Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers

Sybille Hellebrand, *Member, IEEE,* Janusz Rajski, *Member, IEEE,* Steffen Tarnick, *Member, IEEE,* Srikanth Venkataraman, *Student Member, IEEE,* and Bernard Courtois

***Abstract*— In this paper, we propose a new scheme for Built-In Test (BIT) that uses Multiple-polynomial Linear Feedback Shift Registers (MP-LFSR's). The same MP-LFSR that generates random patterns to cover easy to test faults is loaded with seeds to generate deterministic vectors for difficult to test faults. The seeds are obtained by solving systems of linear equations involving the seed variables for the positions where the test cubes have specified values. We demonstrate that MP-LFSR's produce sequences with significantly reduced probability of linear dependence compared to single polynomial LFSR's. We present a general method to determine the probability of encoding as a function of the number of specified bits in the test cube, the length of the LFSR and the number of polynomials. Theoretical analysis and experiments show that the probability of encoding a test cube with s specified bits in an s-stage LFSR with 16 polynomials is $1–10^{-6}$. We then present the new BIT scheme that allows for an efficient encoding of the entire test set. Here the seeds are grouped according to the polynomial they use and an implicit polynomial identification reduces the number of extra bits per seed to one bit. The paper also shows methods of processing the entire test set consisting of test cubes with varied number of specified bits. Experimental results show the tradeoffs between test data storage and test application time while maintaining complete fault coverage.**

***Index Terms*— Built-In Test, hardware test pattern generators, input test data compression and decompression, multiple-polynomial LFSR, reseeding, scan design.**

I. Introduction

ONE of the necessary properties that a Built-In Test (BIT) 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. Enormous practical importance of this problem prompted a lot of research that resulted in a number of techniques developed to address the problem 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.

The techniques for hardware test pattern generation can be classified in the following major groups: exhaustive testing [1]–[3], pseudorandom testing [4], weighted random testing [5]–[7], hardware pattern generators of deterministic tests [8]–[10] and mixed-mode test pattern generation [11]–[14]. These techniques offer different tradeoffs between the test data storage, test application time, hardware overhead, fault coverage and programmability. Their properties have been studied extensively in the literature and an extensive review of the techniques and tradeoffs they offer can be found in [4].

Koenemann [12] proposed a very attractive technique for encoding test data based on intelligent reseeding of single-polynomial LFSR's. This technique is compatible with scan and offers reduced storage requirements, shorter test application time, and smaller area overhead compared to weighted random patterns [12]. The same LFSR is used to generate pseudorandom and deterministic patterns which are encoded as seeds obtained from the test cubes of difficult to test faults. Although the number of bits required to encode a test cube in this way is much smaller than the length of the scan chain, it was estimated, based on the analysis of linear dependencies 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 test cube with s specified bits to less than 10^{-6} [15], [12].

In this paper we introduce a new encoding scheme based on reseeding of Multiple-polynomial LFSR's. Since an MP-LFSR can operate according to one out of many primitive polynomials a linear dependency existing between test cube positions for one polynomial may be overcome by another polynomial. A test cube is encoded as the polynomial identifier and the initial seed. We demonstrate that with 16 polynomials, which require only $4 + s$ bits to encode the choice and the seed, this scheme achieves the same probability of finding the encoding as an $s + 19$ bit single-polynomial LFSR . Encoding a given test cube in an MP-LFSR involves solving systems of linear equations for the polynomials. Although there are 16 polynomials the process stops when the first encoding is

Manuscript received November 28, 1993; revised April 20, 1994. This work was supported by grants from the French Foreign Ministry, the French Ministry of Research and Technology (MRT), the French National Center of Scientific Research (CNRS), in parts by strategic grant MEF0045788 from the Microelectronics Fund of the Natural Science and Engineering Research Council of Canada, and the ESPRIT project 7107 ARCHIMEDES.

S. Hellebrand is with the Institute for Computer Structures, University of Siegen, 57068 Siegen, Germany.

J. Rajski is with Mentor Graphics Corporation, Wilsonville, OR 97070 USA.

S. Tarnick is with Max Planck Society, Fault-Tolerant Computing Group at the University of Potsdam, 14415 Potsdam, Germany.

S. Venkataraman is with the Center for Reliable and High-Performance Computing, C&SRL Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

B. Courtois is with INPG/TIMA Laboratory, 38031 Grenoble, France.

IEEE Log Number 9407126.

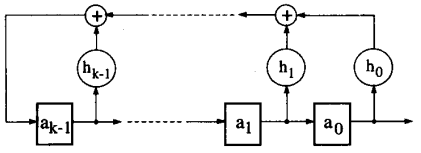


Fig. 1. Linear feedback shift register (LFSR).

found. As a result the average number of polynomials that are analyzed is less than 2.

Finally we introduce a new BIT scheme that efficiently encodes complete test sets. In this scheme all the seeds that use the same polynomial are applied in one contiguous sequence. The generator has a counter that determines the current polynomial. Every seed is presented with one additional bit which indicates when to change the polynomial. This way if all the test cubes had the same number of specified bits equal s, every test cube could be encoded with $s+1$ bits. In reality, however, the number of specified bits in test cubes differs significantly from one test cube to another. We discuss various techniques, like cube merging and concatenation, to reduce that variation. The paper also contains the results of experiments performed on benchmark circuits demonstrating the effectiveness of the BIT scheme.

II. Encoding of Test Cubes—Basic Concepts

The problem of generating a given test cube by a k-stage LFSR is characterized by solving a system of equations derived as follows: A linear feedback shift register as shown in Fig. 1 is represented by its feedback polynomial $h(X) = X^k = \Sigma_{i=0}^{k-1} h_i X^i$. The output sequence denoted by $a = (a_i)_{i\geq 0}$ is completely determined by the feedback polynomial $h(X)$ and the seed vector $(a_0, \cdots, a_{k-1})$. Obviously a given test cube $C = (c_0, \cdots, c_{m-1}) \in \{0,1,x\}^m$ with specified bit positions $S(C) := \{i = 0, \cdots, m-1 | c_i \neq x\}$ can be generated by the LFSR if the output sequence is covered by the test cube, i.e., $a_i = c_i$ holds for all $i \in S(C)$. Applying the feedback equations $a_i = \Sigma_{j=0}^{k-1} h_j a_{i-k+j}$ recursively to the equations $a_i = c_i$ finally provides a system of nonlinear equations depending only on the seed variables $a_0, \cdots, a_{k-1}$ and the feedback coefficients $h_0, \cdots, h_{k-1}$.

Basically there are two approaches possible to reduce the number of variables involved in this system of equations. The first approach 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 encoding scheme introduced in the paper. There are, however, considerable differences between these approaches with respect to the computational effort required. For a fixed seed, the resulting equations in the variables $h_0, \cdots, h_{k-1}$ are still nonlinear, whereas for a fixed feedback polynomial a system of linear equations in the variables $a_0, \cdots, a_{k-1}$ is obtained. The problem is that for both approaches the existence of a solution is not guaranteed. Linear dependencies in the LFSR sequence may for example lead to an unsolvable system of linear equations for the reseeding approach.

