_0, that is, $c_8 + c_6 + c_3 + c_0 = 0$, as predicted by Theorem 3.1, since $x^8 + x^6 + x^3 + 1 \bmod \tilde{P}(x) = 0$.

The formula in Theorem 3.1 has been generalized in [8] to apply to any arbitrary Linear Finite State Machine (LFSM). The generalized formula allows also the linear dependency check to be done in polynomial time. Each stage $c_{a_i}, a_i \in A$, is associated with a proper LFSM stage c_k, $0 \leq k \leq d-1$, according to the relation

$$\lambda(a_i) = \begin{cases} d-1 & \text{if } a_i \geq d \\ a_i & \text{if } a_i \leq d-1 \end{cases}$$

In addition, each stage $c_{a_i}, a_i \in A$, is associated with the following number:

$$\delta(a_i) = \begin{cases} a_{\max} - a_i & \text{if } a_i \geq d \\ a_{\max} - (d-1) & \text{if } a_i \leq d-1 \end{cases}$$

We define also $r(i)$ to be the set of the stages of the corresponding Type–1 LFSR that the ith stage of a given LFSM, $0 \leq i \leq d-1$, depends on (starting from an appropriate state). Let R be a similarity matrix such that $M = RLR^{-1}$, where M is the transition matrix of the given LFSM and L is the transition matrix of the corresponding Type–1 LFSR with the same characteristic polynomial. Then set $r(i)$ consists of all numbers j, $0 \leq j \leq d-1$, such that the $(d-1-j)$th bit of the $(d-i)$th row of R (seen as a binary number) is '1'.

Theorem 3.2 *Given a Linear Finite State Machine with characteristic polynomial $P(x)$ of degree d and its associated LFSM/SR structure with a total number of C stages, a subset A of k LFSR/SR stages $A = \{a_1, a_2, ..., a_k\}$, $0 \leq a_1, a_2, ..., a_k \leq C-1$, has no linear dependencies if and only if the k polynomials*

$$P_i(x) = \sum_{j \in r(\lambda(a_i))} x^{d-1-j+\delta(a_i)} \bmod P(x), \quad 1 \leq i \leq k$$

[1] $\tilde{P}(x)$ is the reciprocal polynomial of $P(x)$ defined as $\tilde{P}(x) = x^d P(1/x)$

are linearly independent.

To apply the above formula, we have to find a similarity matrix R such that $M = RLR^{-1}$. Such a natrix always exists because matrices M and L are known to be similar (see, e.g., [14].) Once a matrix R has been obtained, set $r(i)$ is formed by taking all numbers j, $0 \leq j \leq d-1$, for which the $(d-1-j)$th bit of the $(d-i)$th row of R is '1'.

The application of Theorem 3.2 is illustrated by the following example.

Consider a Cellular Automaton with rule vector . Its characteristic polynomial is x^4+x^3+1. The bit sequences of stages c_7, c_1, c_0 are linearly dependent as can be verified from the observation matrix (see Fig. 2(b)). The transition matrix M of this CA along with the transition matrix L of the corresponding Type–1 LFSR and a similarity matrix R are shown below:

$$M = \begin{bmatrix} 1\,1\,0\,0 \\ 1\,0\,1\,0 \\ 0\,1\,1\,1 \\ 0\,0\,1\,1 \end{bmatrix}, \quad L = \begin{bmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,0\,0\,1 \end{bmatrix}, \quad R = \begin{bmatrix} 0\,0\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,1\,1 \\ 1\,0\,1\,0 \end{bmatrix}$$

We have that $\lambda(7) = 3$, $\lambda(1) = 1$, $\lambda(0) = 0$, $\delta(7) = 0$, $\delta(1) = 4$, $\delta(0) = 4$, $r(3) = \{1, 0\}$ (1st row of R), $r(1) = \{3, 2, 1, 0\}$ (3rd row of R), $r(0) = \{3, 1\}$ (4th row of R). So the double sum in Theorem 3.2 becomes: $(x^{3-1+0} + x^{3-0+0}) + (x^{3-3+4} + x^{3-2+4} + x^{3-1+4} + x^{3-0+4}) + (x^{3-3+4} + x^{3-1+4}) = x^7+x^5+x^3+x^2$, which is divisible by x^4+x^3+1 (quotient x^3+x^2).

