Test Embedding With Discrete Logarithms

Mody Lempel, Sandeep K. Gupta
and Melvin A. Breuer

CENG Technical Report 94-02

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4469

September 1994

Test Embedding with Discrete Logarithms *

Mody Lempel
Computer Science Department
University of Southern California
Los Angeles, CA 90089-2560

Sandeep K. Gupta and Melvin A. Breuer
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2560

September 15, 1994

Abstract

When using Built-In Self Test (BIST) for testing VLSI circuits, a major concern is the generation of proper test patterns that detect the faults of interest. Usually a linear feedback shift register (LFSR) is used to generate test patterns.

We first analyze the probability that an arbitrary pseudo-random test sequence of *short* length detects all faults. The term *short* is relative to the probability of detecting the fault with the fewest test patterns.

We then show how to guide the search for an initial state (seed) for a LFSR with a given primitive feedback polynomial so that all the faults of interest are detected by a minimum length test sequence. Our algorithm is based on finding the location of test patterns in the sequence generated by this LFSR. This is accomplished using the theory of *discrete logarithms*. We then select the shortest subsequence that includes test patterns for all the faults of interest, hence resulting in 100% fault coverage.

*This work was supported by the Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. JFBI90092. The views and conclusions considered in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

1 Introduction

Built-In Self-Test is the capability of a circuit to test itself. The idea behind BIST is to create *pattern generators (PGs)* to generate test patterns for the circuit and *response analyzers (RAs)* to compact the circuit response to the test patterns that are applied. The PGs and RAs are usually implemented from existing registers. Some registers are used as both a PG and a RA.

In this paper we deal with the design of efficient and effective pattern generators based on linear feedback shift registers (LFSR). Their effectiveness is measured in terms of generating test patterns for all the faults of interest, and efficiency in terms of minimum test length (time). Both ends will be accomplished by proper selection of the feedback polynomial (configuration) and the initial seed (state) of the LFSR. In a complementary work [12] we show how to select feedback polynomials to achieve effective (*zero-aliasing*) compaction. When a register is to function both as a PG and a RA, by selecting the feedback polynomial to be the one that achieves the best PG results from a set of zero-aliasing polynomials, both ends are achieved simultaneously.

An n-stage LFSR with a primitive feedback polynomial generates a permutation of all the non-zero binary n-tuples. Changing the polynomial changes the permutation. The seed specifies the starting position for scanning the permutation. To achieve 100% detection of the faults of interest the LFSR must generate patterns until the set of patterns contains at least one test pattern for every fault. The problem is that the permutation might not contain a relatively short subsequence of test patterns for all the faults, and even if it does, it is usually not known where this short subsequence begins.

There are two approaches to deal with the problem of generating all the required test patterns. The first is by trying to estimate the number of test patterns required to achieve 100% fault coverage (fc). The second is by adding hardware to guarantee 100% detection with a small test set. While the first approach keeps the hardware at a minimum, it significantly adds to the required test time. The second approach results in a very short test time, but requires significant hardware overhead. We will first expand on the above approaches and then discuss our ideas for the design of LFSR-based PGs.

To estimate the number of patterns required to achieve 100% fc, a number of authors [13] [17] [21] asked how many test vectors need be applied to achieve a given fc with a specified degree of confidence. The answers depend on the *detectability profile* of the circuit, i.e. the number of

test patterns for each fault. For example, to achieve 100% fc when the probability of detection of the hardest fault is p, Savir and Bardell [17] suggest using a test sequence of length $11/p$. The main motivation behind this question is that it eliminates the need for test generation and for fault simulation. With the advances in test generation and fault simulation tools (not to mention platform speed), we believe this is not as important as it used to be. On the other hand, shorter test sequences for combinational subcircuits allow for shorter test sessions for the whole circuit and allow for easier synthesis of *zero-aliasing* response analyzers [12].

As opposed to trying to detect all the faults of interest with one pseudo-random sequence, a second approach is to either use special hardware to generate small test sets, or to use a short pseudo-random sequence to detect most of the faults, and then use special hardware to detect the remaining faults. The use of non-linear feedback functions are suggested in [6] and [7]. Using this approach, one first creates a test set T that achieves 100% fault coverage and then synthesizes logic to generate the test patterns. In [6] the patterns in T are viewed as states of a finite state machine. Logic is built around the register such that the states of the register generate these patterns. In [7] patterns are added to T such that T can be ordered in a way that consecutive patterns differ in only one bit position. A ROM is used to store these positions. A related approach is that of weighted random patterns [22], [3], [14]. Logic is added to the register to affect the probability that each register cell outputs either a "1" or a "0". The probabilities are based on the test patterns which detect the faults of interest. A store-and-generate approach is suggested in [2]. Precomputed patterns are stored in a ROM. Each pattern is loaded into a LFSR and the LFSR proceeds to generate test patterns. After a certain number of patterns have been generated, the next pattern is loaded into the LFSR. This can be seen as a multiple seeding of the LFSR, which at the extreme can be used to detect all but a certain number of faults with the first seed, and the remaining seeds are patterns for the undetected faults. One implementation of this idea is suggested in [1]. A ROM, counter and additional linear logic is used to generate the patterns of T. The authors note that it is very unlikely that T will be a set for which their scheme will work. Another implementation is suggested in [11]. In this scheme not only is the PG reseeded, but with each seed there is a corresponding feedback polynomial. Decoding logic is used to reconfigure the feedback connections of the register, depending on the encoding of the polynomial. Along similar lines, [8] suggests to divide T into subsets of linearly independent patterns. The linear connections of the LFSR can be modified such that each connection generates a different subset of T. In [20] it is argued that one can do with

just one such subset of T which includes the patterns required to detect the random resistant faults. Remaining faults will be detected by the succeeding patterns generated by the LFSR. A different scheme is suggested in [9], where the patterns are stored in a ROM and the outputs of the CUT act as addresses to the ROM.

All the above PG schemes cause additional area and delay overhead compared to a LFSR-based PG, although they reduce the size of the test set, in some cases considerably. Another drawback of some of these schemes is that changing the feedback function of the LFSR makes the compaction properties of the resulting circuits extremely difficult to analyze.

In this paper we try to find *short one-seed* pseudo-random test sequences that achieve 100% fc. The term *short* is relative to the probability of detecting the hardest fault of a circuit. If this probability is p, then *short* will mean a test length of $L = \frac{1}{p}$. By using just one seed the BIST control circuitry is kept at a minimum. We first show that a pseudo-random test sequence of length at most $2L$ has a high probability of achieving 100% fc. Thus, a random process of selecting a random primitive polynomial and a random seed for a LFSR-based PG is likely to produce the desired test length. By simulating up to $2L$ test patterns, one knows whether the desired sequence is found or not. If not, another random selection is made. This scheme will be best suited for *randomly testable circuits*, i.e. those circuits with high values of p. For *random resistant circuits* we suggest a more sophisticated way of selecting the feedback polynomials and seeds that will produce shorter test sequences and require less computation time. We use the theory of *discrete logarithms* to *embed* a subset of test patterns in a LFSR sequence, from which we produce the test sequence for all faults. *The applicability of these schemes is dependent on the computational effort one is willing to expend and on the time a test sequence is allowed to run.*

The tests we embed can either be one pattern tests for non-sequential faults, or, using schemes such as in [19], two-pattern tests for sequential faults.

As the pattern sequence of LFSRs and one-dimensional *Cellular Automata (CA)* with the same primitive characteristic function are isomorphic, [18], our algorithms will also work for CAs.

The rest of this paper is organized as follows. Throughout the paper, n denotes the number of inputs to the circuit under test (CUT). In Section 2 we analyze the probability of detecting a fault having 2^{k+i} test patterns, given a test sequence of length $L = 2^{n-k}$ and we give lower

and upper bounds on the probability of detecting all the faults of interest. In Section 3 we assume two different detectability profile models and derive the probability bound values for 100% fc for circuits that abide by these models. In Section 4 we find the actual detectability profile for some example circuits, derive the probability of drawing short test sequences and conduct random experiments which validate our analytic results. These sections provide the basis for our claim that short test sequences can be found in acceptable time constraints. We then proceed to introduce our procedures for selecting primitive feedback polynomials and seeds for the LFSRs. In Section 5 we introduce the notion of *discrete logarithms* and show its relation to the sequencing of patterns by a LFSR. In Section 6 we state the *test embedding problem* and propose our solution. Section 7 defines and characterizes faults we classify as *hard faults*, which are of major importance to our algorithm. In Section 8 we present experimental results. We conclude with Section 9.

2 The probability of 100% fault coverage with a random test sequence of length L

Consider a combinational circuit with n inputs. In this section we address the following two questions.

Given a fault with $K = 2^k$ test patterns, out of $N = 2^n$ possible input patterns to the circuit, what is the probability that a random test sequence of length $L = 2^{n-k}$ does not detect the fault, assuming the patterns are drawn with no replacement? What is the probability that a random sequence of length L does not detect a fault with 2^{k+i} test patterns, where $i \geq -1$?

To answer these questions we define the function $p_{nd}(t, L)$ which is the probability that a sequence of length L does not detect a fault with t test patterns. Hence, we are looking for the values of $p_{nd}(t, L)$ when t equals 2^{k+i} for $i \geq -1$. The total number of sequences (with no importance to order) of length L is $\begin{pmatrix} N \\ L \end{pmatrix}$. Of those, the number of sequences that do not

detect a given fault with t test patterns is $\binom{N-t}{L}$. Hence,

$$p_{nd}(t,L) = \frac{\binom{N-t}{L}}{\binom{N}{L}}.$$

Writing this expression in factorial form

$$p_{nd}(t,L) = \frac{(N-t)!(N-L)!}{(N-t-L)!N!}.$$

After cancelling the appropriate terms

$$\begin{aligned} p_{nd}(t,L) &= \frac{(N-L)(N-L-1)\cdots(N-L-t+1)}{N(N-1)\cdots(N-t+1)} \\ &= \left(\frac{N-L}{N}\right)\left(\frac{N-L-1}{N-1}\right)\cdots\left(\frac{N-L-t+1}{N-t+1}\right) \\ &= \left(1-\frac{L}{N}\right)\left(1-\frac{L}{N-1}\right)\left(1-\frac{L}{N-t+1}\right) \qquad (1) \\ &< \left(1-\frac{L}{N}\right)^t. \qquad (2) \end{aligned}$$

For $t = 2^k = K$,

$$\begin{aligned} p_{nd}(2^k,L) &< \left(1-\frac{L}{N}\right)^K \\ &= \left(1-\frac{1}{K}\right)^K \\ &< \frac{1}{e}. \end{aligned}$$

For $t = 2^{k-1} = K/2$,

$$\begin{aligned} p_{nd}(2^{k-1},L) &< \left[\left(1-\frac{1}{K}\right)^K\right]^{\frac{1}{2}} \\ &< \left(\frac{1}{e}\right)^{\frac{1}{2}}. \end{aligned}$$

For $t = 2^{k+i} = K \cdot 2^i$, $i \geq 1$

$$\begin{aligned} p_{nd}(2^{k+i}, L) &< \left[\left(1 - \frac{1}{K}\right)^K\right]^{2^i} \\ &< \left(\frac{1}{e}\right)^{2^i}. \end{aligned}$$

In general, for $t = wK$

$$\begin{aligned} p_{nd}(t, L) &< \left[\left(1 - \frac{1}{K}\right)^K\right]^w \\ &< \left(\frac{1}{e}\right)^w. \end{aligned} \tag{3}$$

In most cases of interest to us, t will be (much) less than L. For these cases, this upper bound is tight. By Equation (1)

$$p_{nd}(t, L) > \left(1 - \frac{L}{N - t + 1}\right)^t$$

hence

$$p_{nd}(t, L) > \left(1 - \frac{1}{K - \frac{t-1}{L}}\right)^t$$

and by the assumption of $t < L$

$$p_{nd}(t, L) > \left(1 - \frac{1}{K - 1}\right)^t.$$

For $t = wK$, this becomes

$$p_{nd}(t, L) > \left[\left(1 - \frac{1}{K - 1}\right)^K\right]^w. \tag{4}$$

When $K = 32$

$$p_{nd}(t, L) > \left(0.9519\frac{1}{e}\right)^w$$

and for $K = 64$

$$p_{nd}(t, L) > \left(0.9762\frac{1}{e}\right)^{w}.$$

Combining Equations (3) and (4) under the assumption $t = wK < L$ results in

$$\left[\left(1 - \frac{1}{K-1}\right)^{K}\right]^{w} < p_{nd}(t, L) < \left[\left(1 - \frac{1}{K}\right)^{K}\right]^{w}$$

and as K increases, the bounds become tighter.