In the sequel we will present a method to determine how the probability of finding an encoding depends on the number of specified bits in a test cube, the length of the LFSR and the number of polynomials. The understanding of this relation is crucial in the design of effective BIT schemes. A generalized encoding scheme based on reseeding of multiple polynomials is shown in Fig. 2. In this scheme every m-bit test cube is 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.

In order to generate an m-bit test vector corresponding to the encoded test cube the seed is loaded into the LFSR, and the feedback links determined by the polynomial identifier are established. Subsequently, in m clock cycles the LFSR produces serially the bits of the test vector which are shifted into the scan chain. The test vector is then applied to the circuit under test, the responses are loaded back into the scan register and shifted out for compaction.

For $q = k = n$ the scheme of Fig. 2 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. The evaluation of the efficiency of encoding for various parameters will be based on the probability of finding a solution. A probabilistic model will be developed to solve the following problem.

Problem Statement: Given a test cube C with s specified bits and an LFSR that can implement 2^q (primitive) polynomials of degree k with an $(n-q)$ bit seed. Determine the probability $P_{\text{succ}}(k,n,q,s)$ that for at least one of the polynomials there is a seed ($n-q$ programmable bits), such that the output sequence of the LFSR is covered by C, and the probability $P_{\text{fail}}(k,n,q,s) := 1 - P_{\text{succ}}(k,n,q,s)$.

These probabilities will also characterize the classical approaches described above, since $P_{\text{succ}}(k,k,0,s) := P_{\text{seed}}(k,s)$ is the probability that for a test cube C and a fixed polynomial of degree k there is a suitable seed. And $P_{\text{seed}}(k,k,k,w) := P_{\text{pol}}(k,s)$ is the probability that for a test cube C and a fixed seed there is a suitable feedback polynomial of degree k, such that the output sequence of the LFSR is covered by C. The corresponding probabilities of failure will be denoted by $P_{\text{noseed}}(k,s) := 1 - P_{\text{seed}}(k,s)$ and $P_{\text{nopol}}(k,s) := P_{\text{pol}}(k,s)$.

As linear independence of the equations $a_i = c_i$ 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 $P_{\text{indep}}(k,s)$ that for a test cube C and a fixed feedback polynomial of degree k the resulting equations for the seed variables are linearly independent, and $P_{\text{dep}}(k,s) := 1 - P_{\text{indep}}(k,s)$.

III. LFSR's with Programmable Feedback Polynomials

In this section the analysis of the encoding scheme based on calculating polynomials is given. Since in this case a fixed seed is assumed the properties of the encoding scheme

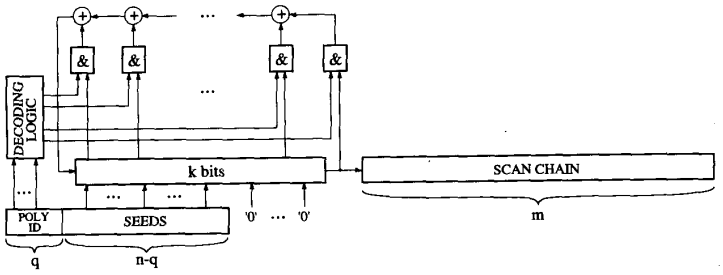


Fig. 2. Generalized encoding scheme based on reseeding of multiple polynomials.

are characterized by the probability $P_{\text{pol}}(k,s)$, $P_{\text{nopol}}(k,s)$ respectively.

Theorem 1: Let $C \in \{0,1,x\}^m$ be a test cube with s specified bits, and let $2k \leq m$. If a fixed seed $(a_0,\cdots,a_{k-1})$ is assumed, then the probability $P_{\text{nopol}}(k,s)$ is given by

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

Proof: There are 2^k different feedback polynomials of degree k. For the fixed seed (0, $\cdots$, 0, 1) the corresponding LFSR's produce 2^k different output sequences. Since $2k \leq m$ 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. Therefore the probability that an arbitrary sequence of length m is not a desired LFSR sequence is $(2^m - 2^k)/2^m$. On the other hand, there are 2^{m-s} sequences of length m which are covered by the test cube C. C cannot be generated by a k-stage LFSR, if none of these sequences is a desired LFSR sequence. Assuming statistical independence this completes the proof. □

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

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 encoding scheme is therefore fully characterized by the approximation formula. As a consequence we can observe the following facts.

Observation 1: The size of the test cube m has no impact on the probability $P_{\text{nopol}}(k,s)$.

Observation 2: The probability $P_{\text{nopol}}(k,s)$ only depends on the difference $k-s$ between the degree of the polynomial and the number of care bits.

Observation 3: It is easily verified that increasing the degree of the polynomial by one results in squaring the probability of failure. $P_{\text{nopol}}(k+1,s) = (P_{\text{nopol}}(k,s))^2$.

These tradeoffs for varying parameters s and k can be seen very clearly in Fig. 3, where for fixed k the values of $P_{\text{nopol}}(k,s)$ 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 1 and 2. The steepness of the curves is a consequence of Observation 3.

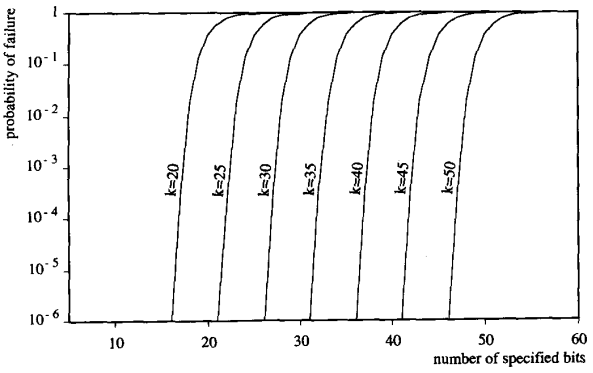


Fig. 3. Theoretical values for $P_{\text{nopol}}(k,s)$.

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

IV. Reseeding of LFSR's with Single Polynomials

The properties of the scheme relying on a single feedback polynomial are quantified by the probabilities $P_{\text{seed}}(k,s)$ and $P_{\text{noseed}}(k,s)$. Since the linear independence of the equations $a_i = c_i$ is a sufficient condition for finding a seed we obviously have $P_{\text{noseed}}(k,s) \leq P_{\text{dep}}(k,s)$ and $P_{\text{seed}}(k,s) \geq P_{\text{indep}}(k,s)$. Therefore we first concentrate on these bounds.

Linear dependencies have been extensively studied for the purposes of exhaustive testing and of random testing [1], [2], [15], [16]. With respect to our objectives especially the probabilistic analysis carried out by Chen is of major interest [15].

