Proceedings IEEE International Test Conference, Baltimore, Maryland, 1992

GENERATION OF VECTOR PATTERNS THROUGH RESEEDING OF MULTIPLE-POLYNOMIAL LINEAR FEEDBACK SHIFT REGISTERS

Sybille Hellebrand*, Steffen Tarnick**, Janusz Rajski***
and Bernard Courtois

TIM3/IMAG
46, avenue Félix Viallet
38031 GRENOBLE CEDEX, France

Abstract. *In this paper we perform a comparative analysis of the encoding efficiency of BIST schemes based on reseeding of single polynomial LFSR's as well as LFSR's with fully programmable polynomials. Full programmability gives much better encoding efficiency. For a testcube with s carebits we need only s+4 bits in contrast to s+19 bits for reseeding of single polynomials, but since it involves solving systems of nonlinear equations it is not applicable to realistic cases. We propose a new BIST scheme where the generator can operate according to a number of primitive polynomials. The testcubes are encoded as the polynomial identifier and a seed. We present models of the encoding efficiency of this scheme and demonstrate, both theoretically and through extensive simulations, that such a scheme with 16 polynomials approaches the efficiency of the scheme based on full polynomial programmability, essentially preserving the computational simplicity of single reseeding.*

1. Introduction

One of the necessary properties that a Built-In Self Test (BIST) scheme must satisfy in order to guarantee a high quality of testing is high fault coverage. Complete, or very high fault coverage is expected to be produced by a simple vector generator in an acceptable number of patterns. To address this problem a number of techniques have been developed assuming varying amount of structural information about the circuit, different fault models, different techniques to characterize compact test sets, and various Design For Testability (DFT) methodologies. These techniques can be classified in three major groups: exhaustive testing, weighted random testing and mixed-mode vector pattern generation.

*) now with the University of Siegen, Germany
**) now with Max-Planck-Society, Fault Tolerant Computing Group, Berlin, Germany
***) on sabbatical leave from McGill University, Montréal, Canada

In exhaustive testing the main objective is to generate all combinations of vectors for every output [6, 5, 15, 16, 19]. Although this strategy yields the coverage of all combinational faults, it is not always possible to achieve; specifically it is not possible to generate all combinations of inputs in circuits with outputs driven by a large number of inputs.

The approach based on weighted random patterns characterizes complete test sets in a compact form by signal probabilities for every input of the circuit [3, 17, 7]. Pseudorandom patterns biased to these probabilities maximize the coverage of hard to test faults, reducing the length of the test. Different faults may require different, conflicting values of signal probabilities [18]. This, combined with the objective to keep the length of the test set within reasonable limits, makes the use of multiple sets of weights necessary, and thus leads to a large volume of test data that has to be stored and manipulated [12].

The other group of methods uses mixed-mode generation of vector patterns. Linear feedback shift registers (LFSR's), or other generators, are used as a source of pseudorandom vectors which cover a large percentage of easily testable faults. Automatic test pattern generation is used to target random pattern resistant faults. In this way partially specified test patterns are generated. The second part of the testing experiment is then designed to cover those hard to test faults. There have been a number of techniques developed to generate deterministic sequences of vectors [1,9, 2, 10, 13]. Most of these techniques do not take advantage of the fact that the test vectors are given in the form of testcubes with many unspecified inputs. They are therefore not suitable for scan design.

Reseeding of LFSR's has been proposed as a technique that is compatible with scan design. The same LFSR used to generate pseudorandom patterns, is loaded with seeds from which it produces vectors that cover the testcubes of difficult to test faults. Although the number of bits required to encode a testcube in this way is much smaller

than the length of the scan chain, it was estimated based on the analysis of linear dependences in LFSR sequences that the LFSR should have the length of s+20 bits in order to reduce the probability of not finding a seed for a testcube with s specified bits to less than 10^{-6} [12, 8].

The objective of this paper is to examine the effectiveness of encoding for schemes based on reseeding and polynomial programming, and to find a scheme that offers the best encoding efficiency with the lowest computational complexity.

In the scheme based on reseeding we assume that one characteristic polynomial is used to control the operation of the LFSR. The seeds can be easily computed by solving systems of linear equations selected by the "care", or specified bits of testcubes. The scheme has the lowest computational complexity. However, the encoding efficiency of this scheme is strongly reduced by linear dependences that exist between some positions of LFSR sequences. As a result we require s+19 bits to encode a testcube with s specified bits with probability 1 - 10^{-6}.

An alternative approach is to use a fully programmable LFSR loaded initially with a unique single seed. In this case we can achieve the same probability of successful encoding using only an (s+4)-bit LFSR. This is the most efficient encoding but the scheme is also computationally the most complex as it involves solving systems of nonlinear equations. In fact, the process of finding the polynomials is computationally so complex that it makes this scheme inapplicable to practical cases.

Therefore we propose to use a scheme based on reseeding and multiple primitive polynomials. In this scheme the LFSR can operate according to one out of many primitive polynomials. The testcube is encoded as the polynomial identifier and the initial seed. We demonstrate that with 16 polynomials, which require 4 bits to encode the choice, this scheme achieves the same probability of finding the encoding as the scheme with full polynomial programmability. Encoding a given testcube involves solving systems of linear equations for the polynomials. Although we have 16 polynomials the process stops when we find the first encoding. In practice the average number of polynomials that are analyzed is only slightly greater than one.

The paper introduces the schemes, presents the theoretical models of their efficiency as well as the experimental validation of the models. The organization of the paper will be as follows. After this introductory section the proposed scheme will be presented in section 2, which also provides the notation used throughout this paper and some basic properties of LFSR sequences. Section 2 will be concluded with the precise problem statement. Subsequently the schemes based on full polynomial programmability, single reseeding and reseeding for multiple polynomials will be examined in sections 3 through 5. Each section will provide a probabilistic model for the respective scheme and a detailed analysis of the encoding efficiency and computational complexity. Finally section 6 shortly describes the experimental validation of our results.

2. The BIST Scheme and Problem Statement

2.1. General Structure of the BIST Scheme

The basic structure of the BIST scheme analyzed in this paper is shown in Figure 1. We assume that a scan based design for testability technique is used. The approach relies on pseudorandom patterns that cover most of the faults in the Circuit Under Test (CUT). For the remaining c hard to test faults m-bit testcubes are determined by automatic test pattern generation. Each of the testcubes is then encoded in an n-bit word that controls the operation of the vector generator. A part of the n-bit word is used to provide the seed, the remaining bits encode the polynomial. In order to generate one test vector corresponding to one testcube the generator selects the successive word from the memory, loads the seed into the LFSR, and establishes the feedback links. Subsequently, in m clock cycles it produces serially the bits of the test vector which are shifted into the scan chain. The test vector is then applied to the CUT, the responses are loaded back into the scan register and shifted out for compaction into the signature register SR.