We defined the sequence length to be $L = \frac{N}{K}$. If the sequence length is doubled, the probability of missing a fault with t test patterns is squared. By Equation (2)

$$\begin{aligned} p_{nd}(t, 2L) &< \left(1 - \frac{2L}{N}\right)^{t} \\ &= \left[\left(1 - \frac{1}{K/2}\right)^{K/2}\right]^{2w} \\ &< \left(\frac{1}{e}\right)^{2w}. \end{aligned}$$

Similarly, if the test sequence length is halved $(L/2)$ the probability is the square root of its value for a sequence of length L.

Using the values of $p_{nd}(2^{k+i}, L)$, we would like to bound from above and below the probability that a sequence of length L detects all the faults of interest. We do this by taking for each value of t the closest power of 2 greater or equal to t and the closest power of 2 less than or equal to t. By considering the power of 2 greater than t we derive the upper bounds, and by considering the power of 2 less than t we derive the lower bounds.

Let $F = \{f_1, f_2, \ldots, f_s\}$ be the set of all the faults of interest in the CUT. Let t_i be the number of tests for fault f_i and let $k_i = \lceil \log t_i \rceil$. Let $k_{min} = \min_i\{k_i\}$ and $k_{max} = \max_i\{k_i\}$. Group the faults into subgroups $C_{k_{min}}, C_{k_{min}+1}, \ldots, C_{k_{max}}$, where $f_i \in C_j$ iff $k_i = j$.

Let $k = k_{min}$. We are interested in finding the probability that a random sequence of length L detects all the faults of F.

Let p_j be the probability that a fault in C_j is not detected by the sequence. Since a fault in C_j has between 2^{j-1} and 2^j test patterns, the probability p_j is lower bound by $p_j \geq p_{nd}(2^j, L)$.

We know that $\frac{1}{e^{2^j-k}} > p_{nd}(2^j, L)$. The value of p_j is either greater than $\frac{1}{e^{2^j-k}}$ or less than or equal to $\frac{1}{e^{2^j-k}}$. Our goal is to get a *pessimistic* upper bound on the probability of detection, hence we will assume that

$$p_j > \frac{1}{e^{2^j-k}}.$$

The probability, q_j, that a fault in C_j is detected by the sequence is upper bound, by our above assumption, by

$$q_j < 1 - \frac{1}{e^{2^j-k}}.$$

Notice that without our assumption, the upper bound on q_j can be greater. Hence, the probability, q, that a random sequence of length L detects *all* the faults of F is upper bound by

$$q < \prod_{j=k}^{k_{max}} \left(1 - \frac{1}{e^{2^j-k}}\right)^{|C_j|}.$$

In the above expression we assume that the detectabilities of any two faults are independent.

Similarly, we can define $d_i = \lfloor \log t_i \rfloor$ and the subgroups $\{F_j\}$ where $f_i \in F_j$ iff $d_i = j$. Let $d = \min_i\{d_i\}$ and $d_{max} = \max_i\{d_i\}$, then d is either k or $k-1$ and d_{max} is either k_{max} or $k_{max}-1$. The probability r_j that a fault in F_j is not detected is upper bound by

$$r_j < \frac{1}{e^{2^j-k}}$$

hence the probability, s_j, that a fault in F_j is detected is lower bound by

$$s_j > 1 - \frac{1}{e^{2^j-k}}.$$

Thus, the probability q of detecting all the faults of interest is bound by

$$\prod_{j=d}^{d_{max}} \left(1 - \frac{1}{e^{2^j-k}}\right)^{|F_j|} < q < \prod_{j=k}^{k_{max}} \left(1 - \frac{1}{e^{2^j-k}}\right)^{|C_j|}. \tag{5}$$

The actual value of q will be closer to the upper (lower) bound when for most of the harder faults the number of test patterns is closer to the power of 2 from above (below).

The super-exponential decrease in the probability of not detecting a fault, as we move from C_j to C_{j+1}, allows us to consider only the first few subgroups of faults when estimating the

probability q. This is an analytic explanation to a similar observation made in [17]. To see this, consider the following question. How many faults x_j, with 2^{j+1} test patterns are needed such that the product of their detection probability equals the detection probability of a single fault with 2^j test patterns? This can be written as

$$\begin{aligned}\left(1-\frac{1}{e^{2(j+1)-k}}\right)^{x_j} &= \left(1-\frac{1}{e^{2j-k}}\right) \quad \Rightarrow \\ x_j &= \frac{\ln\left(1-\frac{1}{e^{2j-k}}\right)}{\ln\left(1-\frac{1}{e^{2j+1-k}}\right)}.\end{aligned}$$

Substituting y for e^{2j-k} we get

$$x_j = \frac{\ln\left(1-\frac{1}{y}\right)}{\ln\left[\left(1-\frac{1}{y}\right)\left(1+\frac{1}{y}\right)\right]} = \frac{\ln\left(1-\frac{1}{y}\right)}{\ln\left(1-\frac{1}{y}\right)+\ln\left(1+\frac{1}{y}\right)}. \tag{6}$$

The power series expansion of $\ln(1+z)$, where $-1 < z < 1$ is

$$z - \frac{1}{2}z^2 + \frac{1}{3}z^3 - \frac{1}{4}z^4 - \ldots.$$

Substituting z for $\frac{1}{y}$ in Equation (6), and expanding to the first $2M$ terms, we get

$$\begin{aligned}x_j &= \frac{\sum_{i=1}^{2M}\frac{z^i}{i}}{\sum_{i=1}^{2M}\frac{z^i}{i}+\sum_{i=1}^{2M}(-1)^i\frac{z^i}{i}} \\ &= \frac{\sum_{i=1}^{2M}\frac{z^i}{i}}{\sum_{i=1}^{M}\frac{z^{2i}}{i}} \\ &= \frac{1}{z}\times\frac{\left[z+\frac{z^2}{2}+\frac{z^3}{3}+\frac{z^4}{4}+\ldots\right]}{\left[z+\frac{z^3}{2}+\frac{z^5}{3}+\frac{z^7}{4}+\ldots\right]} \\ &> \frac{1}{z} = e^{2j-k}.\end{aligned}$$

For $j-k = -1, 0, 1, 2, 3$ and 4, the values of x_j are greater than $2(> e^{2^{-1}})$, $3(> e)$, $7(> e^2)$, $55(> e^{2^2})$, $2981(> e^{2^3})$, and $8886114(> e^{2^4})$ respectively. This means that when computing the

bounds on q, the significance of one fault with 2^{k+4} test patterns to the probability of success is greater than the contribution of e^{16} faults with 2^{k+5} test patterns. Equivalently, the probability of detecting a single fault with 2^{k+4} test patterns is less than detecting e^{16} faults with 2^{k+5} test patterns. Thus, the faults in the subgroups $C_{d+5}, \ldots C_{d_{max}}$ and $F_{k+5}, \ldots F_{k_{max}}$ do not have to be taken into account when analyzing the probability of success.

3 Detectability profile models

The actual value of the probability q depends on the distribution of faults in the different subgroups. To get an idea of this value as a function of the number of faults in C_k and F_d, we assume two detectability profile models, both placing an emphasis on the first few subgroups. These models will let us approximate bounds on the probability q without considering the detectability profile, but only the total number of faults and the number of faults in F_d and C_k. We later use the profiles of actual circuits and the results demonstrate that our two models are very pessimistic, i.e. we should expect better results from actual circuit distributions than from the model approximations.

The first model we consider is the exponential model. In this model we assume $d = k - 1$ and the following distribution:

$$\begin{array}{rclcrcl}
|F_d| & = & u & , & |C_k| & = & v \\
|F_{d+1}| & = & ue^{\frac{1}{2}} & , & |C_{k+1}| & = & ve \\
|F_{d+2}| & = & |F_{d+1}|e & , & |C_{k+2}| & = & |C_{k+1}|e^2 \\
|F_{d+3}| & = & |F_{d+2}|e^2 & , & |C_{k+3}| & = & |C_{k+2}|e^4 \\
|F_{d+4}| & = & |F_{d+3}|e^4 & , & |C_{k+4}| & = & |C_{k+3}|e^8 \\
|F_{d+5}| & = & |F_{d+4}|e^8 & , & |C_{k+5}| & = & |C_{k+4}|e^{16}.
\end{array}$$

The idea behind this model is that the probability of detecting all the faults in F_{d+i} (C_{k+i}) is the same as detecting all the faults in F_d (C_k). Using this model, we get

$$\left(1 - \left(\frac{1}{e}\right)^{2^{-1}}\right)^{|F_d|} \approx \left(1 - \left(\frac{1}{e}\right)^{2^0}\right)^{|F_{d+1}|}$$

$$\left(1 - \left(\frac{1}{e}\right)^{2^0}\right)^{|F_{d+1}|} \approx \left(1 - \left(\frac{1}{e}\right)^{2^1}\right)^{|F_{d+2}|}$$

$$\vdots$$

$$\left(1-\left(\frac{1}{e}\right)^{2^8}\right)^{|F_{d+4}|} \approx \left(1-\left(\frac{1}{e}\right)^{2^{16}}\right)^{|F_{d+5}|}.$$

The same is true for the subgroups $\{C_i\}$. Thus, we can approximate q with the bounds

$$\left(\left(1-\left(\frac{1}{e}\right)^{\frac{1}{2}}\right)^{|F_d|}\right)^{d_{max}-d} < q < \left(\left(1-\frac{1}{e}\right)^{|C_k|}\right)^{k_{max}-k}.$$