To determine $P_{\text{seed}}(k,s)$ and $P_{\text{noseed}}(k,s)$ we consider the process of generating the equations $a_i = c_i$ as a Markov chain $(X_t)_{1\leq t\leq s}$ over a set of states $S = \{0,1,\cdots,k\}$. At step t the tth equation is generated and for $1 \leq d \leq t \leq s$ 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 $P(X_{t+1} = d' | X_t = d)$.

Theorem 2: Let $C \in \{0,1,x\}^m$ be a test cube with s specified bits, $h(X)$ a primitive polynomial of degree k and

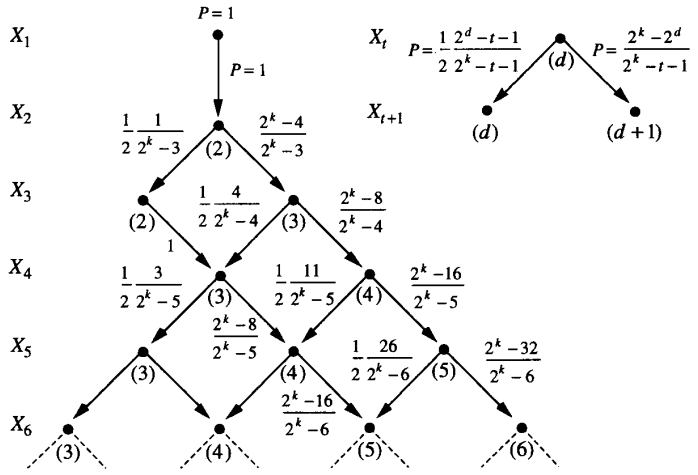


Fig. 4. Structure of the Markov chain $(X_t)_{1\le t\le s}$.

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 $d \neq 1$, and the transition probabilities are given by

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

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

$P(X_{t+1} = 0 | X_t = d) = P(X_{t+1} = d | X_t = d)$ for $d > 0$ and $P(X_{t+1} = 0 | X_t = 0) = 1$. 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 $P(X_{t+1} = 0 | X_t = 0) = 1$ holds.

If $X_t = d \neq 0$, 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 test cube. Therefore we have

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

linearly independent equations out of $m-t$ possible equations which correspond to equations for the test cube and are not contained in E_t. The generation of an additional equation at step $t+1$ therefore has one of the following implications:

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

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

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 $P(X_{t+1} = d+1 | X_t = d)$.

2) With probability $1 - (2^k - 2^d)/(2^k - 1 - t) = (2^d - 1 - t)/(2^k - 1 - t)$ 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 1/2. □

The structure of the Markov chain $(X_t)_{1\le t\le s}$ is represented by the graph shown in Fig. 4. Transitions to the state $d = 0$ are omitted.

In terms of this Markov model the probability $P_{\text{seed}}(k, s)$ is obtained as $P(X_s \neq 0)$. Moreover, as the rightmost branch in Fig. 4 corresponds to generating linearly independent equations, the probability $P_{\text{indep}}(k, s)$ is obtained as $P(X_s = s)$. Obviously the following corollary to Theorem 2 holds.

Corollary 2: Let $C \in \{0, 1, x\}^m$ be a test cube with s specified bits, $h(X)$ 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

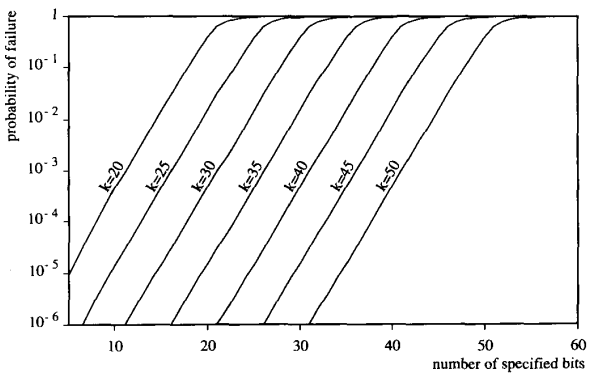


Fig. 5. Theoretical values for $P_{\text{noseed}}(k, s)$.

using

$$P(X_t = d) = P(X_{t-1} = d) \cdot P(X_t = d | X_{t-1} = d) + P(X_{t-1} = d-1) \cdot P(X_t = d | X_{t-1} = d-1).$$

Theorem 2 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 test cube, but for practical applications where we have for example $k \geq 16$ 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 $P_{\text{noseed}}(k, s)$ may become dependent on the length of the test cube. Within the scope of this paper we cannot discuss this problem in more detail.

Fig. 5 shows the values of $P_{\text{noseed}}(k, s)$ as function of s for some representative values of k. This graphical representation and the exact calculation of the values $P_{\text{noseed}}(k, s)$ provides two interesting observations.

Observation 4: The probability $P_{\text{noseed}}(k, s)$ mainly depends on the difference $k - s$ between the degree of the polynomial and the number of care bits.

Observation 5: For $k = s$ the value of $P_{\text{noseed}}(k, s)$ is always close to $0.389678 \approx e^{-1}$.

For a more general analysis the upper bound $P_{\text{dep}}(k, s)$ is used as a rough approximation for $P_{\text{noseed}}(k, s)$. For large values of $k \gg s$ a simplified formula for $P_{\text{dep}}(k, s)$ can be derived as a corollary to theorem on the probability of linear dependencies given by Chen [15].

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

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

Observation 6: Increasing the degree of the polynomial by one results in reducing the probability $P_{\text{noseed}}(k, s)$ by half. For $k \gg s$ we have $P_{\text{noseed}}(k+1, s) \approx P_{\text{noseed}}(k, s-1) \approx 2^{s-k} = 1/2(2^{s-k+1}) \approx 1/2 P_{\text{noseed}}(k, s)$.

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 care bits. However, when considering both probabilities in logarithmic scale, increasing the degree of the polynomial leads to an exponential decrease of $P_{\text{nopol}}(k, s)$, whereas for single reseeding only a linear decrease of $P_{\text{noseed}}(k, s)$ can be achieved.

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

V. Multiple-Polynomial LFSR's

In this section the generalized reseeding scheme introduced in Section II 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 $P_{\text{succ}}(k, n, q, s)$ and $P_{\text{fail}}(k, n, q, s) = 1 - P_{\text{succ}}(k, n, q, s)$. 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 $P_{\text{succ}}(k, n, q, s)$ can be determined as the probability $P_{\text{seed}}(k, q, s + k - n + q)$, where $P_{\text{seed}}(k, q, s)$ denotes the probability that for a test cube C with s specified bits there is a seed for at least one of 2^q primitive polynomials of degree k, such that the resulting output sequence is covered by C. Assuming that calculating seeds for different primitive polynomials corresponds to statistically independent events, the probability $P_{\text{seed}}(k, q, s)$ can be determined using the results of the previous section.