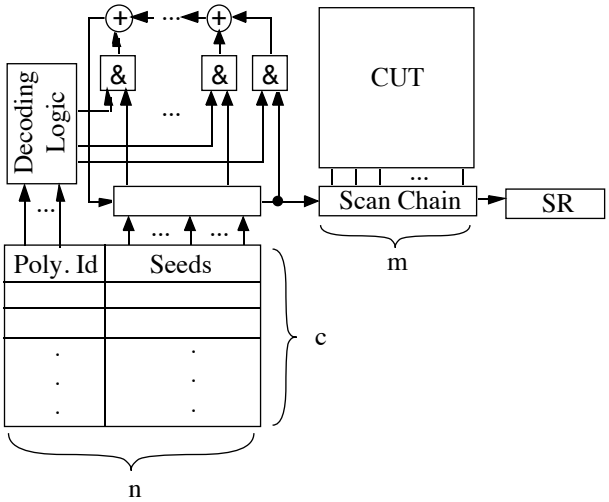

Figure 1. The general structure of the BIST scheme.

The main problem addressed in this paper is how to encode the testcubes into seeds and polynomials to achieve the best encoding efficiency with the least computational complexity. Before we precisely formulate the problem, let us introduce some basic notions and properties of LFSR sequences.

2.2. Testcubes and LFSR's - Basic Definitions

In this section the problem of generating a given testcube by a k-stage LFSR is characterized by a set of equations, and the classical approaches to solve these equations are discussed. First, we introduce the notation used throughout this paper and repeat some basic properties of LFSR sequences.

An LFSR as shown in Figure 2 is represented by its feedback polynomial $h(X) := X^k + \sum_{i=0}^{k-1} h_i X^i$.

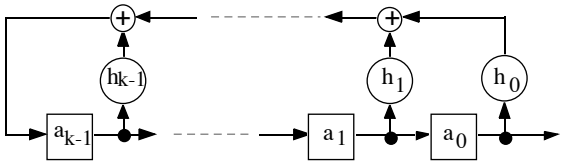

Figure 2. Linear feedback shift register (LFSR).

The output sequence will be denoted by $a := (a_i)_{i\geq 0}$. It is completely determined by the feedback polynomial h(X) and the seed $a(0) := (a_0,\ldots,a_{k-1})$. The t-th state vector $a(t) := (a_t,\ldots,a_{t+k-1})$ can be obtained from the seed by

$$a(t) = a(0) \cdot M^t , \qquad \text{(I)}$$

where $M := \begin{pmatrix} 0 & \ldots & 0 & h_0 \\ 1 & 0 & \ldots & 0 & h_1 \\ & & & & \\ 0 & & \ldots & 0\ 1 & h_{k-1} \end{pmatrix}$ is the companion matrix of the polynomial h(X).

A testcube is respresented by a vector $C := (c_0,\ldots,c_{m-1}) \in \{0,1,x\}^m$. $S(C) := \{i \mid c_i \neq x\}$ denotes the set of specified bits and $s(C) := |S(C)|$ the number of specified bits in the testcube. A k-stage LFSR can generate a given testcube, if its output sequence is "*consistent*" with the testcube.

Definition 1: Let $C = (c_0,\ldots,c_{m-1}) \in \{0,1,x\}^m$ be a testcube and $a = (a_i)_{i\geq 0}$ be a sequence of elements of GF(2). If $c_i = a_i$ holds for all $i \in S(C)$ then a is called *consistent* with C.

To design LFSR sequences which are consistent with a given testcube, equation (I) can be used to derive a set of equations for the feedback coefficients and the initial contents of the LFSR.

Observation 1: Let $C = (c_0,\ldots,c_{m-1}) \in \{0,1,x\}^m$ be a testcube. The output sequence $(a_i)_{i\geq 0}$ produced by an LFSR with feedback polynomial h(X), companion matrix M and seed a(0) is consistent with C, if and only if

$$c_i = a_i = (a(0)\cdot M^i)_1 = (a(0)\cdot M^{i-k+1})_k \qquad \text{(II)}$$

holds for all $i \in S(C)$, where $(a(0)\cdot M^i)_r$ denotes the r-th component of $a(0)\cdot M^i$.
It is easily verified that the equations (II) provide a system of s(C) nonlinear equations in the variables $a_0,\ldots,a_{k-1}$ and $h_0, \ldots, h_{k-1}$. Basically there are two approaches possible to reduce the number of variables involved in this system of equations. The first is to assume a fixed seed and to determine the feedback polynomial. The second approach is to calculate suitable seeds for a fixed feedback polynomial ("reseeding") [12]. Both approaches can be realized in the general structure of the BIST scheme introduced in the paper. There are, however, considerable differences between both approaches with respect to the computational effort required. For a fixed seed, the resulting equations in the variables $h_0 ,\ldots, h_{k-1}$ are still nonlinear, whereas for a fixed feedback polynomial a system of linear equations in the variables $a_0, \ldots, a_{k-1}$ is obtained.

Example 1: A testcube $C := (x,x,1,0,x,0,x) \in \{0,1,x\}^7$ is to be generated by a 3-stage LFSR. The output sequence depends on the seed $a(0) = (a_0,a_1,a_2)$ and the feedback polynomial $h(X) = X^3 + h_2X^2 + h_1X + h_0$. The companion matrix of h(X) is $M = \begin{pmatrix} 0 & 0 & h_0 \\ 1 & 0 & h_1 \\ 0 & 1 & h_2 \end{pmatrix}$. To obtain the equations for a_i, $i \in S(C) = \{2,3,5\}$, the matrices

$M^2 = \begin{pmatrix} 0 & h_0 & h_0\cdot h_2 \\ 0 & h_1 & h_0+h_1\cdot h_2 \\ 1 & h_2 & h_1+h_2 \end{pmatrix}$ and

$M^3 = \begin{pmatrix} h_0 & h_0\cdot h_2 & h_0\cdot h_1+h_0\cdot h_2 \\ h_1 & h_0+h_1\cdot h_2 & h_1+h_0\cdot h_2+h_1\cdot h_2 \\ h_2 & h_1+h_2 & h_0+h_2 \end{pmatrix}$ are computed.

The resulting system of equations is:

(i) $1 = a_2$,
(ii) $0 = a_0\cdot h_0 + a_1\cdot h_1 + a_2\cdot h_2$,
(iii) $0 = a_0\cdot(h_0\cdot h_1+h_0\cdot h_2) + a_1\cdot(h_1+h_0\cdot h_2+h_1\cdot h_2) + a_2\cdot(h_0+h_2)$.