The use of the formulas in Theorems 3.1 and 3.2 above in the case of pseudoexhaustive TPG (see also [4]) is done as follows: Assume the maximum Aset size is w in a combinational or fully scanned circuit. To be able to do the pseudoexhaustive test in 2^w cycles (plus some minor terms for initialization), we have to check for the following: The LFSM part of the LFSM/SR must have $d = w$ stages and its characteristic polynomial must be primitive. Then we go through the list of the primitive polynomials of degree d (such lists have been published or can be computed again) and we try to find one by applying the appropriate formula above that satisfies each and every Aset of the circuit. The number of all primitive polynomials of degree d is given by the formula $\frac{\phi(2^d-1)}{d}$, where $\phi()$ is Euler's function. For example there are 120032 primitive polynomials of degree 22. So any improvements in the time it takes to apply the formula check are significant. For pseudorandom test, a similar procedure is followed but we try to maximize the number of Asets that have rank $\max(d, |A|)$, where d the number of stages in the LFSM part.

4 Finding a similarity matrix

Finding a similarity matrix R that can be used with Theorem 3.2 requires the following: The equation $M = RLR^{-1}$ is equivalent (since we know that R exists) to the homogeneous linear system $MR - RL = 0$ with unknowns the n^2 entries of the $n \times n$ matrix R, and the additional requirement that R is invertible. The system can be put in the form $KV = 0$ where V is an $n^2 \times 1$ vector consisting of the unknowns in R and K is an $n^2 \times n^2$ matrix consisting of the appropriate combinations of coefficients from L and M . Matrix K has rank less than n^2, that is, there are multiple solutions to the equation $M = RLR^{-1}$. One such solution can be found by Gaussian elimination, in time $O((n^2)^3) = O(n^6)$, as there are n^2 unknowns.

We show here that the $O(n^6)$ time complexity can be reduced by obviating the need to solve the linear system $MR = RL$, and then having to check for invertibility of R. A similarity matrix R such that $M = RLR^{-1}$ can be found by inversion of an appropriate $n \times n$ matrix U , which reduces the complexity to $O(n^3)$. The computation of matrix U is done as follows:

We first define the shift register (SR) counterpart of an LFSM $F = (M, I)$ to be another LFSM $F_{\mathrm{SR}} = (M, U)$, such that F and F_{SR} have the same characteristic sequence but the bit sequence produced at output stage i of F_{SR} is a shift–up–by–1 version of stage $i-1$, $1 \leq i \leq d$.

We have proven the following lemmas:

Lemma 4.1 *Given an LFSM $F = (M, I)$ of d cells, its shift register counterpart is $F_{\mathrm{SR}} = (M, U)$, where U is obtained by putting as its ith row, the first row of each matrix M^{i-1}, $1 \leq i \leq d$.*

Lemma 4.2 *For any LFSM $F = (M, I)$, matrix U in $F_{\mathrm{SR}} = (M, U)$ is non–singular.*

Since U is non–singular, matrix $L = U \cdot M \cdot U^{-1}$ exists. Therefore machines $F_1 = (M, U)$ and $F_2 = (L, I)$ are similar and thus equivalent. Because of the similarity, we know that the characteristic polynomial of matrix M is the same as the characteristic polynomial of matrix L. Because of the equivalence, we know that when F_1 is at state s, F_2 is at state $s' = U \cdot s$, and the present output of F_1 is $y = U \cdot s$ which is equal to the present output $y' = s' = U \cdot s$ of F_2. So the bit sequence of stage i of F_2 is the same as the bit sequence of output stage i of F_1 and thus is also a shift–up–by–1 version of the bit sequence in stage $i - 1$, that is, machine $F_2 = (L, I)$ is in fact a Type–1 LFSR.

Now in order to obtain fast the matrix U, we observe the following:

Let $R_1(A)$ denote the first row of matrix A. Then,

$$R_1(M^i) = R_1(M^{i-1} \cdot M) = R_1(M^{i-1}) \cdot M, \quad 1 \le i \le d-1,$$

where $R_1(M^0) = [100...0]$. Considering each $R_1(M^i)$ as a $1 \times d$ vector $\hat{s}_i$ we have that

$$\hat{s}_i = R_1(M^i) = R_1(M^{i-1}) \cdot M = \hat{s}_{i-1} \cdot M \Rightarrow$$

$$(\hat{s}_i)^{\mathrm{T}} = M^{\mathrm{T}} \cdot (\hat{s}_{i-1})^{\mathrm{T}},$$

(A^{T} mean the transpose of matrix A) that is, the $1 \times d$ $R_1(M^i)$ vectors can be seen as successive states ($d \times 1$ vectors) of an LFSM with transition matrix M^{T}, starting with state $(\hat{s}_0)^T = [100...0]^{\mathrm{T}}$. So instead of doing the explicit matrix multiplications to obtain M^i, $0 \le i \le d-1$, we can simply simulate an LFSM with transition matrix M^{T} for just $d - 1$ cycles starting from state $[100...0]^{\mathrm{T}}$. The resulting states comprise the rows of U.