Theorem 3: Let $C \in \{0, 1, x\}^m$ be a test cube with s specified bits and $q \leq \log_2(\varphi(2^k - 1)/k)$ where φ denotes Euler's function. Then $P_{\text{noseed}}(k, q, s) := 1 - P_{seed}(k, q, s) = P_{\text{noseed}}(k, s)^{2^q}$.

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

Consequently the probability $P_{\text{fail}}(n, k, q, s)$ can be determined as $P_{\text{fail}}(n, k, q, s) = P_{\text{noseed}}(k, s + k - n + q)^{2^q}$. 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 $2^{s-k+1} \approx P_{\text{noseed}}(k, s)$ derived in the previous section for large $k \gg s$ Theorem 3 yields $P_{\text{noseed}}(k, q, s) \approx (2^{s-k+1})^{2^q}$ and $P_{\text{fail}}(n, k, q, s) \approx (2^{s-n+q+1})^{2^q}$ for $n \gg s + q$.

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 $P_{\text{fail}}(k, k, q, s)$ converges very quickly to the asymptotic value of $P_{\text{nopol}}(k, s)$. The asymptotic value is practically reached

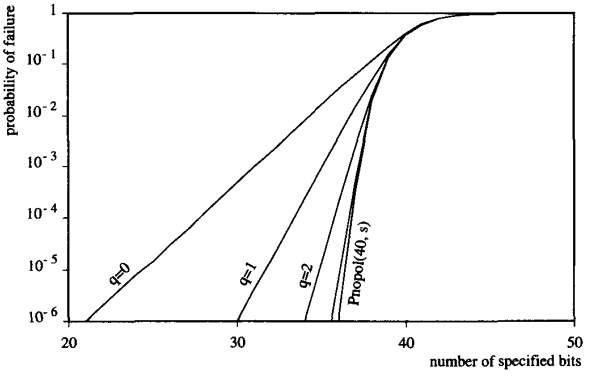


Fig. 6. $P_{\text{fail}}(40, 40, q, s)$ as a function of the number of specified bits for $q = 0, \cdots, 4$.

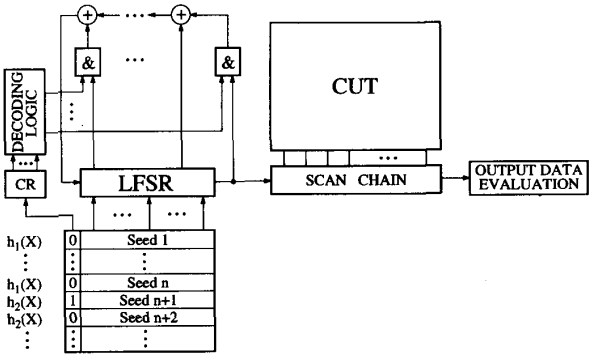


Fig. 7. Conceptual diagram of BIT scheme based on reseeding of a MP-LFSR.

already for $q = 4$. Fig. 6 elucidates this effect showing the probability $P_{\text{fail}}(n, k, q, s)$ 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. Consequently, to keep the probability of not finding an encoding for a test cube with s care bits below 10^{-6} an MP-LFSR of degree s with 16 polynomials is sufficient, and only $s + 4$ bits are required to encode a test cube with s specified bits.

Theorem 3 relates the number of specified bits in a test cube, the length of the MP-LFSR and the number of polynomials to the probability of finding encoding. It is fundamental in the design of MP-LFSR's. Given the maximum number of specified bits in a set of test cubes, Theorem 3 determines the number of different polynomials and their degree to guarantee the required probability of finding encoding.

The generalized reseeding scheme involves, in the worst case, solving 2^q systems of linear equations. The average number of systems of linear equations that must be treated can be easily determined as a function of the P_{noseed} using geometric series [13]. For $k = s$ (the length of the MP-LFSR is equal to the number of specified bits in the test cubes) using the approximation formula from Observation 5 we obtain the average number of systems of equations as $1/(1 - e^{-1}) = 1.58$. Thus the efficiency of encoding of full polynomial programmability is practically reached with a computational effort comparable to that required for single reseeding.

VI. The BIT Scheme and Test Set Processing

As shown in the previous section a single test cube with s specified bits can be encoded with $s+4$ bits using MP-LFSR's. However, when considering an entire test set, the encoding of the required feedback polynomial can be done implicitly by grouping together the seeds for specific polynomials and using a "next-bit" to indicate whether the feedback polynomial has to be changed (Fig. 7). Thus the number of bits required to encode a test cube with s specified bits further reduces to $s+1$.

In this scheme the feedback polynomials are encoded as the states of the q-bit counter CR. A single bit which is appended to each of the seeds instructs the counter when to proceed to its next state, i.e., to change the polynomial. The diagram shown in Fig. 7 illustrates the essential idea which can be implemented in a number of ways depending on the system requirements. The memory may or may not reside on chip. It could be a ROM chip on board or even another board in the system. The seeds can be stored in a system memory and the data can be transferred to the MP-LFSR through boundary scan for decompression. The same scheme can essentially be used with the encoded test cubes being produced by an external tester. To reduce the hardware overhead in the actual implementation of the scheme the MP-LFSR can be broken into two parts. The first, short LFSR, e.g., 32-stage long, is used to generate random patterns. The second part, the extension, used only in reseeding, is built out of the regular scan chain filp-fops of latches.

When encoding an entire set $T = \{C_1, \cdots, C_r\}$ of test cubes further optimization is possible if a preprocessing of the test set is performed before the encoding. The length of the MP-LFSR to encode a set T of test cubes depends on the maximum number $s_{\max}$ of specified bits in any test cube in T. The number of specified bits s_i in any other test cube C_i is in general much smaller than $s_{\max}$. Encoding these test cubes into a word of size $s_{\max} + 1$ results in inefficient encoding of these test cubes. The encoding efficiency E is defined as

$$E = \frac{1}{r s_{\max}} \sum_{i=1}^{r} s_i, \qquad 0 < E \leq 1.$$

This overall efficiency E can be increased by "merging" and "concatenating" the cubes of the original test set T. Merging of test cubes is a widely used technique to compact deterministic test sets. A set of test cubes can be merged to a single test cube of the same length if the test cubes of this set have nonconflicting values in all bit positions where both cubes are specified. To optimize the encoding efficiency the test cubes have to be merged into a minimal number of cubes with the number of care bits not exceeding $s_{\max}$.

This problem can be viewed as a graph theoretical problem by identifying each test cube with a vertex of a graph and connecting two vertices by an edge if the corresponding test cubes are consistent. The cliques in the graph then represent

maximal sets of test cubes which can be merged and known heuristics for partitioning the graph into cliques can be used to solve the problem [17]. We have to take into account the constraint that the number of specified bits in a test cube that is a result of the merging should not exceed $s_{\max}$.