A fixed seed a(0) = (1,1,1) provides a system of equations

(i) $1 = a_2$,
(ii) $0 = h_0 + h_1 + h_2$,
(iii) $0 = h_0\cdot h_1 + h_1 + h_1\cdot h_2 + h_0+h_2 = h_0\cdot h_1 + h_1\cdot h_2$,

which has two solutions $(h_0,h_1,h_2) = (0,0,0)$ and $(h_0,h_1,h_2) = (1,0,1)$.

In contrast, a fixed feedback polynomial $h(X) = X^3 + X + 1$ yields a system of linear equations

(i) $1 = a_2$,
(ii) $0 = a_0 + a_1$,
(iii) $0 = a_0 + a_1 + a_2$,

which has no solution. This is due to the fact that the equations are linearly dependent and demand contradictory values. In general, linear independence of the equations is a sufficient but not a necessary condition for the existence of

a solution. For example the same system of linearly dependent equations has a solution for $a_2 = 1$, $a_3 = 1$, $a_5 = 0$. □

Summarizing we can state that the computational effort required for the reseeding technique is considerably less than the computational effort required for calculating a feedback polynomial. But on the other hand calculating the feedback polynomial seems to offer better chances of finding a solution and therefore possibly a higher efficiency of encoding. For a more exact comparison of both schemes we need a technique to evaluate their properties concerning the efficiency of encoding and the computational complexity. We will develop such a technique for the more general scheme which is presented in the next section.

2.3. Generalized Reseeding Based on Multiple Polynomials

In the following we present a generalized encoding scheme which offers a wide range of trade-offs between single reseeding and full polynomial programmability. The basic idea is to consider not one, but a number of primitive polynomials when calculating seeds. Moreover, for a part of the seed fixed values can be assumed. Then the computational effort is determined by the number of polynomials: in the worst case for each polynomial a system of linear equations has to be solved. And, as it will be shown later, the efficiency of encoding can be considerably increased already for a relatively small number of polynomials taken into account.

A generalized BIST scheme based on reseeding of multiple polynomials is shown in Figure 3.

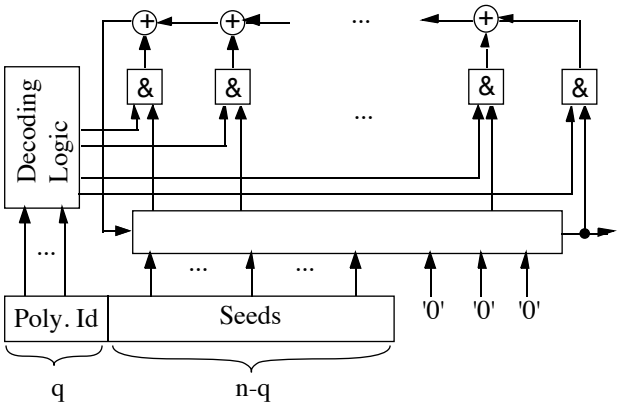

Figure 3. Generalized BIST scheme based on reseeding of multiple polynomials

The testcubes are encoded into n bits of information, where q bits are used to select one out of 2^q (primitive) polynomials of degree $k \geq \max(n - q,q)$, and n-q bits to store the programmable part of the seed.

Example 2: Consider the case from Example 1. Now it is assumed that the seed has one fixed bit $a_2 = 1$ and the testcube C = (x,x,1,0,x,0,x) is to be generated either by the polynomial $h^1(X) := X^3 + X + 1$ or by $h^2(X) := X^3 + X^2 + 1$. Thus we have n = k = 3 and q = 1. As demonstrated in Example 1 the system of equations corresponding to $h^1(X)$ has no solution. But for $h^2(X)$ the resulting equations are

(i) $1 = a_2$,
(ii) $0 = a_3 = a_0 + a_2$,
(iii) $0 = a_5 = a_0 + a_1$,

and $a_0 = a_1 = a_2 = 1$ is a solution. □

For q = k = n the scheme of Figure 3 becomes fully programmable and for k = n and q = 0 we obtain single reseeding. Between these two extremes there is a range of schemes determined by the choice of the parameter q. In the sequel we will develop a technique to evaluate the efficiency of encoding for varying parameters. As the evaluation will be based on the probability of finding a solution we need a probabilistic model for the generalized reseeding scheme, which describes this probability as a function of the involved parameters.

Problem Statement: Given a testcube C with s specified bits and an LFSR that can implement 2^q (primitive) polynomials of degree k. Determine the probability $P_{succ(k,n,q,s)}$ that for a testcube C with s(C) = s and for at least one of 2^q (primitive) polynomials of degree k there is a seed (n-q programmable bits), such that the output sequence of the LFSR is conistent with C.

These probabilities will also characterize the classical approaches described in section 2.2, since $P_{succ}(k,k,0,s) =: P_{seed(k,s)}$ is the probability that for a testcube C with s(C) = s and a fixed polynomial of degree k there is a suitable seed. And $P_{succ}(k,k,k,s) =: P_{pol(k,s)}$ is the probability that for a testcube C with s(C) = s and a fixed seed there is a suitable feedback polynomial of degree k. The corresponding probabilities of failure will be denoted by $P_{fail(k,n,q,s)} := 1 - P_{succ}(k,n,q,s)$, $P_{noseed(k,s)} := 1 - P_{seed}(k,s)$ and $P_{nopol(k,s)} := 1 - P_{pol}(k,s)$.

As linear independence of the equations is a sufficient (but not a necessary condition) for the existence of a seed in the classical reseeding scheme another objective of our probabilistic analysis is to determine the probabilities $P_{indep(k,s)}$ that for a testcube C with s(C) = s and a fixed feedback polynomial of degree k the resulting equations for the seed variables are linearly independent, and $P_{dep(k,s)} := 1 - P_{indep}(k,s)$.

3. Full Polynomial Programmability

In this section a complete analysis of the BIST scheme based on calculating polynomials is given. Since in this case a fixed seed is assumed the general scheme introduced in Figure 3 reduces to the scheme shown in Figure 4.

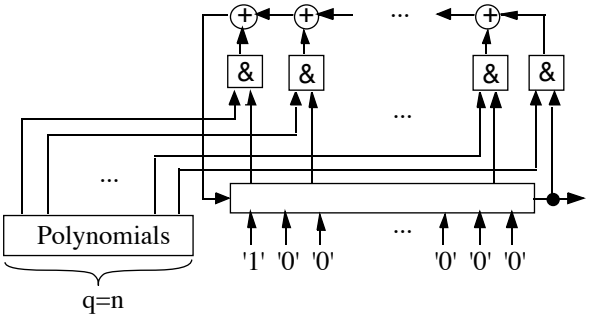

Figure 4. BIST scheme based on calculating feedback polynomials.