As an illustration of the approach, consider the following examples:

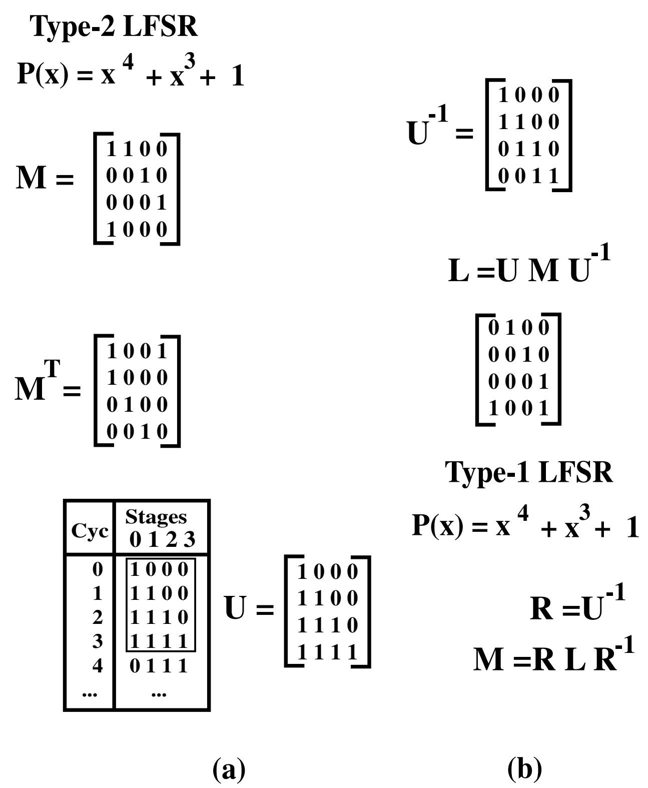


Fig. 3: Illustration of the method for computing a similarity matrix for a Type–2 LFSR.

Example 1 Consider the Type–2 LFSR with characteristic polynomial $P(x) = x^4 + x^3 + 1$. Its transition matrix M is shown in Fig. 3(a). The successive states starting with [1000] of the LFSM with transition matrix M^{T} are also shown in Fig. 3(a). The first four successive states constitute matrix U. As can be verified (Fig. 3(b)), U is indeed a similarity matrix with respect to the Type–1 LFSR with the same characteristic polynomial and transition matrix L. The required matrix R such that $M = RLR^{-1}$ is then $R = U^{-1}$.

Example 2 Consider the Cellular automaton in Fig. 4(a), which is the same as that in Fig. 2(b). The transition matrix M^{T} of the corresponding LFSM is also shown in Fig. 4(a). (The transition matrix of a CA is symmetric, and so $M = M^{\mathrm{T}}$ in this example.) The successive states starting with [1000] of the LFSM with transition matrix M^{T} are shown in Fig. 4(a). The first four successive states constitute matrix U.

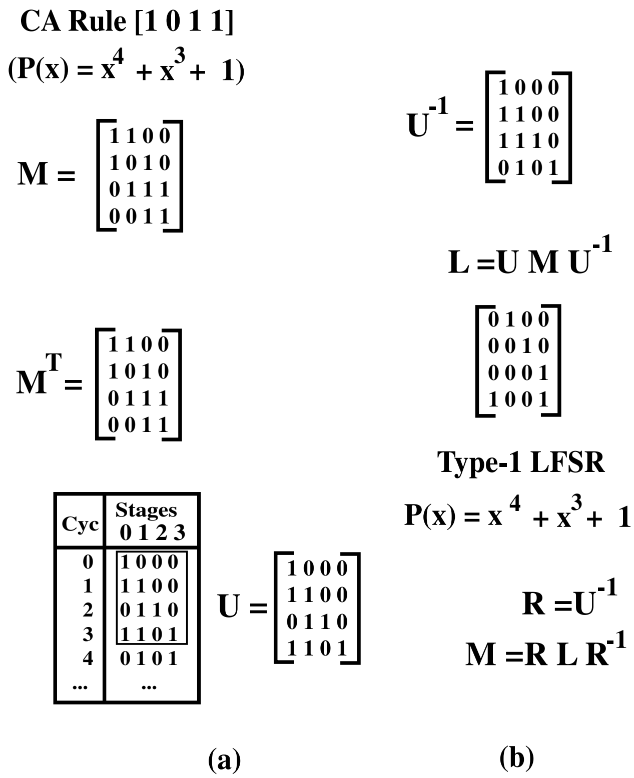


Fig. 4: Illustration of the method for computing a similarity matrix for a CA.