The second technique to increase the overall encoding efficiency E makes use of the fact that the length of an encoded test cube is independent of the length of the original test cube. Thus if a subset $T_i \subseteq T$ of test cubes is concatenated to one long test cube $C(T_i)$ whose number of care bits is not exceeding $s_{\max}$, then encoding the cube $C(T_i)$ requires the same number of bits as each of the original cubes.

To implement the test set T the BIT scheme of Fig. 7 has to be slightly modified. If the subsets T_i consist of at most j m-bit test cubes each, then once the seed for a cube $C(T_i)$ is loaded, the LFSR works in autonomous mode for $j \cdot m$ clock cycles. After each m cycles a test cube is completely loaded into the scan chain and can be applied to the circuit under test. For subsets containing less than j cubes dummy cubes must be inserted to maintain the simplicity of this scheme. Naturally, if test cube concatenation is used the MP-LFSR should not use the scan registers or the seed has to be reloaded.

The necessary size of memory using this approach can therefore be minimized by constructing a minimal number of concatenated cubes which cover the original test set. With $s(C)$ denoting the number of specified bits in a test cube C this problem can be formally stated as follows:

Problem OCT (Optimal Concatenation of Test Cubes): Let $T = \{C_1, \cdots, C_r\}$ be a set of test cubes and $s_{\max} := \max\{s(C_i) | 1 \leq i \leq r\}$ and let j be a positive integer. Partition the set T in a minimal number z of subsets T_i, such that $|T_i| \leq j$ and $s(T_i) = \Sigma_{C \in T_i} s(C) \leq s_{\max}$ holds for all $i = 1, \cdots, z$.

This problem is equivalent to the Bin Packing Problem [17] with an additional constraint on the number of objects that can be packed into a bin. Here the bin size is determined by the LFSR length and the object sizes by the numbers of specified bits in the test cubes.

We use a simple greedy strategy to solve this problem. For each subset T_i we search for a test cube C of T with the maximum number of specified bits such that the two constraints indicated above are still satisfied if we put C into T_i. If there exists such a test cube we take it out of T, put it into T_i and continue the search. If not, we start with a new subset T_{i+1} and continue as described, until T is empty. The algorithm is summarized in Fig. 8.

The process of concatenation attempts to reduce the memory storage size for the encoded test cubes over single cube encoding by increasing the encoding efficiency. An encoding efficiency of $E = 1$ implies storing a single bit in memory (as LFSR seed) per specified bit of test cube. The encoding efficiency may be further improved when an LFSR of size $s^* \geq s_{\max}$ is selected and the number of specified bits in a concatenated cube is bounded by s^*. For a given set of test cubes T and a given maximum number j of cubes allowed to be concatenated together an optimal value of s^* is one that gives the highest encoding efficiency. A method how to predict the optimal LFSR length is given in [18].

Algorithm CONCATENATION

$i := 1$; $T_i := \emptyset$;;

repeat

$s(T_i) := \sum_{C \in T_i} s(C)$;

$Q := \{C \in T \mid s(T_i) + s(C) \leq s_{\max}\}$;

if $Q \neq \emptyset$ and $|T_i| < j$ then

choose $C \in Q$ with $s(C) = \max\{s(C_i) \mid C_i \in Q\}$;

$T_i := T_i \cup \{C\}$; $T := T \setminus \{C\}$;

else $i := i + 1$; $T_i := \emptyset$;

until $T = \emptyset$.

Fig. 8. Algorithm for the concatenation of test cubes.

A possible disadvantage of this scheme over single cube encoding is a possible increase in the testing time. Each dummy cube adds to the testing time. The overall number of such additional cubes is given by $\nu = \Sigma_{i=1}^{z} (j - |T_i|) = j \cdot z - |T|$. The overhead of testing time O_t is defined as

$$O_t = \frac{\nu}{j \cdot z} = \frac{j \cdot z - |T|}{j \cdot z} = 1 - \frac{|T|}{j \cdot z},$$
$$0 \leq O_t \leq 1.$$

To limit this overhead we limit the number j of test cubes we are allowed to concatenate. Varying s^* and j gives a space of solutions with varying encoding efficiencies and time overheads. Depending on the set T this may involve a tradeoff between the two. For our study we bound the time overhead by restricting j to two cubes. Experimental results confirm that this gives good encoding efficiency with a small time overhead.

VII. Experimental Results

To validate our theoretical results and illustrate the design methodology of the BIT scheme based on reseeding of MP-LFSR's we performed experiments with the ISCAS-89 benchmark circuits [19]. Each of the benchmark circuits was initially fault simulated for a varying number of random patterns generated by a 32-stage LFSR with a primitive polynomial. The faults not detected by the random test formed a set of "hard" to test faults. For each fault in this set we generated a test cube using ATPG (SOCRATES [20]). The test cubes of the resulting test set T were then merged as described in Section VI. At this point the length of the MP-LFSR was determined as $s_{\max}$ and 16 polynomials of this length were generated. The analysis presented in Section V relates the encoding efficiency, the complexity of the MP-LFSR and the probability of finding an encoding. The designer using this scheme may however want to reduce the number of polynomials and simplify the MP-LFSR by compromising the probability of finding encoding. Theorem 3 and its graphical illustration on Fig. 6 show a variety of tradeoffs. The choice of 16 polynomials of length $s_{\max}$ illustrated in this section on experimental results ensures high efficiency and probability of encoding.

The test cubes obtained after merging were encoded as seeds of the MP-LFSR and a counter control bit for each

TABLE I
EFFICIENCY OF ENCODING IN A 16-POLYNOMIAL MP-LFSR

0	1 5 3	1 7 0	1 8 3	1 9 4	2 0 1	2 0 6	2 0 9	2 1 1	Average rank	P_{noseed}
1	1	1	1	0	0	1	1	1	211.22	0.37
1	1	1	1	1	1	1	1	0	211.21	0.37
1	1	1	1	1	1	0	1	1	211.18	0.38
1	1	1	1	1	1	1	1	1	211.18	0.38
1	0	1	1	1	1	1	1	1	211.18	0.38
1	1	1	0	0	1	1	1	1	211.17	0.39
1	1	1	1	1	1	1	0	1	211.17	0.35
1	1	1	1	0	1	1	0	1	211.15	0.37
1	1	0	1	1	1	0	1	1	211.15	0.39
1	1	1	0	1	1	1	1	1	211.15	0.41
1	0	1	1	1	0	1	1	1	211.15	0.42
1	0	1	1	1	1	1	1	0	211.14	0.39
1	1	0	1	1	1	1	1	1	211.14	0.40
1	0	1	1	1	1	1	0	1	211.14	0.36
1	0	1	1	1	1	0	1	1	211.14	0.41
1	1	1	1	1	0	1	1	1	211.13	0.38