The properties of this scheme are characterized by the probabilities $P_{pol}(k,s)$ and $P_{nopol}(k,s)$, which are determined by the following theorem.

Theorem 1: Let $C \in \{0,1,x\}^m$ be a testcube with $s(C) = s$, and let $2k \le m$. If a fixed seed $a(0) = (0,\ldots,0,1)$ is assumed, then the probability $P_{nopol}(k,s)$ is given by

$$P_{nopol}(k,s) = \left(\frac{2^m - 2^k}{2^m}\right)^{2^{m-s}}.$$

Proof: There are 2^k different feedback polynomials of degree k. For the fixed seed $a(0) = (0,\ldots,0,1)$ the corresponding LFSR's produce 2^k different output sequences. Since $2k \le m$ the projection onto the first m bits provides 2^k different sequences of length m. Thus there are $2^m - 2^k$ sequences of length m which cannot be output sequences of a k-stage LFSR using the seed a(0). Therefore the probability that an arbitrary sequence of length m is not a desired LFSR sequence is $\frac{2^m - 2^k}{2^m}$. On the other hand, there are 2^{m-s} sequences of length m which are consistent with the testcube C. C cannot be generated by a k-stage LFSR, if none of this sequences is a desired LFSR sequence. Assuming statistical independence this completes the proof. □

For some representative values of s and k the values for $P_{nopol}(k,s)$ are shown in Table I.

For practical applications an approximation formula can be used to calculate the probability $P_{nopol}(k,s)$.

Corollary 1: For large m the probability $P_{nopol}(k,s)$ is given by $P_{nopol}(k,s) \approx (e^{-1})^{2^{k-s}}$.

Proof: By Theorem 1 $P_{nopol}(k,s) = \left(\frac{2^m - 2^k}{2^m}\right)^{2^{m-s}} = \left(\left(1 - \frac{1}{2^{m-k}}\right)^{2^{m-k}}\right)^{2^{k-s}}$. Since $\lim_{x\to\infty}\left(1 - \frac{1}{x}\right)^x = e^{-1}$, $P_{nopol}(k,s) \approx (e^{-1})^{2^{k-s}}$ for sufficiently large m. □

s \ k	20	30	40	50	60
20	0.367880	0.000000	0.000000	0.000000	0.000000
25	0.969233	0.000000	0.000000	0.000000	0.000000
30	0.999024	0.367879	0.000000	0.000000	0.000000
35	0.999969	0.969233	0.000000	0.000000	0.000000
40	0.999999	0.999024	0.367879	0.000000	0.000000
45	1.000000	0.999969	0.969233	0.000000	0.000000
50	1.000000	0.999999	0.999024	0.367879	0.000000
55	1.000000	1.000000	0.999969	0.969233	0.000000
60	1.000000	1.000000	0.999999	0.999024	0.367879

Table I. Theoretical values for $P_{nopol}(k,s)$.

Since 2^m grows quickly with m, the formula derived in Corollary 1 gives a very precise approximation even for relatively small values of m. For practical applications, the BIST scheme is therefore fully characterized by the approximation formula. As a consequence we can observe the following facts.

Observation 2: The size m of the testcube has no significant influence on the probability $P_{nopol}(k,s)$.

Observation 3: The probability $P_{nopol}(k,s)$ only depends on the difference k-s between the degree of the polynomial and the number of care bits.

Observation 4: Increasing the degree of the polynomial by one results in squaring the probability of failure. $P_{nopol}(k+1,s) = P_{nopol}(k,s-1) = (e^{-1})^{2^{k-s+1}} = \left((e^{-1})^{2^{k-s}}\right)^2 = (P_{nopol}(k,s))^2$.

These tradeoffs for varying parameters s and k can be seen very clearly in Figure 5, where for fixed k the values of $P_{nopol}(k,s)$ are shown as a function of the number of care bits. Increasing the degree of the polynomial corresponds to simply shifting the curve to the right, which clearly reflects Observations 2 and 3. The steepness of the curves is a consequence of Observation 4.

Summarizing the results of this section we can characterize the efficiency of encoding for this BIST scheme as follows: If we require that $10^{-6} \ge P_{nopol}(k,s) = (e^{-1})^{2^{k-s}}$, then this can be obtained by an LFSR with feedback polynomial of degree $k \ge s + \log_2(-\ln(10^{-6})) \approx s + 4$.

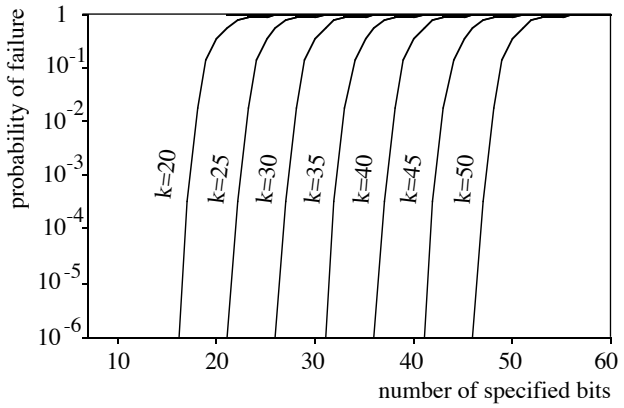

Figure 5. Theoretical values of $P_{nopol}(k,s)$.

4. Reseeding of Single Polynomials

The objective of this section is a complete analysis of the classical reseeding approach. The corresponding BIST scheme shown in Figure 6 is obtained as a specialization of the general scheme introduced in section 2.3.

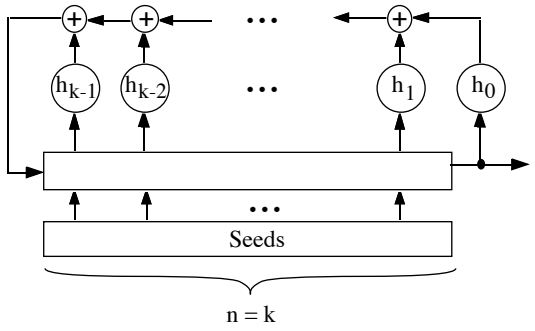

Figure 6. BIST scheme based on reseeding of single polynomials.

The properties of the single reseeding scheme are described by the probabilities $P_{seed}(k,s)$ and $P_{noseed}(k,s)$. Since the linear independence of the equations (II) is a sufficient condition for finding a seed we obviously have $P_{noseed}(k,s) \le P_{dep}(k,s)$ and $P_{seed}(k,s) \ge P_{indep}(k,s)$. Therefore we first concentrate on these bounds.