The values of $k_{max}-k$ and $d_{max}-d$ can be bound as follows. The number of faults in each of the subgroups C_{k+i} $(k+i<k_{max})$ is ve^{2^i-1}. The last subgroup, $C_{k_{max}}$, may not be completely full, hence

$$|F| < ve^{2^{k_{max}-k}}$$

thus,

$$\ln|F| < \ln v + 2^{k_{max}-k} \quad \Rightarrow$$

$$k_{max}-k > \log(\ln|F| - \ln v).$$

Similarly, the subgroup $F_{d_{max}-1}$ is full, hence

$$|F| > ue^{\frac{2^{d_{max}-d-1}-1}{2}}$$

and

$$d_{max}-d < \log(2(\ln|F| - \ln u)+1)+1$$

hence

$$\left(1-\left(\frac{1}{e}\right)^{\frac{1}{2}}\right)^{u\cdot log(2(\ln|F|-\ln u)+1)+1} < q < \left(1-\frac{1}{e}\right)^{v\cdot log(\ln|F|-\ln v)}.$$

Setting, for example, u and v to equal 10, and $|F|=5000$, we get

$$9\cdot 10^{-20} < q < 6.61\cdot 10^{-6}.$$

By doubling the test length the probability bounds become

$$4.34\cdot 10^{-6} < q < 2.28\cdot 10^{-2}.$$

The second distribution is the linear distribution, in which

$$\begin{array}{rclcrcl}
|F_d| & = & u & , & |C_k| & = & v \\
|F_{d+1}| & = & e^{\frac{1}{2}}u & , & |C_{k+1}| & = & ev \\
|F_{d+2}| & = & e^{\frac{1}{2}}|F_{d+1}| & , & |C_{k+2}| & = & e|C_{k+1}| \\
|F_{d+3}| & = & e|F_{d+2}| & , & |C_{k+3}| & = & e|C_{k+2}| \\
|F_{d+4}| & = & e|F_{d+3}| & , & |C_{k+4}| & = & e|C_{k+3}| \\
|F_{d+5}| & = & e|F_{d+4}| & , & |C_{k+5}| & = & e|C_{k+4}|.
\end{array}$$

Using this distribution

$$|F| = v + v \sum_{i=0}^{d_{max}-d-1} e^{i+\frac{1}{2}}.$$

With this distribution, the product of the probability of detecting all the faults in F_{d+2+i} ($0 \leq i \leq 3$) is (much) greater than the probability of detecting one fault in $F_{d+2+i-1}$. The probability of detecting all the faults in F_{d+1} is greater than the probability of detecting all the faults in F_d. The same applies to the faults in the subgroups $\{C_j\}$. Thus, we can approximate the bound on the probability of detecting all the faults by

$$\left(1 - \left(\frac{1}{e}\right)^{\frac{1}{2}}\right)^{2u} < q < \left(1 - \frac{1}{e}\right)^{2v}. \tag{7}$$

Assuming $u = v = 10$, $|F|$ is irrelevant in this case,

$$7.91 \cdot 10^{-9} < q < 10^{-4}.$$

If we double the length of the test sequence, the bounds become

$$10^{-4} < q < 5.45 \cdot 10^{-2}.$$

The upper bounds, for both models, on the probability that a random test sequence of length $2L$ achieves 100% fc is better than $\frac{1}{50}$. These results, of course, are dependent on our choice of values for u, v and F. Irrespective of these values is the effect that doubling the test length has on the probability of 100% fc. This effect is a result of the exponential decrease in the probability of not detecting a fault. While these two detectability profiles seem artificial, experiment circuits show them to be very pessimistic, i.e. *actual profiles give rise to detection probabilities that are better than the models' bounds.*

4 Experimental results on finding short pseudo-random test sequences

We synthesized 13 circuits from the Berkeley [4] benchmarks as multilevel circuits. For each circuit we found its k value. Using a modified version of the ATALANTA [10] test generation system, we generated a list of all the non-equivalent faults of the circuit and proceeded to generate all possible test patterns for each fault. If the pattern count exceeded 2^{k+5}, we discarded the fault and stopped the test generation procedure for the fault. Otherwise, we recorded the number of test patterns for the fault. Having iterated through all the faults, we found the respective sizes of the subgroups $\{F_j\}$ and $\{C_j\}$. We then used Equation (5) to compute upper and lower bounds on the probability of finding a test sequence of length $L = 2^{n-k}$ which detects all the faults. These results are presented in Table 1. The first row for each circuit is the number of faults in the subgroups F_{k-1} through F_{k+5} and the second row is the number of faults in the subgroups C_{k-1} through C_{k+5}. The column labeled q in each row is the bound derived from the row using Equation (5). In the first row is the lower bound on q and in the second the upper bound. The column labeled *lin. bnd.* represents the bounds derived using the linear detectability profile model (Equation (7)). The values of u and v, the number of faults in F_{k-1} and C_k, respectively, are taken from their entries in the table. When $u = 0$, as is the case for the first eight circuits, we cannot calculate the lower bound, hence the entry is left blank. Notice that only for circuit *in5*, where the number of faults in C_k was only 1, did the model give a bound that was higher than the one given by Equation (5). We also computed the bounds on the probability that a sequence of length $2L$ will detect all the faults. The results are in Table 2. Having computed the probability bounds, we conduct 100 experiments (for circuit *chkn* only 50 experiments were conducted) of random selections of polynomials and seeds to produce 100 test sequences. We ran each sequence for at most $2L$ patterns (for circuit *chkn* at most $1.2L$), stopping whenever 100% detection was achieved. We recorded the number of random sequences of length at most L and of length between L and $2L$ that detected all the faults. The results are in Table 3. The first column shows the expected number of sequences of length at most L that detected all faults. These numbers are based on the probability values from Table 1. The second column shows the actual number of such sequences. Column three shows the number of sequences of length greater than L and at most $2L$ that detected all faults. Column four gives the total number of sequences of length at most $2L$ that detected all faults and column

Table 1: Fault distribution and probability bounds for a test sequence of length L

circuit	$k-1$	k	$k+1$	$k+2$	$k+3$	$k+4$	$k+5$	q	*lin. bnd.*
bc0	0	10	18	9	21	30	274	0.00063	
	0	10	14	10	17	30	281	0.0011	0.0001
b3	0	3	6	20	18	33	138	0.0725	
	0	3	3	12	16	20	164	0.1301	0.0638
chkn	0	5	18	37	30	31	127	0.0037	
	0	4	5	30	34	23	152	0.0438	0.0255
cps	0	25	33	39	172	131	269	$4 * 10^{-8}$	
	0	25	33	35	163	101	312	$4.2 * 10^{-8}$	$1.1 * 10^{-10}$
exep	0	12	27	42	54	80	75	$3.6 * 10^{-5}$	
	0	12	22	37	42	72	105	$8.3 * 10^{-5}$	$1.6 * 10^{-5}$
in3	0	3	7	9	2	2	193	0.0772	
	0	3	3	5	8	2	195	0.1485	0.0638
in4	0	3	6	22	25	32	158	0.1078	
	0	3	3	12	22	25	181	0.1298	0.0638
in5	0	1	14	46	55	49	56	0.0346	
	0	1	12	32	42	39	95	0.0602	0.4
in7	0	5	0	0	7	9	77	0.1007	
	0	5	0	0	7	2	84	0.1007	0.01
vg2	5	9	1	9	9	0	33	0.0001	$8.9 * 10^{-5}$
	0	12	3	0	9	9	33	0.0026	$1.6 * 10^{-5}$
vtx1	8	12	0	9	15	17	12	$2 * 10^{-6}$	$3.3 * 10^{-7}$
	0	16	4	0	9	15	29	$3.6 * 10^{-4}$	$4.2 * 10^{-7}$
x1dn	8	12	0	9	15	17	12	$2 * 10^{-6}$	$3.3 * 10^{-7}$
	0	16	4	0	9	15	29	$3.6 * 10^{-4}$	$4.2 * 10^{-7}$
x9dn	8	6	18	20	5	9	17	$1.8 * 10^{-6}$	$3.3 * 10^{-7}$
	0	14	2	27	9	5	26	$7.3 * 10^{-4}$	$2.6 * 10^{-6}$

Table 2: Probability bounds for a test sequence of length $2L$

circuit	lower bound	upper bound
bc0	0.167	0.18
b3	0.5747	0.6091
chkn	0.3422	0.5045
cps	0.0141	0.0142
exep	0.1045	0.1149
in3	0.3207	0.6106
in4	0.5743	0.6091
in5	0.6573	0.6852
in7	0.4833	0.4833
vg2	0.0267	0.1650
vtx1	0.0044	0.0907
x1dn	0.0044	0.0907
x9dn	0.00759	0.1247

five gives the expected number of such sequences, based on the probability values from Table 2. For 19 of 24 cases (omitting circuit *chkn*), we have both expected and actual results. For 9 of the 19 cases, the actual results were in the expected range. Omitting the 5 cases in which expected and actual results were 0, of the remaining 4 cases, in 3 the actual result was closer to the higher expected value and in 1 case it was closer to the lower expected value. Of the 10 cases in which the actual value was not in the expected range, in 8 cases the actual results were higher than the high end of the expected range.

Our first conclusion, based on the empirical results, is that the probability bounds given by Equation (5) are fairly accurate. The fact that the actual results were usually in the high end of the expected range can be explained by the fact that (1) we used pessimistic assumptions to derive Equation (5) and (2) certain correlations between the detectability of some faults may exist but were not considered.

Our second conclusion from both the analytic and empirical results is that the bounds given by the linear distribution model are overly pessimistic and Equation (5) will give more optimistic results.

Given that the actual results tended to be in the high end of the expected range, we conclude that with only a few random selections sequences of length at most $2L$ can be found that produce 100% fc.

Table 3: Number of random sequences of length less than $(2)L$, that detected all faults

circuit	expc$\leq L$	$\leq L$	$L <, \leq 2L$	$\leq 2L$	expc$\leq 2L$
bc0	0	1	23	24	16-18
b3	7-13	12	49	61	57-61
chkn	0-4	1	2	*	34-50
cps	0	0	36	36	1
exep	0	0	14	14	10-12
in3	7-14	0	50	50	32-61
in4	10-13	17	47	64	57-61
in5	3-6	8	64	72	66-69
in7	10	9	37	46	48
vg2	0	0	18	18	2-17
vtx1	0	0	6	6	0-9
x1dn	0	0	2	2	0-9
x9dn	0	2	10	12	0-12

5 Discrete logarithms and LFSR sequences

Up to this point it was shown that the probability that a pseudo-random test sequence of length at most $2L$ achieves 100% fc is typically greater than $\frac{1}{50}$. In the sequel we show a more sophisticated way of selecting the primitive feedback polynomial and seed of the LFSR-based PG which finds shorter sequences in less or competitive time. Given a specific feedback polynomial, we guide the search for the optimal seed by using the theory of *discrete logarithms.*

When initialized to a non-zero state, a n-stage LFSR with a primitive feedback polynomial cycles through all $2^n - 1$ non-zero binary n-tuples. When the feedback polynomial is changed from one primitive polynomial to another the order in which the patterns appear changes. Hence, a n-stage LFSR with a primitive feedback polynomial defines a *permutation* over the non-zero binary n-tuples, each polynomial corresponding to a different permutation.