TABLE II
ENCODING EFFICIENCY AND REDUCTION OF STORAGE REQUIREMENTS FOR VARYING NUMBERS OF RANDOM PATTERNS (AFTER MERGING)

circuit	length of test cubes	# random patterns	# test cubes (merged)	# specified bits (max.)	encoding efficiency	reduction factor
s5378	214	1k	85	128	0.49	1.66
		2k	70	101	0.54	2.10
		5k	44	97	0.51	2.18
		10k	32	97	0.48	2.18
s9234	247	1k	152	126	0.51	1.94
		2k	139	126	0.49	1.94
		5k	116	104	0.55	2.35
		10k	107	95	0.54	2.57
s13207	700	1k	196	263	0.33	2.65
		2k	171	263	0.30	2.65
		5k	135	261	0.30	2.67
		10k	105	261	0.26	2.67
s35932	1763	32	49	253	0.48	6.94
		64	34	245	0.42	7.17
		128	9	111	0.78	15.74
s38417	1664	100k	91	212	0.80	7.81
		150k	91	176	0.77	9.40
		200k	78	176	0.79	9.40
		250k	88	138	0.76	11.97

seed. The seeds are computed by solving systems of linear equations selected by the specified bits of the test cube where the variables are the values of the seed. The Gauss–Jordan elimination with maximum pivot strategy [21] was adopted to handle efficiently Boolean variables by using bit-wise operations [14], [18]. Using this procedure a software package was developed that is capable to process 1000 test cubes with up to 300 specified bits and length of 2000 bits in less than a minute on SPARC 10 workstation.

To keep the hardware requirements of the pattern generator low the feedback polynomials for an LFSR where chosen such that they need only a small number of feedback links each and share as many feedback links as possible. Table I contains 16 polynomials used to encode test cubes for circuit s38417 with $s_{max} = 212$. The first 9 columns show the feedback polynomial, i.e., h_0, h_{153}, h_{170}, h_{183}, h_{194}, h_{201}, h_{206}, h_{201}. These 16 polynomials provide that capability to encode all the 91 test cubes which, combined with 100k other pseudorandom vectors generated the 32-stage LFSR, guarantee complete coverage of testable single stuck-at faults. To demonstrate the encoding capability of this MP-LFSR we analyzed 10 000 random test cubes with 212 specified bits each. The last column contains the frequency of not finding a seed. Notice that this value is very close to the value $0.389678 \approx e^{-1}$ theoretically determined in Observation 5. The probability of not finding encoding for all 16 polynomials is estimated by the product of all these values as $0.222 * 10^{-6}$. This is consistent with the results of Theorem 3.

Column "Average rank" in the Table I shows the average rank of the row-reduced echelon matrix which is produced during the process of solving the systems of linear equations [21]. The average rank can be interpreted as the average number of linearly independent equations. This parameter can be used to evaluate the suitability of polynomials for reseeding. In this example, if the value was 212 the equations would always be linearly independent and the solution would always be guaranteed.

The result of the encoding of the merged cubes is shown in Table II. The length of a test cube for each circuit is defined as the number of primary inputs and the number of flip-flops which together form the scan chain. The reduction factor in the last column of the table gives the ratio of the number of bits that we had to store without and with encoding of the test cubes:

$$\text{reduction factor} := \frac{(\text{length of a test cube}) \cdot (\#\text{test cubes})}{(\max.\#\text{specified bits} + 1) \cdot (\#\text{test cubes})}.$$

As can be seen from Table II the reduction factor tends to increase as the test cube length increases. The encoding scheme is more efficient for larger circuits.

In the second part of the experiments we concatenated the test cubes obtained after merging. We limited the maximum number of test cubes allowed to be concatenated to two. The concatenation process was performed using the algorithm of Fig. 8. The optimal LFSR length for each case was obtained by computing the encoding efficiency for varying values of s^* around the predicted value of s^* and then choosing that value that gives the highest encoding efficiency [18].

The results for the concatenation for selected benchmark circuits are shown in Table III. Column 6 of Table III shows the number of dummy cubes which have to be inserted to keep the scheme simple. As can be seen the time overhead caused by these additional cubes is very low. Similar to merging a reduction factor was computed for the concatenation of test cubes, defined as shown at the bottom of the page.

$$\text{reduction factor} = \frac{(\text{length of an original test cube}) \cdot (\#\text{ test cubes before concatenation})}{(\max.\#\text{specified bits} + 1) \cdot (\#\text{ test cubes after concatenation})}$$

TABLE III
ENCODING EFFICIENCY AND REDUCTION OF STORAGE REQUIREMENTS AFTER CONCATENATION

circuit	length of test cubes	# random patterns	# test cubes (concat.)	# specified bits (max.)	# additional test cubes	encoding efficiency	reduction factor
s5378	428	1k	43	132	1	0.94	3.18
		2k	35	117	0	0.93	3.63
		5k	22	104	0	0.88	4.08
		10k	17	98	2	0.90	4.07
s9234	494	1k	82	127	12	0.94	3.58
		2k	75	127	11	0.91	3.58
		5k	61	115	6	0.95	4.05
		10k	56	101	5	0.97	4.63
s13207	1400	1k	99	266	2	0.74	5.14
		2k	86	264	1	0.60	5.25
		5k	68	262	1	0.60	5.28
		10k	53	262	1	0.53	5.27
s35932	3526	32	26	254	3	0.90	13.03
		64	18	246	2	0.76	13.38
		128	5	169	1	0.93	18.67
s38417	3328	100k	46	360	1	0.94	9.12
		150k	46	283	1	0.95	11.59
		200k	39	296	0	0.94	11.21
		250k	44	217	0	0.97	15.27

As in the case of merging, the reduction factor increases with the circuit size, and for all examples the concatenation process provided a considerably higher reduction factor that simply merging the cubes. Also it can be seen from column 7 of Table III that the encoding efficiency is greater than 0.90 in most cases which means that in the average we have to store less than 1.1 bits per specified bit of the test set. Cube concatenation can be used only if the MP-LFSR does not share flip-flops with the scan.

In all cases reported in Table II and III the MP-LFSR's had 16 polynomials with s_{max} stages. We were always able to find an encoding for every test cube for the *a priori* selected polynomials. Figs. 9(a) and (b) show very clearly the effect concatenation has on the distribution of the numbers of specified bits in the test set. While these numbers are almost uniformly distributed between the minimal and maximal number of specified bits before concatenation the whole spectrum shifts to the extreme right after concatenation. This is achieved by combining test cubes having a large number of specified bits with those cubes with a small number of specified bits. The average number of specified bits is much closer to the maximum number, and therefore a higher encoding efficiency can be achieved.

As already pointed out full fault coverage is achieved using the MP-LFSR as mixed-mode vector pattern generator. The partitioning of the test into random and stored pattern test affects both, test application time and test data storage and we have to trade-off one against the other. A comparison between test application time and test data storage for the circuits s5378, s9234, and s13207 after concatenation is shown in Fig. 10.