Linear dependences have been extensively studied for the purposes of exhaustive testing and of random testing [5, 15, 16, 8, 4]. With respect to our objectives especially the probabilistic analysis carried out by Chen is of major interest [8]. He showed that for primitive polynomials the probabilities for linear dependences in the LFSR sequence only depend on the degree k of the polynomial and the number of considered positions in the sequence.

Theorem 2 (Chen): Let h(X) be a primitive polynomial of degree k and let $C \in \{0,1,x\}^m$ be a testcube with s(C) = s. Then $P_{dep}(k,s) = 1 - \prod_{j=0}^{s-1} \frac{2^k - 2^j}{2^k - j - 1}$ holds.

To determine $P_{seed}(k,s)$ and $P_{noseed}(k,s)$ we consider the process of generating the equations (II) as a Markov chain $(X_t)_{1 \le t \le s}$ over a set of states $S = \{0,1,\ldots,k\}$. At step t the t-th equation is generated and for $1 \le d \le t \le s$ the equality $X_t = d$ is interpreted as follows: The system of t equations generated so far has rank d and there is a solution for this system. $X_t = 0$ means that the system has no solution. The Markov chain is described by its initial distribution and the transition probabilities $P(X_{t+1} = d' \mid X_t = d)$.

Theorem 3: Let $C \in \{0,1,x\}^m$ be a testcube with s(C) = s, h(X) a primitive polynomial of degree k and let $2^k - 1 \ge m$. The Markov chain $(X_t)_{1 \le t \le s}$ as defined above has the initial distribution $P(X_1 = 1) = 1$, $P(X_1 = d) = 0$ for $d \ne 1$, and the transition probabilities are given by

$$P(X_{t+1} = d+1 \mid X_t = d) = \begin{cases} \frac{2^k - 2^d}{2^k - 1 - t} & \text{for } 1 < d+1 \le k \\ 0 & \text{otherwise} \end{cases},$$

$$P(X_{t+1} = d \mid X_t = d) = \begin{cases} \frac{1}{2} \cdot \frac{2^d - 1 - t}{2^k - 1 - t} & \text{for } d>0 \text{ and } t+1 \le 2^d - 1 \\ 0 & \text{otherwise} \end{cases},$$

$P(X_{t+1} = 0 \mid X_t = d) = P(X_{t+1} = d \mid X_t = d)$ for $d > 0$ and $P(X_{t+1} = 0 \mid X_t = 0) = 1$. All other transition probabilities are zero.

Proof: Let E_t denote the system of t equations generated up to step t.

If $X_t = 0$, then the system E_t has no solution. Therefore adding an additional equation at time t+1 provides a system E_{t+1} which also cannot have a solution. Therefore $P(X_{t+1} = 0 \mid X_t = 0) = 1$ holds.

If $X_t = d \ne 0$, the system E_t has rank d, and a solution exists. Since the rank is d, $2^d - 1$ different linear combinations can be obtained from the t equations in E_t. $2^d - 1 - t$ of them are not contained in E_t and the total number of possible equations which are not contained in E_t is $2^k - 1 - t$. Assuming that the $2^d - 1 - t$ linearly dependent equations are uniformly distributed within the $2^k - 1 - t$ possible equations, a fraction of them, i.e.

$$\frac{m - t}{2^k - 1 - t} \cdot (2^d - 1 - t)$$

corresponds to equations for the testcube. Therefore we have

$$m - t - \frac{m - t}{2^k - 1 - t} \cdot (2^d - 1 - t)$$

linearly inpendent equations out of m - t possible equa-

tions which correspond to equations for the testcube and are not contained in E_t. The generation of an additional equation at step t+1 therefore has one of the following implications:

(i) For $d < k$ a system E_{t+1} of rank d+1 is obtained with probability

$$\frac{m - t - \frac{m - t}{2^k - 1 - t} \cdot (2^d - 1 - t)}{m - t} = \frac{2^k - 2^d}{2^k - 1 - t} .$$

For this system the existence of a solution is also guaranteed. For $d = k$ the rank can no longer be increased and therefore we have the formula for $P(X_{t+1} = d+1 \mid X_t = d)$.

(ii) With probability $1 - \frac{2^k - 2^d}{2^k - 1 - t} = \frac{2^d - 1 - t}{2^k - 1 - t}$ the rank of the new system E_{t+1} remains d. The existence of a solution is no longer guaranteed. Since the LFSR produces a pseudorandom sequence, the probability that the additional equation does not introduce a contradiction can be assumed as $\frac{1}{2}$. □

The structure of the Markov chain $(X_t)_{1 \le t \le s}$ is represented by the graphic shown in Figure 7. Transitions to the state d = 0 are omitted.

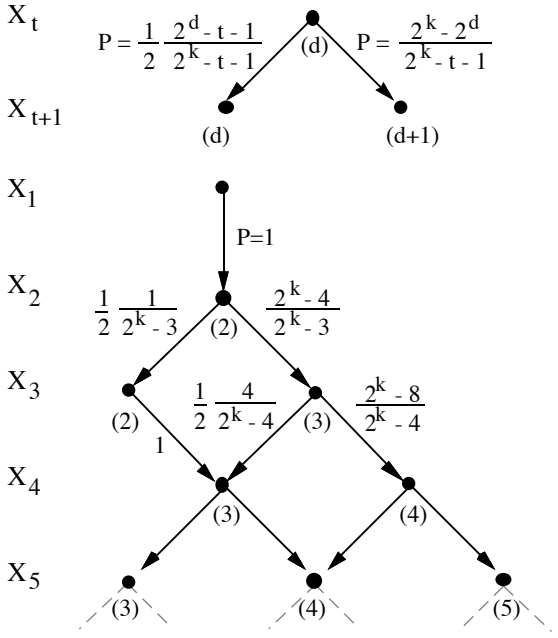

Figure 7. Structure of the Markov chain $(X_t)_{1 \le t \le s}$.

In terms of this Markov model the probability $P_{seed}(k,s)$ is obtained as $P(X_s \neq 0)$. Moreover, as the rightmost branch in Figure 7 corresponds to generating linearly independent equations, the probability $P_{indep}(k,s)$ is obtained as $P(X_s = s)$. Obviously the following corollary to Theorem 3 holds.