Given a subset of binary n-tuples, the minimum subsequence of a LFSR needed to cover all the tuples of the subset will vary depending on the permutation defined by the feedback polynomial. If the position of the tuples in the permutation was known, it would be straight forward to find the minimum covering subsequence. The position of a tuple in a sequence can be obtained from the theory of *discrete logarithms.*

Let $f(x) = \sum_{i=0}^{n} f_i x^i = x^n + h(x)$ be a primitive polynomial of degree n over $GF[2]$ and let

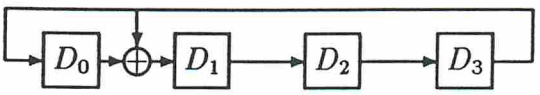

Figure 1: A LFSR with the feedback polynomial $f(x) = x^4 + x + 1$.

α be a root of f. The non-zero elements of $GF[2^n]$ can all be expressed as distinct powers of α, i.e.

$$\forall \beta \neq 0 \in GF[2^n],\ \exists\ 0 \leq j \leq 2^n - 2\ \ s.t.\ \beta = \alpha^j \tag{8}$$

Since α is a root of f we have $\alpha^n = h(\alpha)$, hence β can be represented as a unique non-zero polynomial in α of degree less than n.

If β is the j-th power of α, j is said to be the *discrete logarithm* of β to the base α. The discrete logarithm problem can be stated as follows: given α (equivalently, given f) and β, find j.

Discrete logarithms relate to the order in which patterns are generated by a LFSR as follows. Consider a LFSR with f as its feedback polynomial, as in Figure 1. We number the cells $D_0, D_1, \ldots, D_{n-1}$, with the feedback value coming out of cell D_{n-1} and feeding D_i iff $f_i = 1$. Consider the following mapping between non-zero patterns of the register and non-zero polynomials in α of degree less than n. A "1" in cell D_i represents α^i. Since every non-zero element of $GF[2^n]$ corresponds to a unique polynomial in α of degree less than n, the non-zero elements of $GF[2^n]$ correspond to the patterns of the register. Assume the initial state has a "1" in the least significant cell and 0 in all other cells. A shift will move the "1" to the second cell, corresponding to multiplication by α. After the n-th shift, the "1" is fed back, corresponding to substituting $h(\alpha)$ for α^n. Thus, the j-th pattern, interpreted as a polynomial in α, represents the element α^j. If we want to know when a pattern will appear we need to find the discrete logarithm of the element corresponding to the pattern.

We considered two algorithms for solving the discrete logarithm problem. The first is due to Pohlig and Hellman [16] and the second is due to Coppersmith [5]. An excellent expository of these two algorithms and of the discrete logarithm problem can be found in [15].

To avoid a lengthy discussion on the exact analysis of the computational complexity of these two algorithms we refer the reader to the above references. We just state the issues

which influenced our decision to prefer the Pohlig-Hellman algorithm over Coppersmith's. While Coppersmith's algorithm has an asymptotic run time which is better than the Pohlig-Hellman algorithm, it is also much more complex. The computational complexity of the Pohlig-Hellman algorithm is dependent on the size and multiplicity of the prime factors of $2^n - 1$ (the number of non-zero elements in the field of computation). When the number of circuit inputs is less than 64, which is true for the cases we are targeting, the prime factors of $2^n - 1$ are small enough (except for n = 62,61,59, 49,41,37,31) for the Pohlig-Hellman algorithm to run faster than Coppersmith's.

6 The test embedding problem

In this section we define and present an algorithm to solve the *test embedding problem.* We analyze the computational complexity of our algorithm and discuss a strong limitation. We then present a way to overcome this limitation.

The *test embedding problem* is defined as follows. Given (1) a set of faults $F = \{f_1, \ldots, f_s\}$, (2) a set of test patterns $T = \cup_{i=1}^{s} T_i$, with $T_i = \{t_{i,1}, \ldots, t_{i,n_i}\}$ being the set of *all* tests for fault f_i, and $t_{i,j}$ being a binary n-tuple, and (3) a primitive polynomial, p, of degree n, find a minimum length subsequence of patterns generated by the corresponding LFSR that includes at least one test pattern for each f_i in F.

When given a set $P = \{p_1(x), \ldots, p_u(x)\}$ of primitive polynomials we would like to select the polynomial that generates the shortest such subsequence.

Having found the polynomial, the initial state of the LFSR and the sequence length, we have *embedded* a test set for F in the sequence that will be generated by the LFSR. The resulting test sequence is referred to as the *embedded (test) sequence.* The faults in F are referred to as *embedded faults* and the test patterns of T are the *embedded test patterns.*

To solve this problem we propose a two-stage procedure, referred to as the *embedding procedure (EP).* The first stage is referred to as the *discrete logarithm stage* and the second stage is referred to as the *windowing stage.*

In the *discrete logarithm stage* we first find the logarithm of all the test patterns $t_{i,j}$ with respect to a root α of the primitive polynomial p. We then sort the logarithms and for each logarithm we create a list of faults detected by the corresponding pattern.

In the *windowing stage* the idea is to use a sliding window on the cycle of logarithms, identifying those windows that detect all the faults, and selecting the smallest window (or one of the windows of shortest length when there are more than one). The outline of the procedure for this stage is given in Figure 2.

The array *covered*[] keeps track of the number of patterns in the window that detect each fault. The counter *not_covered* keeps count of the number of faults that are not detected by the patterns in the window. The array *log_table*[] stores the ordered logarithms, while the array *pattern_table*[] stores the patterns corresponding to the ordered logarithms. The procedure begins with the window containing the first logarithm. It is then extended, until all the faults are detected. The size of the window is recorded. It is then compared with the previous best and the smaller of the two is kept. At the end of each iteration the *tail* is advanced. In the following iteration the *head* is adjusted, if necessary, so that the window detects all the faults. The procedure terminates when all the logarithms have been considered as *tails*.

Denote the ordered logarithm cycle by $lg_1, lg_2, \ldots, lg_{|T|}$ where lg_i is followed by lg_{i+1} and $lg_{|T|}$ is followed by lg_1. Denote the window whose *tail* is lg_i by w_i and denote the *head* of w_i by h_i. We have the following Lemma.

Lemma 1: The head of w_{i+1}, h_{i+1}, is in the subcycle $[h_i \ldots lg_{i+1})$, i.e. it is not in the subcycle $[lg_{i+1} \ldots h_i)$.

Proof: Assume h_{i+1} is in the subcycle $[lg_{i+1} \ldots h_i)$, then the window $[lg_i \ldots h_{i+1}]$ detects all faults, contradicting the minimality of w_i. □

As a result of Lemma 1, Procedure *window*() finds the shortest window associated with each *tail*. Since the procedure considers windows beginning at all the logarithms and considers the logarithms of all the test patterns for all the faults, the procedure finds a smallest subsequence that detects all faults.

The complexity of Procedure *window*() is a function of the number of *tail* and *head* movements taken. There are at most $|T|$ logarithms, hence at most $|T|$ *tail* movements. Each window contains at most $|T|$ logarithms, hence there are at most $2|T|$ *head* movements. For each *tail* or *head* movement there are at most $|F|$ accounting operations that are needed, hence the complexity is $O(|T||F|)$.

Procedure 1: $window(n, \#\_of\_flts, \#\_of\_logs, log\_table, pattern\_table)$

1. $best = 2^n - 1$
2. for $(i = 0; i < \#\_of\_flts; i++)$
   (a) $covered[i] = 0$
3. $not\_covered = \#\_of\_flts$
4. $end = \#\_of\_logs - 1$
5. for $(start = 0; start < \#\_of\_logs; start++)$
   (a) $tail = log\_table[start]$
   (b) while$(not_covered > 0)$
      i. $head = log\_table[(++end) \pmod{\#\_of\_logs}]$
      ii. for all faults j covered by $head$
         A. if$(covered[j] == 0)$, $not\_covered--$
         B. $covered[j]++$
   (c) $size = head - tail + 1 \pmod{2^n - 1}$
   (d) if $best > size$
      i. $best = size$
      ii. $seed = pattern\_table[start]$
   (e) for all faults j covered by $tail$
      i. if$(covered[j] == 1)$, $not\_covered++$
      ii. $covered[j]--$

Figure 2: The procedure for the *windowing stage* of *EP*

The complexity of EP is given by $O(PPDL + |T|(DL + \log|T| + |F|))$, where $PPDL$ is the preprocessing effort required for the Pohlig-Hellman algorithm, DL is the time required for one logarithm computation and $|T|\log|T|$ is the time required to sort the logarithms.

As mentioned in the previous section, for the cases we are targeting $PPDL$ and DL will not require much effort. The bulk of the work required by $PPDL$ is the construction of a hash table of size $2 \cdot \sum_{i=1}^{m} q_i$, where m is the number of distinct prime factors of $2^n - 1$ and the q_i's are the distinct prime factors. Each entry involves one polynomial multiplication and one polynomial reduction modulo p and one insert operation into the table. The work required for one DL operation is $O(mn)$ polynomial multiplications and reductions modulo p and m integer multiplications and reductions.

The major factor in the complexity expression is the number of test patterns, $|T|$, which might be overwhelming, rendering the algorithm impractical. We must, therefore, find a way to limit the size of T. This is done in two ways, based on a *user-set limit* on the effort allowed for embedding a fault. The first is by considering only a subset of all possible faults, those which will be classified as *hard*. For some circuits the set of hard faults will be empty. These circuits are classified as *randomly testable* circuits and no embedding is done for them. A short test sequence for these circuits is found by random selections. The second is by limiting for each *hard* fault the number of test patterns that will participate in EP (whose logarithms we compute). This limit will typically apply to only a small subset (if at all) of the hard faults. In the next section we describe our heuristic for identifying hard faults.

7 Identifying hard faults

The amount of work needed for EP is strongly affected by the number of test patterns. We have to modify the procedure in a way that will reduce the test set to a manageable size. The modification is based on partitioning the set of faults in a circuit into two sets - *randomly testable faults* and *random resistant faults*, with each set being possibly empty. By *randomly testable faults* we mean faults that can be detected by choosing a primitive feedback polynomial and an initial seed at random, and using a test sequence of *acceptable length* (this term will be defined later). By *random resistant faults* we mean faults that cause the test length (for 100% fault coverage) to dramatically increase beyond the *acceptable length* when the feedback polynomial and the initial seed are chosen at random.

This partition is justified by the super-exponential decrease in the probability that a fault is missed by a random test sequence as the number of test patterns for the fault is doubled.

Our thesis is that by proper identification of the *random resistant faults* (referred to as *hard faults* in the sequel), and by embedding test patterns for each of these faults in a LFSR subsequence, the subsequence will also detect all the *randomly testable faults.*

As was shown in Section 2, the probability that a random sequence does not detect a fault in C_k or C_{k+1} is (much) greater than the probability that the sequence does not detect any other fault, hence any procedure that identifies hard faults must be able to identify the faults in both these subsets. We refer to the union of C_k and C_{k+1} as HF.

