Correspondence

Measures of the Effectiveness of Fault Signature Analysis

JAMES E. SMITH

Abstract—**A linear feedback shift register can be used to compress a serial stream of test result data. The compressed erroneous bit stream caused by a fault is said to form the "signature" of the fault. Since the bit stream is compressed, however, it is possible for an erroneous bit stream and the correct one to result in the same signature.**

In this correspondence, measures of the effectiveness of using linear feedback shift registers for detecting faults in logic networks are examined. After a brief discussion of the underlying theory of fault signature analysis, measures of effectiveness proposed by others are examined and are shown to be of questionable validity since they depend on an assumption of independent errors. To provide more accurate measures, two classes of dependent errors that are likely to occur in practice are considered. These are burst errors and errors with incorrect bits spaced at intervals equal to some power of 2 (because most digital systems have dimensions based on powers of 2). Means for determining the effectiveness of fault signature analysis at detecting these classes of errors are given.

Index Terms—**Data compression, dependent errors, fault detection, fault signature, linear feedback shift registers.**

I. Introduction

An economical way to deal with the large amount of data and operating speeds that are often required for testing digital systems is to compress test result data using simple compression algorithms. This involves first designing the system so that test data can be repeated, beginning with a special "start" signal and ending with a special "stop" signal. The test stimulus resulting in the repeatable stream should be carefully chosen to exercise the circuit under test. It is then possible to monitor key lines in the circuit, using probes [1] or some built-in mechanism [2], and to compress the data stream observed during the test. The compressed test result can then be compared with the one that is known to be correct in order to determine whether a fault is present in the system. This type of analysis can result in considerable reduction of test result storage, but the compression algorithm must be simple enough to be performed at high speed. For example, one could count the number of 1's in the stream or the number of 0 to 1 or 1 to 0 transitions ("transition count testing" [3]). Reference [4] contains a summary of count functions that have been proposed.

A data compression technique that is of interest in this paper is based on linear feedback shift registers (LFSR's) as in [1], [2], [5]. Fig. 1 shows such a shift register. Beginning with the register in an initial state (typically all 0's), serial input data are shifted into the register. This compresses the stream of inputs to the length of the LFSR and forms what is referred to as a "signature."

The use of LFSR data compression has a number of applications in digital systems. It can be used in a production test stand or built into a system for periodic testing during operation [2]. Because it leads to accurate fault diagnosis requiring relatively simple diagnostic equipment, fault signature analysis is particularly attractive for field testing of microprocessor-based systems. For this reason, examples used in this correspondence are drawn from microprocessor systems.

As with other data compression techniques, LFSR signature analysis allows some errors to go undetected. Hence, some circuit faults may go undetected. It is the purpose of this correspondence to evaluate and develop measures of the effectiveness of LFSR's at detecting faults. We emphasize that it is the detection of faults that is of primary concern. However, one cannot directly observe faults, only their effect: errors. Hence, measures are formulated in terms of errors detected. Nevertheless in order for a measure to be meaningful it must be based on the detection of errors that are likely to occur in response to network faults. Accordingly, measures are discussed in light of fault/error relationships.

Section II of the paper briefly presents the mathematics underlying signature analysis. Only the binary case is discussed, since this is of greatest practical interest; however, all the results can be easily generalized to include multivalued systems. Section III discusses measures of LFSR effectiveness proposed by others. Section IV develops new measures based on errors that are deemed likely to occur in response to faults, and the correspondence concludes with Section V.

II. Signature Analysis as Polynomial Division

The mathematics upon which fault signature analysis is based is essentially the same as the basis for algebraic coding theory. A more detailed discussion of the material in this section can be found in nearly any coding theory textbook, for example, [8].

Binary vectors can be represented as polynomials with binary coefficients. For example, 1, 0, 0, 1, 1 can be represented as $X^4 + x + 1$. One can also perform arithmetic on the polynomials, with coefficient addition and multiplication beyong one modulo 2. Hence, these are no carries during addition, and when binary polynomials are added, multiplied, or divided binary polynomials result.