As can be verified (Fig. 4(b)), U is indeed a similarity matrix with respect to the Type–1 LFSR with the same characteristic polynomial and transition matrix L. The required matrix R such that $M = RLR^{-1}$ is then $R = U^{-1}$.

Indicative timings on a SUN Blade 1000 on the computation of similarity matrices by the old and the new method for LFSMs (versions of CA) of various sizes are given in Table 1. All timings are in seconds. The effect of the reduction from the $O(n^6)$ to the $O(n^3)$ complexity is obvious.

Table 1

Timings for computing similarity matrices.

LFSM size	Old	New
20	0.10	≈0.0
25	0.31	≈0.01
30	0.74	≈0.01
35	1.58	≈0.01
40	3.05	0.02
50	9.02	0.03
100	290.78	0.24

5 Conclusions

We showed how the computation of a formula for determining linear dependencies in an extended LFSM structure can be sped up by finding the similarity matrix required in the formula in $O(n^3)$ time instead of $O(n^6)$. This allows for faster search of an appropriate LFSM structure for pseudoexhaustive and/or pseudorandom TPG.

References

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

[2] P. H. Bardell, W. H. McAnney, J. Savir, *Built–in Test for VLSI*, Wiley, New York, 1987.

[3] P.H. Bardell, "Calculating the effects of Linear Dependencies in m–Sequences as Test Stimuli," *IEEE Transactions on CAD/ICAS*, vol. 11, no. 1, pp. 83–85, 1992.

[4] Z. Barzilai, D. Coppersmith, A. L. Rosenberg, "Exhaustive Bit Generation with Application to VLSI Self–Testing," *IEEE Transactions on Computers*, vol. C-32, pp. 190–194, 1983.

[5] C. L. Chen, "Linear Dependencies in Linear Feedback Shift Registers", *IEEE Transactions on Computers*, vol. 35, pp. 1086–1088, Dec. 1986.

[6] A. Gill, *Linear Sequential Circuits*, McGraw–Hill, New York, 1966.

[7] W.-B. Jone, C. A. Papachristou, "A Coordinated Approach to Partitioning and Test Pattern Generation for Pseudoexhaustive Testing", *26th ACM/IEEE Design Automation Conference* , pp. 525–530, 1989.

[8] D. Kagaris, "Linear Dependencies in Extended LFSMs," *IEEE Transactions on CAD/ICAS*, vol. 21, n. 7, pp. 852–858, July 2002.

[9] D. Kagaris, F. Makedon, S. Tragoudas, "A Method for Pseudo–Exhaustive Test Pattern Generation," *IEEE Transactions on CAD/ICAS*, vol. 13, no. 9, pp. 1170–1178, 1994.

[10] D. Kagaris, S. Tragoudas, "Avoiding Linear Dependencies in LFSR Test Pattern Generators," *Journal of Electronic Testing: Theory and Applications*, vol. 6, pp. 229–241, 1995.

[11] A. Lempel, M. Cohn, "Design of Universal Test Sequences for VLSI," *IEEE Trans. on Information Theory*, vol. 31, no. 1, pp. 10–15, 1985.

[12] J. Rajski, P. Tyszer. "On Linear Dependencies in Subspaces of LFSR–Generated Sequences," *IEEE Transactions on Computers*, vol. 45, no. 10, pp. 1212–1221, 1996.

[13] R. Srinivasan, S. K. Gupta, M. A. Breuer. "Novel Test Pattern Generators for Pseudoexhaustive Testing," Proc. International Test Conference, 1993, pp 17-21.

[14] H. S. Stone, *Discrete Mathematical Structures and Their Applications*, Science Research Associates, Chicago, IL, 1973.

[15] D. T. Tang, C. L. Chen, "Logic Test Pattern Generation Using Linear Codes," *IEEE Trans. on Computers*, vol. 33, no 9, pp. 845–850, 1984.

[16] H.-J. Wunderlich, S. Hellebrand, "Tools and devices supporting the pseudo-exhaustive test", *Proc. of the European Design Automation Conference*, pp. 13–17, 1990.