Before presenting our procedure for identifying hard faults, we present a brief discussion on the applicability of our algorithm. Every circuit can be associated with parameters (n, k). These parameters determine the *work factor* for *EP* and the *predicted length* of the embedded test sequence. We expect the embedded sequence to be of length between $L/2 = 2^{n-k-1}$ and $L = 2^{n-k}$. When embedding test patterns we ensure that in the resulting test sequence the embedded faults will be detected and we rely on chance that the non-embedded faults will also be detected. It follows that we must embed the faults in HF. Assuming little overlap between the test sets for each of the faults, the embedded test set is at least of order $|T_{emb}| = 2^k \cdot |C_k| + 2^{k+1} \cdot |C_{k+1}|$. The size of T_{emb} increases with k, whereas the predicted test length decreases. The applicability of our algorithm depends on the amount of preprocessing work (embedding) we are willing to do and the amount of time we are willing to allow the test session (test length).

The bound we place on the amount of pre-processing work determines what we consider as a *hard* circuit and what we consider as an *easy* circuit. If we allow each embedded fault at most 2^δ test patterns, then for a circuit for which $k = \delta$ we expect to find an embedded sequence of length longer than $2^{n-\delta-1}$. Thus, if for a given circuit we find a random test sequence of length less than $2^{n-\delta-1}$, we consider the circuit to be randomly testable, i.e. *easy*. In our experiments we chose to allow each embedded fault at most $2^{13} - 2^{14}$ test patterns. This led us to classify circuits as *easy* when we found a random test length that was less than 2^{n-15} (the *acceptable length*).

The bound we impose on the test length determines whether or not the resulting embedded sequence is applicable to a circuit. For example, if for a 32 input circuit we find that the hardest faults have 2^6 test patterns, we expect to find a test sequence of length $2^{25} - 2^{26}$. If the maximum

allowed test length is 2^{20}, then our scheme is not practical for this circuit. On the other hand, no other scheme that uses just one seed and no reconfiguration of the LFSR will detect 100% of the faults within the imposed time constraints. We denote the maximum allowed test length by 2^{max}.

In the remainder of this section we present our heuristic for identifying hard faults followed by a probabilistic analysis of its effectiveness. The experimental results validate the proposed process.

7.1 The heuristic

The process by which we identify hard faults (referred to as *IHF*) is made of three phases. In the first phase we determine which of the faults of interest are redundant. They are eliminated from the set. The second and third phases are based on simulation (sampling) experiments. The goal of the second phase is to determine an estimate l for k. The goal of the third phase is to identify the random resistant faults using our estimate for k.

The second phase should be a fast process that comes as close as possible to identifying k. The outline of this phase is given in Figure 3. The procedure determines a cutoff value, *undet*, which we set to be the minimum between 50 and 5% of the irredundant faults. It applies random test sequences on the circuit, with increasing lengths, until a sequence that misses at most *undet* faults is found. The initial sequence length is $2^j = 2^{n-15}$. It is understood that $(n - 15) \leq max$, otherwise the procedure will not result in an embedded test sequence we are willing to use, hence there is no sense in carrying it out. If during this iteration a sequence is generated that detects all faults then a satisfying embedding for *all* the faults is found and the circuit is considered *easy*. Otherwise, j is incremented and another fault simulation is performed. This process is repeated (each time simulating all faults) until the number of undetected faults is less than *undet* or until j is greater than max. If $j > max$ the procedure stops, it will not produce a satisfying test sequence. Otherwise four more fault simulation experiments of length 2^j are conducted. For each of the five experiments the set of undetected faults is recorded and at the end of the experiments the union of these sets is constructed. For each fault in the union all test cubes are generated in order to find the fault with the fewest number of test patterns. [1]

[1] A test pattern is a binary vector, indicating exact values in every position, whereas a test cube is a ternary vector, the third symbol being a *don't care* symbol.

Procedure 2: *second_phase*$(circuit, k, max, \#irredundant\_flts)$

1. $undet = \min\{50, 0.05 \cdot \#irredundant\_flts\}$
2. for$(j = n - 15 \ ; \ j \leq max \ ; \ j++)$
   (a) $ND = \{$ faults not detected by a random sequence of length 2^j $\}$
   (b) if$((j = n - 15)$ and $(ND = \phi))$ return(-2) /* the circuit is *easy* */
   (c) if $|ND| < undet$, break
3. if $(j > max)$ return(-1) /* the expected sequence length is unacceptable */
4. $UND = ND$
5. for$(i = 0 \ ; \ i < 4 \ ; \ i++)$
   (a) $ND = \{$ faults not detected by a random sequence of length 2^j $\}$
   (b) $UND = UND \cup ND$
6. $t = \min_i\{|T_i|\}_{i=1}^{|UND|}$ where T_i is the set of all test patterns for the i-th fault in UND.
7. return$(\lceil \log t \rceil)$

Figure 3: The procedure for the second phase of identifying hard faults

If the number of test patterns for this fault is t, then the estimate of k is $l = \lceil \log t \rceil$.

Remark 1: Although test generation is considered a hard problem, and, a fortiori, generation of all test patterns, this proved to be a very low cost task in our experiments, where the faults of interest were single stuck-at faults.

After the execution of Procedure *second_phase()*, a circuit is classified as either *easy* or *hard*. It is classified as *easy* if one of two conditions is met: (1) a random test sequence of length less than 2^{n-15} was found that detects all faults; or (2) the estimate l is greater or equal to 15. In all other instances a circuit is classified as *hard*.

For circuits classified as *hard*, we turn to the third phase. We run a set of fault simulation experiments (we ran 20) with $L/2 = 2^{n-l-1}$ test patterns. *The set of faults which are not detected by at least one experiment constitute the set of embedded faults (EF).* These are the faults for which we embed test patterns. It is essential that the faults in EF include all the faults of HF. The parameters used in selecting the number of simulation runs in phases two and three were chosen to ensure that the probability that EF includes HF is very close to 1 (this is shown in the next section). If a fault f is included in EF, we say it is *classified as hard*. Otherwise, we say it is *classified as easy*.

Remark 2: We conduct 5 fault simulation experiments in phase two in order to increase the probability that at least one fault of C_k will be in the union of the undetected faults. It is important that l be as close as possible to k because this will reduce the number of faults we have in EF (the smaller l is, the longer the simulation lengths in phase three are, decreasing the probability of not detecting a random fault, hence fewer faults are in EF), thus reducing the number of test patterns we need to embed. While this also reduces the probability of detection for some of the faults (as will be seen in the next section), it is a tradeoff that must be made.

In the sequel, whenever EP is mentioned, it is understood that IHF was used in a preprocessing step (stage 0 if you will) to create a reduced target fault set.

7.2 Probabilistic analysis of the heuristic

The purpose of IHF is twofold. First, to find all the faults in HF. Second, by our thesis, embedding test patterns for these hard faults results in a test sequence that detects all the

faults of interest. We must answer two questions regarding our heuristic.

1. What is the probability that a fault that *should* be classified as *hard* (i.e. has less than 2^{k+1} test patterns) is classified as *easy*?

2. What is the probability that an *easy* fault will not be detected by the embedded sequence?

In answering these questions we assume that the embedded sequence is made of a totally random set of patterns, although, given that petterns are generated by a LFSR, this might not be completely accurate.

In answering the first question we compute the probability that a fault with t test patterns will not be classified as *hard* by *IHF*. We will be interested in the probability value when t is less than or equal to 2^{l+1}, where l is the *IHF* estimate for k.

Let $p_{miss}(2^{n-l-1}, t)$ be the probability that a test sequence of length 2^{n-l-1} does not detect a fault with t test patterns. This would require that the first pattern not detect the fault, the second not detect the fault, and so on. Assuming the patterns are completely random, uncorrelated, with selections done without replacement and the all zero pattern does not participate in the drawing of patterns (and is not one of the test patterns), we have

$$\begin{aligned} p_{miss}(2^{n-l-1}, t) &= \frac{(2^n-1)-t}{(2^n-1)} \cdot \frac{(2^n-2)-t}{(2^n-2)} \cdot \ldots \cdot \frac{(2^n-2^{n-l-1})-t}{(2^n-2^{n-l-1})} \\ &= \left(1-\frac{t}{(2^n-1)}\right) \cdot \left(1-\frac{t}{(2^n-2)}\right) \cdot \ldots \cdot \left(1-\frac{t}{(2^n-2^{n-l-1})}\right). \end{aligned}$$

Let $p_{detect}(2^{n-l-1}, t)$ denote the probability that a sequence of length 2^{n-l-1} detects a fault with t test patterns, then

$$p_{detect}(2^{n-l-1}, t) = 1 - p_{miss}(2^{n-l-1}, t).$$

Let $\#sim$ denote the number of simulation experiments. The probability, $p_{all\_det}(2^{n-l-1}, t)$, that a fault with t test patterns is detected by all the experiments is

$$p_{all\_det}(2^{n-l-1}, t) = (p_{detect}(2^{n-l-1}, t))^{\#sim}.$$

$p_{all\_det}(2^{n-l-1}, t)$ is also the probability that a fault with t test patterns will not be classified as hard. This probability can be increased or decreased by changing the number of simulation experiments. In our experiments, $p_{all\_det}(2^{n-l-1}, 2^l)$ ranged from $7.9 \cdot 10^{-9}$ to 10^{-8}. $p_{all\_det}(2^{n-l-1}, 2^{l+1})$ ranged from 10^{-4} to $1.25 \cdot 10^{-4}$. This suggests that with very high probability EF will include all the faults in HF. EF will typically include some other faults, those with higher detection probabilities than the faults in HF, thus although EF might vary from one run of EP to another, all sets will share the same core of "hardest" faults.

We turn to the second question, namely what is the probability, $p_{nd}(el, t)$, that an embedded sequence, es, of length el does not detect a fault f, where there are t possible test patterns for f. We first answer the complement question, that is, what is the probability that es detects f.

$$prob(es \text{ detects } f) = prob(es \text{ detects } f \mid f \text{ is classified } hard) \cdot prob(f \text{ is classified } hard) +$$

$$prob(es \text{ detects } f \mid f \text{ is classified } easy) \cdot prob(f \text{ is classified } easy).$$

By construction, es contains at least one test patterns for each of the hard faults, hence

$$prob(es \text{ detects } f \mid f \text{ is classified } hard) = 1.$$

By definition

$$prob(f \text{ is classified } hard) = 1 - p_{all\_det}(2^{n-l-1}, t).$$

The probability $prob(es \text{ detects } f \mid f \text{ is classified } easy)$ is the probability that a random sequence of length el detects f. By definition

$$prob(f \text{ is classified } easy) = p_{all\_det}(2^{n-l-1}, t).$$

Thus,

$$prob(es \text{ detects } f) = 1 \cdot (1 - p_{all\_det}(2^{n-l-1}, t)) + p_{detect}(el, t) \cdot p_{all\_det}(2^{n-l-1}, t)$$

$$= 1 - p_{all\_det}(2^{n-l-1}, t) + p_{all\_det}(2^{n-l-1}, t) -$$

$$p_{miss}(el, t) \cdot p_{all\_det}(2^{n-l-1}, t)$$

$$= \quad 1 - p_{miss}(el,t) \cdot p_{all\_det}(2^{n-l-1},t).$$

Hence

$$p_{nd}(el,t) = p_{miss}(el,t) * p_{all\_det}(2^{n-l-1},t).$$

From the above expression the probability of f not being detected is the probability that f is classified as *easy* (p_{all_det}) and a random sequence of length el fails to detect f (p_{miss}).