In this paper, we deal primarily with polynomial division implemented with LFSR's. LFSR's have two basic building blocks: storage devices (for example *D*-flipflops) and mod 2 adder/subtracters, i.e., exclusive-OR gates. Storage devices are drawn as squares, and exclusive-OR gates are represented with the symbol ⊕.

Fig. 1(a) shows an LFSR that divides by $x^5 + x^4 + x^2 + 1$. The storage elements are set to 0 initially. The dividend is shifted in serially, highest degree coefficient first. After the entire dividend has been shifted in, the entire quotient has been shifted out and the remainder is held in the shift register. Fig. 1(b) shows a shift register computation of $x^7 + x^6 + x^5 + x^4 + x^2 + 1/x^5 + x^4 + x^2 + 1$. An alternate divider circuit for the divisor $x^5 + x^4 + x^2 + 1$ appears in Fig. 2(a). The same division as above is shown in Fig. 2(b). We observe that the quotient produced by the alternate divider is the same, but the content of the register after the division is not the remainder as is the case with the divider shown in Fig. 1. That is, the input/output behavior is the same, but the internal states are different.

An input data stream of length k can be represented as a degree

Manuscript received August 6, 1979; revised January 30, 1980. This work was supported by the National Science Foundation under Grant ENG78-05778.

The author is with the University of Wisconsin-Madison, Madison, WI 53706.

0018-9340/80/0600-0510$00.75

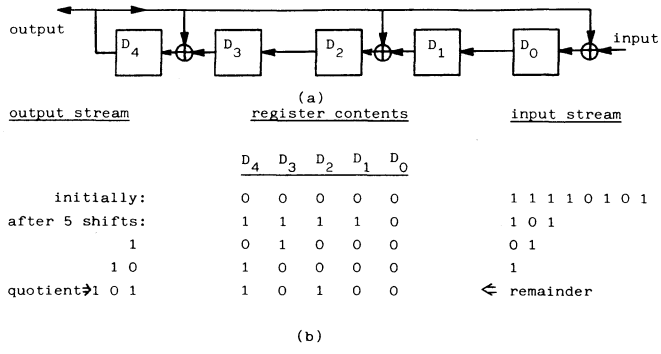

Fig. 1. (a) An LFSR for dividing by $x^5 + x^4 + x^2 + 1$. (b) The division of $x^7 + x^6 + x^5 + x^4 + x^2 + 1$ by $x^5 + x^4 + x^2 + 1$.

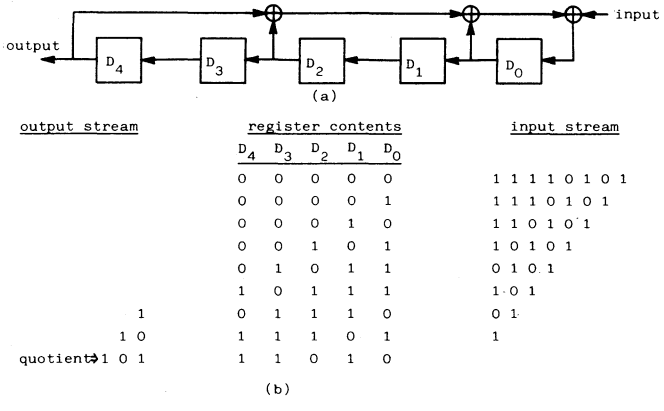

Fig. 2. (a) An alternate LFSR for dividing by $x^5 + x^4 + x^2 + 1$. (b) The division of $x^7 + x^6 + x^5 + x^4 + x^2 + 1$ by $x^5 + x^4 + x^2 + 1$. *Note:* The final contents are *not* the remainder.