This comparison is very useful in deciding at which point the random test should stop and the deterministic test phase should start. Considering for example the circuit s9234 it can be seen that a random test beyond 30k random patterns has no significant influence on the amount of test data that has to be stored for the deterministic test.

The MP-LFSR consists of a q-bit counter (q is usually from 2 to 4), s_{max}-stage shift register, programmable feedback

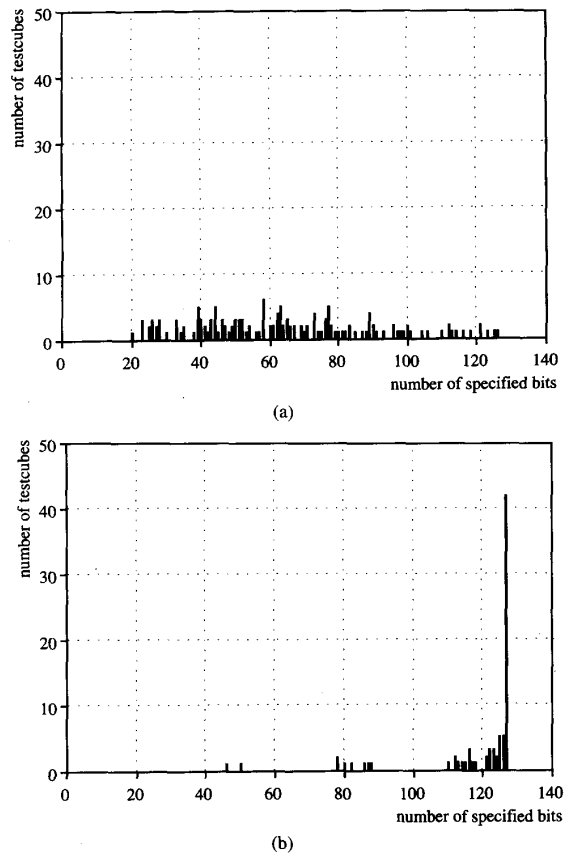


Fig. 9. Distribution of the numbers of specified bits (circuit s9234, after 1k random patterns) (a) after merging, (b) after concatenation.

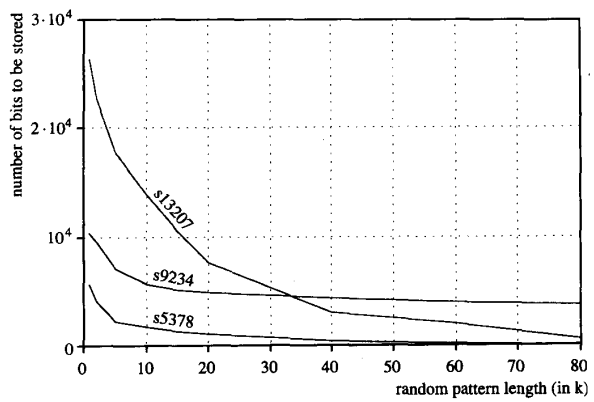


Fig. 10. Tradeoff between random pattern length and test data storage.

polynomial, and control logic. The number of flip-flops in this scheme depends on the maximum number of specified bits in a test cube that the scheme is designed to encode. The other logic is constant and independent of the size of the circuit. As an example let us consider the area overhead in circuit s38417 with scan implemented using edge triggered D-type flip-flops. If the MP-LFSR does not share the flip-flops with the scan chain, the area overhead of a 212-stage MP-LFSR

and 32-stage compactor for s38417 is approximately 5%. If the scan register flip-flops are shared with the MP-LFSR than, assuming 32-stage generator which is not shared, the overhead is 1.5%. In this number only less than 0.25% of the total area is contributed by feedback polynomial reconfiguration logic. In other words for circuit s38417 with scan, 32-stage generator and 32-stage compactor it takes additional 0.25% area to implement this new MP-LFSR BIT scheme. In all analyzed cases we were able to implement this scheme with less than 100 additional gates. Since the complexity of this hardware does not depend on the size of the circuit, for circuits larger than 100k gates the additional area overhead of this scheme is less than 0.1%.

VIII. Conclusions

High quality requirements demand that microelectronics circuits are thoroughly and completely tested. However, due to area or performance constraints it is not always possible to design large digital systems to be completely testable by random patterns. Those circuits can still be completely tested by a combination of random and deterministic patterns. Such solution offers small test data volume and complete fault coverage.

In this paper we presented a new and very efficient scheme for the compression and decompression of input test data which is based on reseeding of MP-LFSR's. The test hardware is simple as it involves less than a hundred additional gates to implement the scheme in a circuit that already has scan and BIST logic. The scheme is compatible with scan, parallel scan, partial scan and boundary scan design. This whole approach relies on a very mature domain of automatic test pattern generation to obtain the test cubes for difficult to test faults. The scheme is applicable to any fault model, including delay faults, as long as the test cubes can be generated. The mixture of random and encoded deterministic patterns guarantees a complete fault coverage. It is also very flexible since the decompression hardware is programmable with the seed information which can be modified even after the circuit design has been completed.

The compressed test data can be stored in an inexpensive system memory, moved easily through slow serial channels, like boundary scan, to the chip, and then decompressed with a very simple hardware. The benefits arising from the reduced bandwidth requirements for the compressed data can also be exploited by external testers with the "scan option." Currently the test data for scan designs is very often overspecified, i.e., test cubes are represented as test vectors with all positions specified. The scan memory is very often one of the most heavily used and limiting resources in traditional testers. The overspecification of test data also increases the time required to load the data on chip and effectively increasing the test application time.

This compression scheme creates also new challenges for automatic test pattern generation. Here the objective of test pattern generation is focused on the minimization of the total number of specified bits in test cubes for difficult to test faults, rather than the number of test vectors as it is done in generation of compact tests [22].

Acknowledgment

The authors wish to thank B. Koenemann for the opportunity he gave the authors to present an early version of this work at the IBM Test Forum, R. Mehio for his ingenious implementation of the algorithm to systems of linear equations and many experiments he performed, J. Tyszer for many discussions on linear dependencies, H.-J. Wunderlich for his comments of test cube concatenation, and the anonymous reviewers for their valuable comments. SOCRATES was provided by the Technical University of Munich.

References

[1] Z. Barzilai, D. Coppersmith, and A. L. Rosenberg, "Exhaustive generation of bit patterns with applications to VLSI self-testing," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 190–194, Feb. 1983.

[2] L. T. Wang and E. J. McCluskey, "Circuits for pseudo-exhaustive test pattern generation," in *Proc. IEEE Int. Test Conf.*, 1986, pp. 25–37.

[3] H.-J. Wunderlich and S. Hellebrand, "The pseudo-exhaustive test of sequential circuits," *IEEE Trans. Computer-Aided Design Integrated Circuits Syst.*, vol. 11, no. 1, pp. 26–33, Jan. 1992.

[4] P. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI*. New York: Wiley-Interscience, 1987.