The probability $p_{nd}(el,t)$ is a product of two functions of t. But whereas one of these functions (p_{all_det}) increases with t, the other (p_{miss}) decreases. We focus our attention on the behavior of p_{nd} as a function of t. As will be seen in Theorem 2, p_{nd} has a maximum value and no local maxima, i.e. as t increases, p_{nd} rises, reaches a maximum value and decreases from there on. The overall probability that the embedded sequence detects all the faults is dependent upon the probability of occurrence of the faults whose number of test patterns is in the peak area. The values of $p_{nd}(el,t)$ for $t = 2^j$, $l \leq j \leq l+5$, from our experiments are tabulated in Table 10. The peaks can clearly be seen.

Let $N = 2^n$, $r = 2^{n-l-1}$, $q = el$ and $s = \#sim$. We can rewrite p_{nd} as follows

$$p_{nd}(N,q,t,r,s) = \prod_{v=N}^{N-q+1} \left(\frac{v-t}{v}\right) \left[1 - \prod_{u=N}^{N-r+1} \left(\frac{u-t}{u}\right)\right]^s$$

We assume $t \leq N - q$ since if $t > N - q$, then a test sequence of length q will always have to include at least one of the t test patterns, hence p_{nd} will be zero.

Theorem 2 shows that once p_{nd}, as an integer function in t, either levels off or starts falling, it will keep on falling, hence it has one maximum value with no local maxima.

Theorem 2: If $p_{nd}(N,q,t,r,s) \geq p_{nd}(N,q,t+1,r,s)$ then

1. For all i s.t. $t+1 \leq i < \min\{N-q, N-r\}$

$$p_{nd}(N,q,i,r,s) > p_{nd}(N,q,i+1,r,s)$$

2. If $N - r < N - q$, then for all i s.t. $t + 1 < N - r \leq i < N - q$

$$p_{nd}(N, q, i, r, s) > p_{nd}(N, q, i + 1, r, s)$$

□

Proof: We look at the ratio $p_{nd}(N, q, t, r, s)/p_{nd}(N, q, t + 1, r, s)$ and we show that if this ratio is greater or equal to 1, then all succeeding ratios will be greater than 1. We begin with the first part of the theorem.

$$\frac{p_{nd}(N, q, i, r, s)}{p_{nd}(N, q, i + 1, r, s)} = \frac{\prod_{v=N}^{N-q+1} \left(\frac{v-i}{v}\right) \left[1 - \prod_{u=N}^{N-r+1} \left(\frac{u-i}{u}\right)\right]^s}{\prod_{v=N}^{N-q+1} \left(\frac{v-(i+1)}{v}\right) \left[1 - \prod_{u=N}^{N-r+1} \left(\frac{u-(i+1)}{u}\right)\right]^s}$$

Denote

$$U_i = \prod_{u=N}^{N-r+1} \left(\frac{u - i}{u}\right)$$

then

$$\frac{p_{nd}(N, q, i, r, s)}{p_{nd}(N, q, i + 1, r, s)} = \frac{N - i}{N - i - q} \left[\frac{1 - U_i}{1 - U_{i+1}}\right]^s.$$

By the hypothesis, for $i = t$, the ratio is greater or equal to 1, hence

$$\left[\frac{1 - U_i}{1 - U_{i+1}}\right]^s \geq \frac{(N - i - q)}{(N - i)}$$

$$= 1 - \frac{q}{N - i}.$$

Denote the above inequality as

$$M_i^s \geq Q_i.$$

For $i \geq t$, $Q_i > Q_{i+1}$. If we show that $M_{i+1} > M_i$ then we get

$$M_{i+1}^s > M_i^s \geq Q_i > Q_{i+1}$$

which proves the first statement. To show that $M_{i+1} > M_i$ we need to show that

$$(1 - U_{i+1})^2 - (1 - U_i)(1 - U_{i+2}) > 0. \tag{9}$$

We use the fact that

$$\frac{U_i}{U_{i+1}} = \frac{N-i}{N-i-r} = 1 + \frac{r}{N-i-r}$$

and that

$$\frac{U_{i+2}}{U_{i+1}} = \frac{N-i-r-1}{N-i-1} = 1 - \frac{r}{N-i-1}$$

to show

$$\begin{aligned}(1 - U_{i+1})^2 - (1 - U_i)(1 - U_{i+2}) &= \frac{r(r-1)}{(N-i-1)(N-i-r)}U_{i+1} + \\ &\quad \frac{r}{(N-i-1)(N-i-r)}U_{i+1}^2 \\ &> 0.\end{aligned}$$

The last inequality in based on the fact that $i < min(N-q, N-r)$.

To prove the second statement, notice that for $i = N - r$, U_{i+1} and U_{i+2} are both equal to zero, hence Equation (9) becomes $U_i > 0$ and there is nothing to prove. For $i > N - r$, a test sequence of length r will always include at least one of the i test patterns, hence p_{all_det} is equal to 1. The ratio becomes

$$\frac{\prod_{v=N}^{N-q+1}\left(\frac{v-i}{v}\right)}{\prod_{v=N}^{N-q+1}\left(\frac{v-i-1}{v}\right)} = \frac{N-i}{N-q-i} > 1$$

□

8 Experimental results

Experiments were conducted using a modified versions of the ATALANTA fault simulation and test generation system [10]. These modifications included fault simulating random patterns generated by LFSRs with primitive feedback polynomials, and generating all test cubes per

Table 4: Experimental results for some ISCAS85 combinational benchmarks.

circuit	#pi	#po	#gates	depth	#nr.flts	sp_time	best	worst	$\#pi - 15$
c432	36	7	196	18	520	1	480	3136	21
c499	41	32	243	12	750	1	896	1536	21
c880	60	26	443	25	942	1	7520	34176	45
c1355	41	32	587	25	1566	2	1344	2848	26
c1908	33	25	913	41	1870	4	5216	33152	18
c3540	50	22	1719	48	3291	3	8960	50016	35
c6288	32	32	2448	125	7710	13	64	256	17

fault (or as many as ATALANTA could find, depending on the allowed backtrack limit) instead of stopping once the first test pattern is found.

We first tried *EP* on the ISCAS85 benchmark circuits. The characteristics of some of these circuits, as well as the results of the experiments are given in Table 4. The first five columns of the table are self explanatory. The sixth column (#nr.flts.) gives the number of irredundant faults for each circuit. All of these circuit were classified as *easy* by Procedure *second_phase*(). Column seven (sp_time) is the CPU time (in seconds) required for Procedure *second_phase()* on a SPARC 2 workstation.

Since these circuits were classified as *easy*, we did not continue on with *EP*. To get an idea of the required pseudo-random test length for these circuits, we conducted 10 *random selection* experiments (referred to as *RS* experiments in the sequel) for each circuit. In each RS experiment a primitive feedback polynomial and an initial seed were chosen at random and test patterns generated until 100% fault coverage was achieved. Columns eight and nine give the length of the best (shortest) and worst (longest) sequence that was required for each of the circuits. By looking at the values in column ten, we see that the sequence lengths that were found are much smaller than 2^{n-15}.

We did not try to embed test sequences for circuits $c2670\,(n = 233)$, $c5315\,(n = 178)$, and $c7552\,(n = 207)$ because the number of inputs for each of these circuits made the embedding impractical. These circuits are either randomly testable or the embedded test length would be so long (greater than 2^{n-15}) that test set embedding would not be a viable test option.

Since the ISCAS circuits do not provide a good test bed for our algorithm, we turned to the Berkeley circuits [4]. We synthesized circuits with 20 to 40 inputs as multilevel circuit. Some

Table 5: Characteristics of synthesized circuits.

circuit	#pi	#po	#gates	depth	#nr.flts	l	sp_time
b4	33	22	171	9	562	*easy*	2
in6	33	23	172	9	564	*easy*	2
jbp	36	57	353	13	1051	*easy*	3
misj	35	12	63	5	89	*easy*	1
signet	39	8	243	8	709	*easy*	1
x6dn	38	5	207	13	651	*easy*	2
bc0	21	11	372	15	1434	4	21
b3	32	19	415	17	1283	12	63
chkn	29	7	282	13	959	5	997
cps	24	109	486	18	1837	3	890
exep	28	62	256	12	995	11	21
in3	34	29	240	14	714	14	77
in4	32	20	422	23	1303	12	70
in5	24	14	192	14	630	10	6
in7	26	10	117	12	330	9	5
vg2	25	8	189	12	576	7	8
vtx1	27	6	207	12	616	7	72
xparc	39	69	729	21	2614	12	*
x1dn	27	6	207	12	616	7	34
x9dn	27	7	215	12	636	7	83

circuits were eliminated for pathological reasons.

The characteristics of the synthesized circuits are given in columns two (#pi) to six (#nr.flts) of Table 5. Column eight (sp_time) is the CPU time, in seconds, required to run the second phase on a SPARC 2 workstation. The second and third phase of IHF were combined for circuit *xparc*. The reason for this was the long simulation time required for this circuit. Thus, no time is reported for this circuit in Table 5.

To identify the randomly testable circuits from the random resistant ones we ran each circuit through *IHF*. The simulation experiments of the second and third phase of *IHF* used a pool of 150 primitive polynomials. For a circuit with n inputs we used the first 150 primitive polynomials of degree n, when ordered lexicographically (i.e. $(x^n + x + 1) > (x^n + 1)$).

The results of Procedure *second_phase()* on these circuits is given in column seven (l) of Table 5. For each of the *easy* circuits we ran 10 RS experiments to find a short pseudo-random test sequence. We also conducted a more thorough search for k for these circuits. We allowed

Table 6: RS results for the randomly testable synthesized circuits.

circuit	$n-l$	best	worst	$n-15$
b4	15	48,244	130,016	18
in6	15	55,712	130,016	18
jbp	13	20,064	32,768	21
misj	10	672	4096	20
signet	13	7,232	32,768	24
x6dn	14	22,368	65,536	23

each RS experiment to generate at most 2^{n-l+1} test patterns, where l is the estimate for k. The results of these experiments are in Table 6. The value of $n-l$ is given in column two. The best (shortest sequence) and worst (longest sequence) results are given in columns three and four. None of the worst sequences achieved 100% fault coverage. The total CPU time needed for these experiments was $1-2$ minutes, as reported by ATALANTA on a SPARC 1 workstation. The best random sequence that was found was always shorter than 2^{n-l+1} $(2L)$. Also, $n-l+1$ was always less than $n-15$ (column five).

For the circuits classified as *random resistant circuits* we tried to find short test sequences in two ways. The first was by RS experiments and the second was by continuing our embedding procedure (the first two phases of *IHF* were already executed).

The purpose of finding a minimum sequence in two ways was to evaluate the quality of the result of *EP* in terms of test length and processing time.