Corollary 2: Let $C \in \{0,1,x\}^m$ be a testcube with $s(C) = s$, $h(X)$ a primitive polynomial of degree k and let $(X_t)_{1 \le t \le s}$ be the Markov chain defined above. Then

$$P_{seed}(k,s) = \sum_{d=1}^{s} P(X_s = d) = \sum_{d=\lceil \log_2(s+1) \rceil}^{\min(k,s)} P(X_s = d).$$

The probabilities $P(X_s = d)$ can be determined recursively using $P(X_t = d) = P(X_{t-1} = d) \cdot P(X_t = d \mid X_{t-1} = d) + P(X_{t-1} = d-1) \cdot P(X_t = d \mid X_{t-1} = d-1)$.

Theorem 3 relies on two assumptions. The first is that the period $2^k - 1$ of the LFSR sequence is bigger than the length m of the testcube, but for practical applications where we have for example $k \ge 16$ and $m = 1000$ this is no restriction. Secondly it is assumed that the equations which are linearly dependent on the system E_t are uniformly distributed within the set of all equations not contained in E_t. If this second assumption is not made, the probabilities $P_{noseed}(k,s)$ may become dependent on the length of the testcube. Within the scope of this paper we cannot discuss this problem in more detail, but it will be a topic of further research.

In Table II the values for $P_{noseed}(k,s)$ are listed for some representative values of s and k.

s \ k	20	30	40	50	60
20	0.389634	0.000488	0.000000	0.000000	0.000000
25	0.969234	0.015503	0.000015	0.000000	0.000000
30	0.999024	0.389678	0.000488	0.000000	0.000000
35	0.999969	0.969234	0.015503	0.000015	0.000000
40	0.999999	0.999024	0.389678	0.000488	0.000000
45	1.000000	0.999969	0.969234	0.015503	0.000015
50	1.000000	0.999999	0.999024	0.389678	0.000488
55	1.000000	1.000000	0.999969	0.969234	0.015503
60	1.000000	1.000000	0.999999	0.999024	0.389678

Table II. Theoretical values for $P_{noseed}(k,s)$.

From this table and the graphical representation shown in Figure 8 two interesting observations can be made.

Observation 5: The probability $P_{noseed}(k,s)$ mainly depends on the difference k-s between the degree of the polynomial and the number of care bits.

Observation 6: For k = s the value of $P_{noseed}(k,s)$ is always close to $0.389678 \approx e^{-1}$.

Both observations have been confirmed by calculating the values of $P_{noseed}(k,s)$ for $5 \le k,s \le 60$. For a more general analysis the upper bound $P_{dep}(k,s)$ is used as a rough appoximation for $P_{noseed}(k,s)$. For large values of $k >> s$ a simplified formula for $P_{dep}(k,s)$ can be derived as a corollary to Theorem 2.

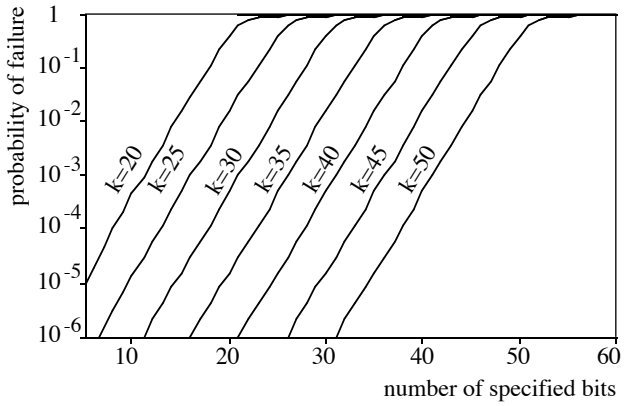

Figure 8. Theoretical values for $P_{noseed}(k,s)$.

Corollary 3: Let h(X) be a primitive feedback polynomial of degree k and let s << k. Then for large k the probability $P_{dep}(k,s)$ is given by $P_{dep}(k,s) \approx 2^{s+1-k}$.

Proof: By Theorem 2 $P_{dep}(k,s) = 1 - \prod_{j=0}^{s-1} \frac{2^k - 2^j}{2^k - j - 1}$.

For large values of k $2^{-k} \approx 0$ can be assumed. This yields $\prod_{j=0}^{s-1} \frac{2^k - 2^j}{2^k - j - 1} = \prod_{j=0}^{s-1} \frac{1 - 2^{j-k}}{1 - (1 - j)\cdot 2^{-k}} \approx \prod_{j=0}^{s-1} (1 - 2^{j-k}) =$

$(1 - 2^{-k})\cdot(1 - 2^{1-k})\cdot(1 - 2^{2-k})\cdot\ldots\cdot(1 - 2^{s-k}) \approx 1 - (2^{1-k} + 2^{2-k} + \ldots + 2^{s-k}) = 1 - 2^{1-k}\cdot(2^s - 1) \approx 1 - 2^{s+1-k}$. □

Corollary 3 also provides the following observation, which can be regarded as an analogon to Observation 4.

Observation 7: Increasing the degree of the polynomial by one results in reducing the probability $P_{noseed}(k,s)$ by half. For k >> s we have $P_{noseed}(k+1,s) \approx P_{noseed}(k,s-1) \approx 2^{s-k} = \frac{1}{2}(2^{s-k+1}) \approx \frac{1}{2}P_{noseed}(k,s)$.

Comparing these results to the results of the previous section we see that both for full polynomial programmability and for single reseeding the probabilities for not finding a solution only depend on the difference k-s between the degree k of the polynomial and the number s of carebits. However, increasing the degree of the polynomial leads to an exponential decrease of $P_{nopol}(k,s)$, whereas for single reseeding only a linear decrease of $P_{noseed}(k,s)$ can be achieved.

10^{-6} is used again as an upper bound for the probability of failure. It is known from the previous section, that $10^{-6} \geq P_{nopol}(k,s)$ if $k \geq s + 4$. Using the approximation formula of Corollary 2 we obtain $10^{-6} \geq P_{noseed}(k,s)$, if $2^{s-k+1} \approx P_{noseed}(k,s) \leq 10^{-6}$, i.e. if $k \geq s + 1 - \log_2(10^{-6}) \approx s + 21$. In fact, calculating the exact values for $P_{noseed}(k,s)$, it can be observed that $10^{-6} \geq P_{noseed}(k,s)$, if $k \geq s + 19$.

5. Multiple Polynomials

The results of the previous sections have confirmed our conjecture that calculating the feedback polynomial offers a considerably higher efficiency of encoding than the classical reseeding approach. In general, however, the method is not feasible for practical applications because of the high computational effort required. In this section the generalized reseeding scheme introduced in section 2.3 will be analyzed, and it will be shown that with this scheme the encoding efficiency of calculating polynomials can be achieved by the computational effort required for the classical reseeding technique.