[5] F. Brglez *et al.*, "Hardware-based weighted random pattern generation for boundary-scan," in *Proc. IEEE Int. Test Conf.*, Washington, DC, 1989, pp. 264–274.

[6] H.-J. Wunderlich, "Self test using unequiprobable random patterns," in *Proc. 17th Int. Symp. Fault-Tolerant Comput.*, Pittsburgh, PA, 1987, pp. 258–263.

[7] ——, "Multiple distributions for biased random test patterns," in *Proc. IEEE Int. Test Conf.*, Washington, DC, 1988, pp. 236–244.

[8] R. Dandapani, J. H. Patel, and J. A. Abraham, "Design of test pattern generators for built-in test," in *Proc. IEEE Int. Test Conf.*, Washington, DC, 1984, pp. 315–319.

[9] W. Daehn and J. Mucha, "Hardware test pattern generators for built-in test," in *Proc. IEEE Int. Test Conf.*, 1981, pp. 110–113.

[10] S. B. Akers and W. Jansz, "Test set embedding in built-in self-test environment," in *Proc. IEEE Int. Test Conf.*, Washington, DC, 1989, pp. 257–263.

[11] C. Dufaza and G. Gambon, "LFSR-based deterministic and pseudo-random test patterns generator structures," in *Proc. Europ. Test Conf.*, Munich, Germany, 1991, pp. 27–34.

[12] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proc. Europ. Test Conf.*, Munich, Germany, 1991, pp. 237–242.

[13] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, 1992, pp. 120–129.

[14] S. Venkataraman, J. Rajski, S. Hellebrand, and S. Tarnick, "An efficient BIST scheme based on reseeding of multiple polynomial linear feedback shift registers," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Santa Clara, CA, 1993, pp. 572–577.

[15] C. L. Chen, "Linear dependencies in linear feedback shift registers," *IEEE Trans. Comput.*, vol. C-35, no. 12, pp. 1086–1088, Dec. 1986.

[16] P. Bardell, "Design considerations for parallel pseudorandom pattern generators," *J. Electron. Testing: Theory and Application*, vol. 1, no. 1, pp. 73–87, Feb. 1990.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[18] S. Venkataraman, "Built-in self test based on reseeding of linear feedback shift registers," Masters Thesis, Dept. of Electrical Engineering, McGill University, Montreal, July 1993.

[19] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1989, pp. 1929–1934.

[20] M. Schulz and E. Auth, "Advanced automatic test generation and redundancy identification techniques," in *Proc. 18th Int. Symp. Fault-Tolerant Comput.*, Tokyo, Japan, 1988, pp. 30–35.

[21] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed. New York: McGraw-Hill, 1978.

[22] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," in *Proc. IEEE Int. Test Conf.*, 1991, pp. 194–203.

Sybille Hellebrand (A'89) received the diploma degree in mathematics from the University of Regensburg, Germany, in 1986.

In 1986 she joined the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, where she received the Ph.D. degree in 1991. Then she was a postdoctoral fellow at the TIMA/IMAG-Computer Architecture Group, Grenoble, France. Since May 1992 she has been an Assitant Professor at the University of Siegen, Germany. Her main research interests include BIST for high quality applications and synthesis of testable systems.

Janusz Rajski (A'87) received the M.Eng. degree in electrical engineering from the Technical University of Gdansk, Poland, in 1973, and the Ph.D. degree in electrical engineering from the Technical University of Poznan, Poland, in 1982.

From 1973 to 1984, he was a member of the faculty of the Technical University of Poznan. In June 1984, he joined McGill University, Montreal, P.Q., Canada, where from 1989 he was an Associate Professor. In January 1995 he accepted the position of a Chief Scientist at Mentor Graphics Corporation, Wilsonville, OR. His main research interests include design automation and testing of VLSI systems, design for testability (DFT), built-in self test (BIST), and logic synthesis. He has published more than 60 research papers in these areas. He was co-recipient of the 1993 Best Paper Award for the paper on synthesis of testable circuits published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has done a contract work and has been a consultant in the area of testing to a number of companies.

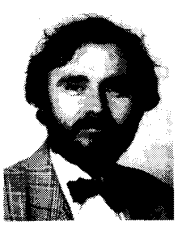
Dr. Rajski was a Guest Co-editor of the June 1990 and January 1992 Special Issues of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS devoted to the 1977 and 1979 International Test Conference, respectively. He is a member of the editorial board of the *Journal of Electronic Testing* (JETTA) and an associate editor for the IEEE DESIGN AND TEST OF COMPUTERS Magazine. He has served on technical program committees of various conferences, coordinated the topic of Automatic Test Pattern Generation and Delay Testing for the International Test Conference. He is a co-founder and the Chair of the Program Committee of the International Test Synthesis Workshop.

Steffen Tarnick (A'92) received the diploma degree in mathematics from the Technical University of Dresden, Germany, in 1989.

From 1989 to 1991 he was a research assistant at the Central Institute for Cybernetics and Information Processes, Berlin, Germany. In 1990, he spent two months at the Institute for Computer Design and Fault-Tolerance at the University of Karlsruhe, Germany. Beginning in April 1991, he spent a year at the TIM3 laboratory, Grenoble, France, where he worked in the field of VLSI circuits testing. In 1992, he spent one month at the MACS laboratory at the McGill University, Montreal, P.Q., Canada. His research interests include built-in self-test, design for testability and self-checking circuits design. He is currently with the Max Planck Society Group for fault-tolerant computing at the University of Potsdam, Germany, where he is working toward his Ph.D. degree.

Srikanth Venkataraman (S'87) received the B.E. (Hons.) degree in electrical and electronics engineering and the M.Sc. (Tech.) degree in computer science from the Birla Institute of Technology and Science, Pilani, India, in 1991, the M.Eng. degree in electrical engineering from McGill University, Montreal, in 1993.

He is currently working toward the Ph.D. degree in electrical and computer engineering at the University of Illinois, Urbana–Champaign. He is a research assistant at the Center for Reliable and High-Performance Computing, University of Illinois, Urbana–Champaign. His research interests include testing and diagnosis of VLSI systems.



Bernard Courtois received the Engineer degree in 1973 from the Ecole Nationale Supérieure d'Informatique et Mathématiques Appliquées de Grenoble, Grenoble, France, and next the "Docteur-Ingénieur" and "Docteur ès-Sciences" degrees from the Institut National Polytechnique de Grenoble.

He is currently Director of the Laboratory of Techniques of Informatics and Microelectronics for Computer Architecture (TIMA) where researches include CAD, architecture and testing of integrated circuits and systems. He is also the Director of CMP Service that is servicing French and Foreign Universities and Companies for chip prototyping and small volume, and he is a member of the Executive Board of EUROCHIP and of the Core Group of CHIPSHOP. He was the General Chair of EDAC-EUROASIC 1993 and Co-Program Chair of EDAC-ETC-EUROASIC 1994 Conference and Exhibition.