We ran 100 RS experiments for each circuit, except for circuit *chkn* for which we ran just 50 experiments and circuit *xparc* for which we ran just 3 experiments. The results of these experiments are in Table 7. Column *rs_time* represents the total CPU time (read as $hrs : min$) needed for these experiments, as reported by ATALANTA on a SPARC 2 workstation. We allowed each RS experiment to run for at most 2^{n-l+1} test patterns. For all the circuits, the worst sequence lengths are the maximum allowed lengths per experiment and all such sequences did not detect all the faults. Comparing the best sequence length and the value 2^{n-l}, we see that the best value was less than 2^{n-l} (L) for 8 of the circuits, while it was between L and $2L$ for the remaining 6 circuits.

For the random resistant circuits we completed the execution of *IHF* and proceeded to embed the test patterns for these faults. In Table 8, the results of *IHF* are shown. For each circuit,

Table 7: RS results for the random resistant synthesized circuits.

circuit	#sim.	best	2^{n-l}	worst	rs_time
bc0	100	114,624	131,072	262,144	0:36
b3	100	487,168	1,048,576	2,097,152	6:00
chkn	50	15,561,920	16,777,216	20,000,000	25:15
cps	100	2,292,512	2,097,152	4,194,304	11:46
exep	100	155,584	131,072	2621448	0:32
in3	100	660,224	1,048,576	2,097,152	4:29
in4	100	548,480	1,048,576	2,097,152	5:19
in5	100	11,424	16,384	32,768	0:04
in7	100	56,544	131,072	262,144	0:25
vg2	100	267,552	262,144	524,288	1:01
vtx1	100	1,345,152	1,048,576	2,097,152	4:29
xparc	3	248,802,208	134,217,728	268,435,456	41:36
x1dn	100	1,948,192	1,048,576	2,097,152	4:30
x9dn	100	1,022,944	1,048,576	2,097,152	4:49

we show the total number of irredundant faults, the number of faults that were classified as hard and the percentage of hard faults. The total number of faults that were classified as hard, over all circuits, make up 7% of the total number of faults in the circuits. Thus, when also considering that the hard faults are those with the fewest test patterns, several orders of magnitude of reduction in the work effort is achieved when embedding only the hard faults.

In Table 9 we show the results of *EP* for each of the random resistant circuits. The second column shows the total number of embedded patterns, i.e. the union of the sets of test patterns for the hard faults. Column three shows the number of different primitive polynomials considered, i.e. the minimum length embedded sequence was computed for each of these polynomials. The polynomials we used were the first in the lexicographical ordering. Columns four and five give the best and worst embedding results for these polynomials, and column six gives the total CPU time (read as $hrs : min$) required for *EP*, using an HP-700 workstation. When comparing the best sequence length with the value 2^{n-l} for each of the circuits (see Table 7), we see that the length of the best embedded sequence was always less than 2^{n-l} except for the circuit *exep*. For six of the circuits, the worst sequence length was also less than 2^{n-l}. When comparing the worst embedding length with the best RS length, for 6 of the circuits the worst embedding length was shorter than the best RS length.

Table 8: Hard fault identification for synthesized circuits.

circuit	#nr.flts.	#h.flts.	%	circuit	#nr.flts.	#h.flts.	%
bc0	1434	65	4.53	b3	1283	29	2.26
chkn	959	88	9.18	cps	1837	388	21.12
exep	995	86	8.64	in3	714	16	2.24
in4	1303	28	2.15	in5	630	77	12.22
in7	330	8	2.42	vg2	576	15	2.60
vtx1	616	29	4.71	xparc	2614	130	4.97
x1dn	616	28	4.55	x9dn	636	43	6.76

Table 9: Embedding results for the synthesized circuits.

circuit	#emd.vecs.	#pols.	best	worst	emb_time
bc0	3,120	50	74,240	196,741	0:30
b3	171,707	4	385,824	614,368	7:01
chkn	13,328	20	9,459,968	12,551,540	1:02
cps	9,024	25	1,254,272	3,464,351	1:27
exep	281,235	2	149,120	150,912	4:22
in3	148,683	4	176,960	239,296	3:27
in4	194,556	3	530,752	617,600	5:51
in5	176,310	3	6,530	7,416	3:22
in7	9,728	15	9,274	29,488	0:24
vg2	2,556	100	111,456	320,438	0:35
vtx1	5,216	25	838,176	1,621,458	0:41
xparc	536,818	1	133,667,296	*	13:20
x1dn	5,216	25	838,176	1,621,458	0:41
x9dn	6,144	25	446,624	1,200,397	0:45

Table 10: $p_{nd}(el,t)$ values for $t = 2^j$, $l \leq j \leq l+5$.

circuit	(n,l)	$(el,2^l)$	$(el,2^{l+1})$	$(el,2^{l+2})$	$(el,2^{l+3})$	$(el,2^{l+4})$	$(el,2^{l+5})$
bc0	(21,4)	5.67(−9)	3.90(−5)	5.99(−3)	7.01(−3)	*	*
b3	(32,12)	5.48(−9)	5.00(−5)	1.25(−2)	3.64(−2)	2.76(−3)	8.00(−6)
chkn	(29,5)	5.06(−9)	3.64(−5)	5.89(−3)	7.38(−3)	1.11(−4)	1.24(−8)
cps	(24,3)	6.96(−9)	4.33(−5)	5.53(−3)	5.00(−3)	4.77(−5)	2.30(−9)
exep	(28,11)	2.54(−9)	1.07(−5)	5.76(−4)	7.68(−5)	1.23(−8)	1.52(−16)
in3	(34,14)	6.68(−9)	7.40(−5)	2.78(−2)	1.79(−1)	6.67(−2)	4.51(−3)
in4	(32,12)	4.77(−9)	3.80(−5)	7.21(−3)	1.20(−2)	3.02(−4)	9.23(−8)
in5	(24,10)	5.33(−9)	4.69(−5)	1.11(−2)	2.85(−2)	1.68(−3)	2.85(−6)
in7	(26,9)	7.43(−9)	9.06(−5)	4.12(−2)	3.93(−1)	3.20(−1)	1.04(−1)
vg2	(25,7)	5.33(−9)	4.52(−5)	1.00(−2)	2.30(−2)	1.09(−3)	1.21(−6)
vtx1	(27,7)	3.66(−9)	2.10(−5)	2.24(−3)	1.13(−3)	3.00(−6)	7.18(−12)
xparc	(39,12)	2.93(-9)	1.42(-5)	1.02(-3)	2.39(-4)	1.19(-7)	1.44(-14)
x1dn	(27,7)	3.66(−9)	2.10(−5)	2.24(−3)	1.13(−3)	3.00(−6)	7.18(−12)
x9dn	(27,7)	5.32(−9)	4.50(−5)	1.00(−2)	2.28(−2)	1.08(−3)	1.18(−6)

From the embedding length for each circuit we can calculate the probability $p_{nd}(el,t)$ that an embedded sequence of length el will not detect a fault that has t test patterns, for various values of t. This is shown in Table 10. The values in the parenthesis in each table entry are exponents of 10. For example, the value of $p_{nd}(el,2^l)$ for circuit *bc0* is $5.67 * 10^{-9}$. Depending on the values, one might decide that the probability of not detecting a fault is small enough such that verification by simulation is not necessary.

Having performed the RS experiments and the embedding experiments for the random resistant circuits, we proceeded to compare the results in terms of the shortest sequence length found and the processing time for the experiments. These comparisons are shown in Table 11. Columns two and three give the best test length and the time required for the RS experiments. Columns four and five give these values for the embedding experiments. In column six we give the ratio between the best RS sequence length and the best embedding sequence length. Column seven gives the ratio between the processing time of the RS experiments and the embedding experiments. These ratios were *normalized*, so they would reflect the ratio if both experiments were run on a SPARC 2. To find the normalization factor we ran *EP* for circuit *chkn* on a SPARC 2 workstation and compared the run time with the run time for the same procedure and circuit on the HP-700. The normalization factor came out to be 2. Columns eight, nine and ten show the values of l, $n-l$, and $n-2l$ respectively.

Table 11: Comparison of embedding results and random selection result.

	random selection		embedding		rs/emb				
circuit	test length	time	test length	time	test length	time	l	$n-l$	$n-2l$
in5	11,424	0:04	6,530	3:22	1.75	0.01	10	14	4
exep	155,584	0:32	149,120	4:22	1.04	0.06	11	18	7
b3	487,168	6:00	385,824	7:01	1.26	0.43	12	20	8
in4	548,480	5:19	530,752	5:51	1.03	0.46	12	20	8
in7	56,544	0:25	9,274	0:24	6.10	0.52	9	17	8
bc0	114,624	0:36	74,240	0:30	1.54	0.60	4	17	13
in3	660,224	4:39	176,960	3:27	3.73	0.68	14	20	6
vg2	267,552	1:01	111,456	0:35	2.40	0.87	7	18	11
xparc	248,802,208	41:36	133,667,296	13:20	1.86	1.56	12	27	15
x9dn	1,022,944	4:49	446,624	0:45	2.29	3.21	7	20	13
vtx1	1,345,152	4:29	838,176	0:41	1.60	3.28	7	20	13
x1dn	1,948,192	4:30	838,176	0:41	2.32	3.29	7	20	13
cps	2,169,248	11:46	1,254,272	1:27	1.73	4.06	3	21	18
chkn	15,561,920	25:15	9,459,968	1:02	1.65	12.22	5	24	19

The first fact we can notice from Table 11 is that the test length resulting from *EP* is always shorter than the test length resulting from the RS experiments. The second fact we notice is that the ratio of the processing times required by the two procedures varies. This can be explained as follows. The factor that affects the processing time for *EP* is the number of embedded patterns, which is strongly affected by the value of l. *EP* will require less time for circuits with lower values of l (omitting the obvious influence of the number of embedded faults and the number of polynomials). The factor that affects the processing time for the RS experiments is the length of each experiment, i.e. the value of $(n-l)$. The higher the value of l, the lower the value of $(n-l)$, resulting in lower processing time for the RS experiments. Notice that for the eight circuits whose l-value is less than 10, the time ratio is greater than 1 for five and greater than 0.5 for all eight. In absolute time, *EP* required an hour or less for seven of the eight and an hour and a half for the eighth. Of the eight circuits for which the value $n-2l$ was greater than 10, for six of the eight, *EP* required less time than the RS experiments. Define the product of the test length ratio and the processing time ratio (columns six and seven) as a measure of *EP* effectiveness. When this measure is greater than 1 then any relative advantage *RS* has over *EP* in either length or time is negated by the bigger advantage *EP* has in the other. Except for circuit *bc*0 (whose measure was 0.924), the circuits with $n-2l > 10$ all had effectiveness

measures greater than 1. Of those, only *vg2* had a time ratio less than 1. Circuits *in7* and *in3* also had effectiveness measures greater than 1. Altogether, 9 of the 14 circuits had effectiveness measures greater than 1.