The properties of this general scheme are measured by the probabilities $P_{succ}(k,n,q,s)$ and $P_{fail}(k,n,q,s) = 1 - P_{succ}(k,n,q,s)$. Since k - n + q bits of the seed are fixed, there are k - n + q additional equations determining the output sequence $(a_i)_{i\geq 0}$ of the LFSR. Therefore $P_{succ}(k,n,q,s)$ can be determined as the probability $P_{seed}(k,q,s+k-n+q)$, where $P_{seed}(k,q,s)$ denotes the probability that for a testcube C with s(C) = s there is a seed for at least one of 2^q primitive polynomials of degree k, such that the resulting output sequence is consistent with C. Assuming that calculating seeds for different primitive polynomials corresponds to statistically independent events, the probability $P_{seed}(k,q,s)$ can be determined using the results of the previous section.

Theorem 4: Let $C \in \{0,1,x\}^m$ be a testcube with s(C) = s and $q \leq \log_2\left(\frac{\phi(2^k-1)}{k}\right)$, where ϕ denotes Euler's function. Then $P_{noseed}(k,q,s) := 1 - P_{seed}(k,q,s) = P_{noseed}(k,s)^{2^q}$.

Proof: The number of primitive polynomials of degree k is $\frac{\phi(2^k-1)}{k}$. As $q \leq \log_2\left(\frac{\phi(2^k-1)}{k}\right)$, it is guaranteed that there are $2^q \leq \frac{\phi(2^k-1)}{k}$ different primitive polynomials and thus 2^q statistically independent events. □

Consequently the probability $P_{fail}(n,k,q,s)$ can be determined as $P_{fail}(n,k,q,s) = P_{noseed}(k,s+k-n+q)^{2^q}$.

Similarly as in the previous two paragraphs an approximation formula can be used to examine the tradeoffs for varying parameters k, q and s. Using the approximation $2^{s-k+1} \approx P_{noseed}(k,s)$ derived in the previous section for large k >> s Theorem 4 yields $P_{noseed}(k,q,s) \approx (2^{s-k+1})^{2^q}$ and $P_{fail}(n,k,q,s) \approx (2^{s-n+q+1})^{2^q}$ for n >> s+q.

This formula shows that increasing the parameter q results in a slightly increased probability of failure for reseeding with respect to the single polynomials, but this is overcompensated by the decrease due to the growing number of

polynomials. In fact for increasing values of q and n = k the probability $P_{fail}(k,k,q,s)$ converges very quickly to the asymptotic value of $P_{nopol}(k,s)$. This is illustrated by Table III for n = k = 40. The columns show the differences $P_{fail}(40,40,q,s)$ - $P_{nopol}(40,s)$ for q = 0, …, 4. The asymptotic value is practically reached already for q = 4.

s \ q	0	1	2	3	4
20	0.000000	0.000000	0.000000	0.000000	0.000000
22	0.000002	0.000000	0.000000	0.000000	0.000000
24	0.000008	0.000000	0.000000	0.000000	0.000000
26	0.000031	0.000000	0.000000	0.000000	0.000000
28	0.000122	0.000000	0.000000	0.000000	0.000000
30	0.000488	0.000001	0.000000	0.000000	0.000000
32	0.001951	0.000015	0.000000	0.000000	0.000000
34	0.007782	0.000240	0.000001	0.000000	0.000000
36	0.030766	0.003670	0.000190	0.000006	0.000000
38	0.099112	0.030327	0.004742	0.000936	0.000210
40	0.021799	0.004613	0.001052	0.000261	0.000062

Table III. Differences $P_{fail}(40,40,q,s)$ - $P_{nopol}(40,s)$ for $0 \le q \le 4$.

Figure 9 elucidates this effect showing the probability $P_{fail}(n,k,q,s)$ as a function of s for fixed parameters n = k and q. Clearly the largest gain is obtained by changing q from 0 to 1, and for q = 4 the asymptotic behavior is practically achieved.

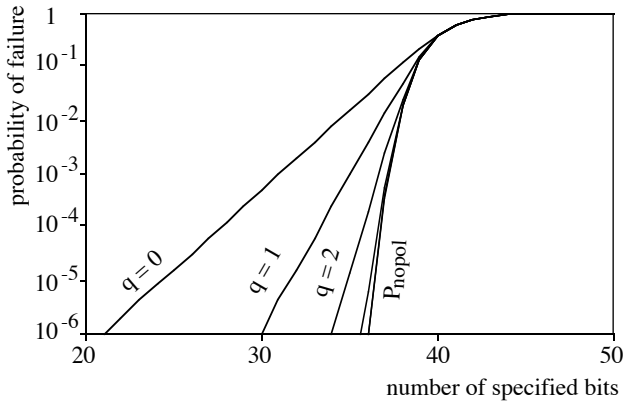

Figure 9. $P_{fail}(40,40,q,s)$ as a function of the number s of specified bits for q = 0,…,4.

With respect to the computational complexity it was already pointed out in section 2.3 that the generalized reseeding scheme involves in the worst case solving 2^q systems of linear equations. With the probabilistic model developed in this section we are able to carry out an analysis of the average case computational complexity. The average number of systems of linear equations that must be treated will be denoted by Q(n,k,q,s).

Theorem 5: In the average case the generalized reseeding scheme involves solving Q(n,k,q,s) =

$$\frac{1 - P_{noseed(n,s+k-n+q)}^{2^q-1}}{1 - P_{noseed(n,s+k-n+q)}} + P_{noseed(n,s+k-n+q)}^{2^q-1}$$

systems of linear equations.

Proof: Let $P := P_{noseed(n,s+k-n+q)}$. Then the probability that t systems of linear equations have to be treated is $P^{t-1}\cdot(1 - P)$ for $t < 2^q$ and P^{2^q-1} for $t = 2^q$. Therefore the average number of attempts is

$$\sum_{t=1}^{2^q-1} t\cdot P^{t-1}\cdot(1 - P) + 2^q\cdot P^{2^q-1} = \sum_{r=1}^{2^q-1}\sum_{t=r}^{2^q-1} P^{t-1}\cdot(1 - P) +$$

$$2^q\cdot P^{2^q-1} = \frac{1 - P^{2^q-1}}{1 - P} - (2^q - 1)\cdot P^{2^q-1} + 2^q\cdot P^{2^q-1}. \qquad \square$$

If 2^q-1 is large then $Q(n,k,q,s) \approx \frac{1}{1 - P_{noseed(n,s+k-n+q)}}$. Using the approximation formula from Corollay 2 for large n >> s+q the average computational complexity can be estimated as $\frac{1}{1 - 2^{s-n+q}}$.

This allows us to fully determine the tradeoff between the achievable efficiency of encoding and the average computational complexity. In Figure 10 this is done for n = k = 40 and s = 36.