$k - 1$ polynomial $m(x)$. We use a divider similar to Fig. 1 to compress $m(x)$ into a signature. This is done by shifting $m(x)$ into the divider and taking the remainder as the signature; the quotient is ignored. Let $p(x)$ be the degree r polynomial forming the divisor, and let the remainder (signature) be $s(x)$, a polynomial of degree less than r. Then $s(x)$ is related to $p(x)$ and $m(x)$ in the following way:

$$m(x) + q(x) \cdot p(x) + s(x).$$

We can also express an error pattern as a polynomial $e(x)$ so that each nonzero coefficient represents an error in the corresponding bit position. For example, if 10111 is the correct data stream and an erroneous one is 11101, then we let $m(x) = x^4 + x^2 + x + 1$ represent the correct stream and $m'(x) = x^4 + x^3 + x^2 + 1$ represent the erroneous one. The error pattern is represented by $e(x) = x^3 + x$ so that $m'(x) = m(x) + e(x)$. An undetectable error is one which satisfies

$$m(x) = e(x) = q'(x) \cdot p(x) + s(x);$$

that is, $m(x)$ and $m(x) + e(x)$ have the signature.

An alternate LFSR for generating a signature is a divider of the type shown in Fig. 2. This implementation may be preferred because it does not require placing exclusive-OR gates between shift register stages; this simplifies MSI realizations. Since the final contents of the divider of Fig. 2 are not the remainder, the signature differs from that used by the LFSR of Fig. 1. Nevertheless, the two signature generators do have an important property in common.

Theorem 1: Let $s(x)$ be the signature generated for input $m(x)$ using the polynomial $p(x)$ as a divisor in either LFSR shown in Figs. 1 and 2. For an error polynomial $e(x)$, $m(x)$ and $m(x) + e(x)$ have the same signature $s(x)$ if and only if $e(x)$ is a multiple of $p(x)$.

Theorem 1 as applied to an LFSR of the type shown in Fig. 1 is a fundamental result from algebraic coding theory. A demonstration than an LFSR of the type shown in Fig. 2 has the stated property follows from work appearing in [10].

III. Previously Proposed Measures of Effectiveness

As has been indicated, some circuit failures escape detection when signature analysis is used because some erroneous sequences are compressed to the same signature as the correct one. Consequently, one would like measures of the effectiveness of fault signature analysis at detecting faults. A primary use of such measures is to assure users of signature analysis that a satisfactorily high percentage of faults is detected. In addition, such measures can be used to compare LFSR signature analysis with other compression techniques and to compare the relative effectiveness of different polynomials at detecting faults.

We begin by considering measures proposed in [1]. Although the results are the same as those in [1], the proofs are simpler since they use Theorem 1 as a starting point.

Theorem 2: For a data stream of length k, if all possible error patterns are equally likely, then the probability that a length r signature generator will not detect an error is

$$\frac{2^{k-r} - 1}{2^k - 1}.$$

Proof: If a length k data stream is being compressed, it can be represented by a degree $k - 1$ polynomial. An error can be represented by an error polynomial of degree $k - 1$ or less. There are $2^k - 1$ possible error polynomials because there are k coefficients in a degree $k - 1$ polynomial and each coefficient can be 0 or 1. The 1 is subtracted from 2^k to eliminate $e(x) = 0$.

If $p(x)$ is degree r, then it has $2^{k-r} - 1$ nonzero multiples of degree less than k. Each of these represents an undetectable error, and all undetectable errors are represented by one of the multiples of $p(x)$ (from Theorem 1). Hence, $2^{k-r} - 1$ of the $2^k - 1$ errors are undetectable, and the theorem follows. □

As $k \to \infty$, the probability of Theorem 2 approaches 2^{-r}. The degree r of the polynomial chosen in [1] is 16, and the probability from Theorem 2 becomes 0.000015. This low probability is often quoted to point out the effectiveness of LFSR signature analysis. However, we observe that there is no qualification in Theorem 2 as to what degree r polynomial is used. That is, it holds regardless of the polynomial chosen. This includes the polynomial x^r that has a degenerate LFSR with no feedback; the signature is just the last r bits of the data stream. In fact, further examination reveals that we could also use the first r bits, truncate the test sequence to length r, and still have probability 2^{-r} of an undetected error. Consequently, if all error patterns are indeed equally likely, signature analysis is not called for since long test sequences are not needed.

A second measure used in [1] is based on the detection of single bit errors.

Theorem 3: An LFSR based on any polynomial with two or more nonzero coefficients detects all single bit errors.