Remark 3: The time reported for *EP* is the time required for the logarithm computations. It does not include the time required for phase three of *IHF*, the time it took to generate the test patterns and sort them before computing the logarithms, nor the time for Procedure *window*(). This is because these times were very small when compared with the time required by the RS experiments or the time required for logarithm computations. The time required for phase three was always between $5\% - 10\%$ of the RS time. For circuits *b3* and *in3*, generating and sorting the test patterns took just over 2 : 30 minutes on a SPARC 2 and Procedure *window*() took just over 4 : 00 minutes for *b3* and less than that for *in3* on an HP-700. These times are insignificant when compared with the RS and logarithm computation times.

One can ask whether we really needed all the 100 RS experiments, or would 20 or 50 have been enough to produce a test sequence of length comparable with the one found by *EP*. A partial answer to this can be found in Table 12. The columns of Table 12 represent ratios between the sequence lengths from the RS experiments and the best embedding result. An integer entry in position (i, j) indicates the number of experiments for circuit i where the ratio was between the value of the headings for the j-th and $(j + 1)$-th columns. For example, for circuit *cps* there was one RS experiments whose ratio was between 1.50 and 1.75. The last non-zero entry in each row includes all the experiments whose ratio was greater or equal to the ratio of the column it sits in, e.g. for circuit *in4* 70 experiments had a ratio greater than 2.50. These 70 experiments include all those that ran for the maximum allowed time and failed to detect all the faults.

The results in this table tend to support the notion that we didn't get "lucky" with random selections and usually all the experiments were necessary to ensure that we find a short sequence length.

For those circuits whose time ratio was less than one, we conducted additional RS experiments to bring the ratio up to one. The results of these experiments are in Table 13. For four of the eight circuits the additional experiments did not find shorter sequences. For two circuit, the additional experiments found shorter sequences, but still longer than the embedded

Table 12: Distribution of simulation length vs. embedding length ratio.

	ratio						
circuit	1.00	1.25	1.50	1.75	2.00	2.25	2.50
bc0	*	*	1	1	2	3	93
b3	*	2	1	1	1	3	92
chkn	*	*	1	2	47	*	*
cps	*	*	1	0	4	4	91
exep	1	5	8	86	*	*	*
in3	*	*	*	*	*	*	100
in4	2	3	6	6	8	5	70
in5	*	*	1	3	1	3	92
in7	*	*	*	*	*	*	100
vg2	*	*	*	*	*	2	98
vtxl	*	*	3	0	2	1	94
xparc	*	*	*	1	2	*	*
xldn	*	*	*	*	*	2	98
x9dn	*	*	*	*	*	2	98

Table 13: Results of additional RS experiments to get equal time comparisons.

circuit	add. exp.	best result	best embedding
in5	9,900	8,192	6,530
exep	1,666	134,848	149,120
		144,864	
b3	132	611,296	385,824
in4	117	456,704	530,752
		501,536	
in7	92	68,064	9,274
bc0	66	163,680	74,240
in3	47	495,168	176,960
vg2	15	456,448	111,456

sequence. For two circuits the additional experiments produced two sequences that had shorter lengths than the embedded sequences. Looking closer at these circuits, the embedding were done relative to only two and three polynomials and their length ratio relative to the first 100 RS experiments was poor to begin with. Thus, there was already indication that the polynomials used for the embeddings were poor choices.

Remark 4: As mentioned in Section 7, our thesis is that the test sequence found for the hard faults will also detect the easy faults. This was true in *all* the embedding experiments except for one polynomial for the circuit *in3*, in which the embedded sequence missed nine irredundant faults, and three polynomials for circuit *in7*, of which two sequences missed one fault and one sequence missed eleven faults.

Remark 5: For circuits *in3, exep, b3, in4, in5, in7* and *xparc*, some of the hard faults had more than the maximum allowed number of test patterns (2^{14} for *in3*, 2^{12} for *exep* and 2^{13} for the others). For these faults we embedded only the maximum allowed number of tests. This caused the predicted length (the distance between the *head* and *tail* of the shortest window) for some of the embeddings to be greater than the actual length. This is a result of the fact that we did not embed *all* possible tests, but only a subset. Additional patterns were found in the embedded sequence, rendering some of the embedded patterns unnecessary, hence the shorter length. Thus, when embedding only subsets of test sets, the length of the embedded sequence is an upper bound on the actual test length. When the complete test sets are embedded the length of the embedded sequence is the actual test length.

9 Conclusions

In this paper we address the question of finding *short* pseudo-random test sequences that achieve 100% fault coverage for LFSR-based PGs. We first show that if the probability of detecting the hardest fault in the circuit is p, then the probability that a pseudo-random test sequence of length $\frac{2}{p}$ will achieve 100% fault coverage is typically greater than $\frac{1}{50}$. We then present an algorithm for embedding test patterns in test sequences generated by LFSRs. The algorithm is based on the theory of discrete logarithms. It produces a one-seed test sequence that detects all the faults of interest. The algorithm can also embed two-pattern tests and can also embed test

patterns in sequences generated by CAs. The advantage of our approach over existing schemes that achieve 100% fault coverage is the low overhead we incur in terms of both circuit area and delay.

The applicability of the embedding algorithm depends on two user specified constraints. The first is the computational effort one is willing to expend (the parameter δ in Section 7.1) and the second is the length of the test session one is willing to allow (the parameter max in Section 7.1). Thus, for circuits where the hardest faults have more than 2^{δ} test patterns the computational effort required will be too high and for circuits with too many inputs, i.e. when $n-\delta-1 > max$, the resulting sequence will be too long to be useful.

We find *short* test sequences either by the embedding algorithm or with random selections. This is decided by classifying circuits as either *randomly estable* or *random resistant.* This classification is based on the number of test patterns for the hardest fault in the circuit. If the logarithm of this number, denoted by k, is greater than δ, or if during the classification process a random sequence of length shorter than $2^{n-\delta-1}$ that achieves 100% fc is found, the circuit is classified as *randomly testable.* In all other cases it is classified as *random resistant.*

For *randomly testable* circuits our probabilistic and empirical results show that short test sequences can be found with few random selections.

For *random resistant* circuits we compared the results achieved by the embedding algorithm with those achieved by random selections of primitive polynomials and seeds for the LFSRs. In all cases the sequences found by our algorithm were shorter than those found by the random experiments. In almost half the cases, our algorithm was also faster than the random experiments. When the PG register is also to function as a RA, by considering only polynomials that achieve *zero-aliasing* [12] as candidates for EP, both objectives are satisfied with one polynomial.

The circuits on which we ran the experiments are relatively small (only several hundred gates). Most of the effort for our algorithm is spent on logarithm computations, and only a small portion is spent on simulation. As circuit sizes increase, the cost of our algorithm will be only slightly affected whereas the cost of the random experiments will increase. Therefore, we expect the effectiveness of the algorithm to increase as circuit size increases.

Acknowledgement: We wish to thank Prof. L. R. Welch for his mathematical suggestions and Prof. D. S. Ha of Virginia Polythechnic for the use of ATALANTA.

References

[1] M.E. Aboulhamid and E. Cerny, *A Class of Test Generators for Built-In Testing*, IEEE Trans. Computers, Vol. C-32, No. 10, October 1983, pp. 957-959

[2] V.K. Agarwal and E. Cerny, *Store and Generate Built-In Testing Approach*, Proc. Intl. Sym. on Fault-Tolerant Computing, pp. 35-40, June 1981

[3] F. Berglez, C. Gloster and G. Kedem, *Built-In Self-Test with Weighted Random Pattern Hardware*, Proc. Intl. Conf. on Computer Design, pp. 161-166, 1990

[4] R.K. Brayton, G.D. Hachtel, C. McMullen and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, MA, 1984

[5] D. Coppersmith, *Fast Evaluation of Logarithms in Fields of Characteristic Two*, IEEE Trans. Inform. Theory, Vol. IT-30, No. 4, pp. 587-594, July 1984

[6] W. Daehn and J. Mucha, *Hardware Test Pattern Generation for Built-In Testing*, Proc. Intl. Test Conf., pp. 110-113, 1981

[7] R. Dandapani, J.H. Patel and J.A. Abraham, *Design of Test Pattern Generators for Built-In Test*, Proc. Intl. Test Conf., pp. 315-319, 1984

[8] C. Dufaza and G. Cambon, *LFSR based Deterministic and Pseudo Random Test Pattern Generator Structures*, Proc. 2nd European Test Conf., pp. 27-34, 1991

[9] G. Edirisooriya and J.P. Robinson, *A New Built-In Self-Test Method Based on Prestored Testing*, Proc. VLSI Test Sym., pp. 10-16, 1993

[10] D.S. Ha, *Automatic Test Pattern Generators and Fault Simulators for Combinational and Sequential Circuits*, Technical Report, Virginia Polytechnic, June 1992

[11] S. Hellebrand, S. Tarnick and J. Rajski, *Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers*, Proc. Intl. Test Conf., pp. 120-129, 1992

[12] M. Lempel and S.K. Gupta, *Zero-Aliasing for Modeled Faults*, CENG TR 94-12, Dept. EE-Systems, University of Southern California. Submitted to IEEE Trans. Computers.

[13] A. Majumdar and S. Sastry, *On the Distribution of Fault Coverage and Test Length in Random Testing of Combinational Circuits*, Proc. Design Automation Conf., pp. 341-346, 1992

[14] F. Muradali, V.K. Agarwal and B. Nadeau-Dostie, *A New Procedure for Weighted Random Built-In Self-Test*, Proc. Intl. Test Conf., pp. 660-669, 1990

[15] A.M. Odlyzko, *Discrete Logarithms in Finite Fields and Their Cryptographic Significance*, Adv. in Cryptology (Proc. of Eurocrypt '84), Lecture Notes in Computer Science, Vol. 209, Springer-Verlag, New York, pp. 224-314, 1984

[16] S.C Pohlig and M. Hellman, *An Improved Algorithm for Computing Logarithms over GF[p] and Its Cryptographic Significance*, IEEE Trans. Inform. Theory, Vol. IT-24, No. 1, pp. 106-110, January 1978

[17] J. Savir and P.H. Bardell, *On Random Pattern Test Length*, IEEE Trans. Computers, Vol. 33, No. 6, pp. 467-474, June 1984

[18] M. Serra, T. Slater, D.M. Miller and J.C. Munzio, *The Analysis of One-Dimensional Cellular Automata and their Aliasing Properties*, IEEE Trans. CAD, Vol. 9, No. 7, pp. 767-778, July 1990

[19] C.W. Starke, *Built-In Test for CMOS Circuits*, Proc. Intl. Test Conf., pp. 309-314, 1984

[20] B. Vasudevan, D.E. Ross, M. Gala and K.L. Watson, *LFSR Based Deterministic Hardware for At-Speed BIST*, Proc. VLSI Test Sym., pp. 201-207, 1993

[21] K.D. Wagner, C.K. Chin and E.J. McCluskey, *Pseudorandom Testing*, IEEE Trans. Computers, Vol. 36, No. 3, pp. 332-343, March 1987

[22] J.A. Waicukauski, E. Lindenbloom, E.B. Eichelberger and O.P. Forlenza, *A Method For Generating Weighted Random Patterns*, IBM J. Res. Develop., pp. 149-161, March 1989