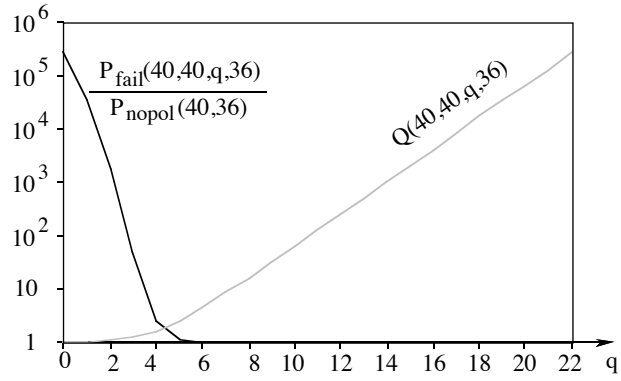

Figure 10. Tradeoff between the efficiency of encoding and the average computational complexity for n = k = 40 and s = 36.

The straight line represents the ratio $P_{fail}(40,40,q,36)/P_{nopol}(40,36)$ and the dotted line the average computational complexity Q(40,40,q,36), both as functions of the parameter q. Obviously Q(40,40,q,36) as well as the ratio $P_{fail}(40,40,q,36)/P_{nopol}(40,36)$ are close to one for q = 4, and thus the efficiency of encoding of full polynomial progammability is practically reached with a computational effort comparable to that required for single reseeding.

6. Experimental Validation

To validate the assumptions made in sections 3 and 4 we performed a series of experiments based on a Monte-Carlo simulation. For a randomly generated set of 10000 testcubes we determined by exhaustive search the number of suitable polynomials and suitable seeds respectively for 8 ≤ k ≤ 16 and 5 ≤ s ≤ 20. Comparing the results to the corresponding numbers predicted by our theoretical models we observed only small differences. These observations were confirmed by a confidence test for which we divided each experiment into 10 smaller experiments with 1000 testcubes each [14].

To validate the assumption that calculating seeds for different polynomials corresponds to statistically independent events we took 10 different polynomials of degree 10 and determined for each pair of polynomials and varying parameter s the relative frequencies that for none of both polynomials a suitable seed exists. Comparing the mean values to our theoretical values $P_{noseed}(10,2,s)$ only small differences could be observed. A detailed description of these and further experiments can be found in [11].

7. Conclusion

In this paper we presented a new pattern generator for BIST. The generator is based on an LFSR capable of implementing a number of primitive polynomials. The polynomials are selected by controlling the feedback links in the LFSR. We demonstrated that with approximately 16 polynomials the generator can produce sequences with the phenomenon linear dependence practically eliminated. This property is utilized to enhance the encoding efficiency of testcubes for difficult to test faults. Very high probability of successfull encoding is achieved with s + 4 bits for a testcube with s specified bits. 4 bits are used to identify one of 16 polynomials and s bits for the seed. Because of the reduced linear dependences other interesting applications of this scheme are seen in the area of pseudo-exhaustive testing.

Acknowledgement

This research was supported by grants from the French Foreign Ministry, the French Ministry of Research and Technology (MRT) and the French National Center of Scientific Research (CNRS).

References

1 V. K. Agarwal, E. Cerny: Store and Generate Built-in-Testing Approach; Proc. IEEE 11th Int. Symposium on Fault-Tolerant Computing, FTCS-11, 1981, pp. 35-40

2 S. B. Akers, W. Jansz: Test Set Embedding in a Built-in Self-Test Environment; Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 257-263

3 P. Bardell, W. H. McAnney, J. Savir: Built-in Test for VLSI; Wiley-Interscience, New York, 1987

4 P. H. Bardell: Design Considerations for Parallel Pseudorandom Pattern Generators; Journal of Electronic Testing: Theory and Applications, Vol. 1., No. 1, February 1990, pp. 73-87

5 Z. Barzilai, D. Coppersmith, A. L. Rosenberg: Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing; IEEE Trans. on Comp., Vol. C-32, No. 2, February 1983, pp. 190-194

6 E. J. McCluskey, S. Bozorgui-Nesbat: Design for Autonomous Test; IEEE Trans. on Circuits and Systems, Vol. Cas-28, No. 11, November 1981, pp. 1070-1078

7 F. Brglez et al.: Hardware-Based Weighted Random Pattern Generation for Boundary-Scan; Proc. IEEE Int. Test Conf., Washington D.C., 1989, pp. 264 - 274

8 C. L. Chen: Linear Dependencies in Linear Feedback Shift Registers; IEEE Trans. on Comp., Vol. C-35, No. 12, December 1986, pp. 1086-1088

9 W. Daehn: Deterministische Testmustergenerierung für den eingebauten Selbsttest von integrierten Schaltungen; Grossintegration, NTG-Fachberichte 82, Baden-Baden 1983

10 C. Dufaza, G. Cambon: LFSR based Deterministic and Pseudo-Random Test Pattern Generator Structures; Proc. European Test Conference, Munich, 1991, pp. 27-34

11 S. Hellebrand, S. Tarnick, J. Rajski, B. Courtois: Generation of Vector Patterns Through Reseeding of Multiple Polynomial LFSR's; TIM3 Research Report, Grenoble, March 1992

12 B. Koenemann: LFSR-Coded Test Patterns for Scan Designs; Proc. European Test Conference, Munich, 1991, pp. 237-242

13 J. L. Massey: Shift Register Synthesis and BCH Decoding; IEEE Trans. on Information Theory, Vol. IT-15, No. 1, January 1969, pp. 122-127

14 R. L. Masson, R. F. Gunst, J. L. Hess: Statistical Design and Analysis of Experiments, John Wiley & Sons, 1989

15 D. T. Tang, C. L. Chen: Logic Test Pattern Generation Using Linear Codes; IEEE Trans. on Comp., Vol. C-33, No. 9, September 1984, pp. 845-850

16 E. J. McCluskey, L. T. Wang: Circuits for Pseudo-Exhaustive Test Pattern Generation; Proc. IEEE Int.Test Conf., 1986, pp. 25-37

17 H.-J. Wunderlich: Self Test Using Unequiprobable Random Patterns; Proc. IEEE 17th International Symposium on Fault-Tolerant Computing, FTCS-17, Pittsburgh 1987

18 H.-J. Wunderlich: Multiple Distributions for Biased Random Test Patterns; Proc. IEEE Int. Test Conf., Washington D.C., 1988, pp. 236-244

19 H.-J. Wunderlich, S. Hellebrand: The Pseudo-Exhaustive Test of Sequential Circuits; IEEE Trans. on CAD of Integrated Circuits and Systems, January 1992