Proof: Let $p(x)$ be any polynomial with two or more nonzero coefficients. Then all nonzero multiples of $p(x)$ must have at least two nonzero coefficients. Hence, any error with only one nonzero coefficient cannot be a multiple of $p(x)$ and must be detectable from Theorem 1. □

The polynomial $x + 1$ fits the constraint of having 2 or more nonzero coefficients, so a simple LFSR consisting of only one storage element and one exclusive-OR gate detects all single bit errors.

Two measures from [1] have been discussed and it has been shown here that very simple shift registers are just as effective as the more complex one given in [1]. Nevertheless, one tends to have an intuitive feeling that the more complex register should detect more faults. The reason for this apparent contradiction can be traced to the difference between types of errors that occur in digital systems due to faults and the error assumptions used as a basis for the measures.

If one assumes all error patterns are equally likely, then as $k \to \infty$

this becomes equivalent to assuming each individual bit has probability $\frac{1}{2}$ of being in error independent of the others. Concern with single errors also follows from an assumption of independent errors although in a more indirect way. In many communication channels, assuming independent errors makes single errors the most likely type of error.

For many communication applications, an assumption of independent errors is quite valid; hence, much of classical coding theory uses this assumption. However, as has been pointed out ([6, p. 6], for example), a logic network with a fault does not behave in the same way as a classical communication channel. Quite simply, errors due to a logic fault are not independent.

Before continuing to the next section, where dependent errors are discussed, we should first mention measures of effectiveness proposed in [5]. In [5], polynomials of the form $x^r + 1$ are suggested, and measures are developed not only for the detection of errors, but for locating them. These follow from the observation that different faults are likely to result in different error patterns. If the errors result in different signatures then the faults can be distinguished. The measures are intended to indicate the likelihood of getting different signatures for different errors.

All of the results derived in [5] are based on an assumption of independent errors. Since such an assumption appears to be of questionable validity, we do not discuss them in detail. Briefly, it appears that all polynomials of degree r with a nonzero coefficient of x^0 will do equally well using the measures given in [5], although this has not been proved.

IV. New Measures Based on Dependent Errors

In this section we develop measures of the effectiveness of fault signature analysis based on dependent error assumptions. To do this, we first develop error models by considering the behavior of faulty logic networks under test.

A. Burst Errors

Consider a microcomputer system consisting of a microprocessor, some ROM's, some RAM's, and some I/O interfaces. Such a system is typical of one where signature analysis might be used. A common way of testing such a system (see [7], for example) is to perform the following steps in sequence:

1) Test the "kernel"; the kernel consists of that part of the system which is required for the execution of any instruction, i.e., the fetching of instructions from memory in correct sequence.

2) Use the kernel to test each microprocessor instruction and the registers.

3) Use the microprocessor to test ROM1, ROM2, $\cdots$, ROMi.

4) Use the microprocessor to test RAM1, RAM2, $\cdots$, RAMj.

5) Use the microprocessor and a RAM to test the I/O interfaces.

Since many parts of the system are only exercised during one step of the sequence (or during only a portion of one step), many faults result in errors where erroneous bits are restricted to occur within some neighborhood. That is, a burst error occurs while the faulty portion of the system is being exercised.

Definition 1: A (n, d)—*burst error* is one where all erroneous bits are within n consecutive bit positions and at most d bits are in error.

We now turn to coding theory to determine measures for the detectability of (n, d)—burst errors. In the following theorem a polynomial $p(x)$ *generates* a code of length n if all multiples of $p(x)$ of degree less than n represent code words.

Theorem 4: Let $p(x)$ be a degree r polynomial that generates a d-error detecting (distance $d + 1$) code of length n. Then LFSR signature analysis using $p(x)$ detects all (n, d)—burst errors regardless of the length of the input stream.

Proof: Any polynomial $e(x)$ with fewer than $d + 1$ nonzero coefficients and degree less than n is detected since $p(x)$ generates a d-error detecting code. Hence, $e(x)$ does not contain $p(x)$ as a factor. An $(n, d$-burst is of the form $x^i e(x)$ where $e(x)$ is of degree less than n with fewer than $d + 1$ nonzero coefficients. If $p(x) = x^j$, the theorem holds trivially because $d = 0$. Otherwise if $p(x)$ is not a factor of $e(x)$, it is not a factor of $x^i e(x)$. The theorem then follows. □

For comparison here and in the next section, we choose four specific polynomials:

$P_A(x) = x^{16} + x^{12} + x^9 + x^7 + 1$; used in [1].

$P_B(x) = x^{16} + x^{15} + x^{12} + x^7 + x^6 + x^5 + x^4 + 1$;
a generator for a length 31, 7-error detecting BCH code.

$P_C(x) = x^{16} + 1$; proposed in [5].

$P_D(x) = x^{16}$; the polynomial requiring no feedback.

As was pointed out earlier, all four polynomials perform equally well when all errors are equally likely, and all but $P_D(x)$ detect all single errors.

$P_A(x)$ is a primitive polynomial [8]. A primitive polynomial of degree r generates a 2-error detecting Hamming code of length $2^r - 1$. Hence, all $(2^{16} - 1, 2)$-burst errors are detected by $P_A(x)$.

On the other hand, if one uses $P_B(x)$, then all (31, 7)-burst errors are detected. In the same neighborhood of 31, $P_A(x)$ can detect at best all (31, 4)-burst errors since the polynomial itself has 5 nonzero coefficients. In the larger neighborhood $2^{16} - 1$, $P_B(x)$ cannot detect all 2 errors. There appears to be a tradeoff between n and d, and we have indicated two points in the tradeoff curve. A related tradeoff is well known in coding theory where the tradeoff is between the code distance and the number of check bits. The two polynomials just discussed optimize n for the given value of d, however, there are some polynomials that detect all (n, d)-bursts for neither high n nor d. For example, $P_C(x)$ fails to detect the (17, 2) burst corresponding to $e(x) = P_C(x)$. $P_D(x)$ generates a distance 1 code so it is very ineffective against bursts. One can study the (n, d)-burst detecting characteristics for a given polynomial more completely by referring to "shortened cyclic codes" [8]. However, we choose to end this discussion with a final theorem that holds when the length of the burst neighborhood is reduced to r, the degree of $p(x)$.

Theorem 5: If $p(x)$ is degree r and the coefficient of x^0 is 1, then all (r, r)-bursts are detected with signature analysis using $p(x)$.

Proof: Any nonzero multiple of $p(x)$ must have two nonzero coefficients appearing farther apart than r. A more complete proof of this well-known theorem as it applies to coding theory can be found in [8]. □

B. Errors Due to Repeated-Use Faults

The second type of dependent error we consider are those where all the erroneous bits are in a regular pattern in positions, $i, i + bn_1$, $i + bn_2, \cdots$, where b is a power of 2, typically the byte or word length. In terms of polynomials, these are errors of the form $e(x) = x^i E(x^b)$ $0 \le i \le b - 1$. For example, if $b = 4$ then $x^1((x^4)^3 + (x^4)^1 + (x^4)^0)$ $= x^{13} + x^5 + x$ is of this type.

For an example of a fault that results in such an error, refer to Fig. 3 where 4-bit bytes of data are being converted to serial form. Such a conversion is common in microprocessor systems where pin counts must be kept low and speed is not critical. The x_2 input of the multiplexer is stuck-at-1. The 4 bytes shown at the left of the figure are to be transmitted, and the correct and erroneous outputs are shown at the right. In this case, $b = 4$ and the error polynomial is $x^{13} + x^5 + x$.

Such errors were first pointed out in [9] and are the result of "repeated-use faults." According to Avizienis, "this error pattern occurs in a byte-organized computer during transfer, complementation, and addition; it will also be observed in parallel multiplication and division which employ b-bit shifts." Since short word length microprocessors have many characteristics of the byte-organized computers of which Avizienis is speaking, these errors also appear to have relevance in microcomputer systems. Errors with erroneous bits separated by powers of 2 are also mentioned in [1] although no measures are given for their detection.

We now determine the proportion of these errors that are detected by $p(x)$; or equivalently, the probability of detecting such an error if they are equally likely. We assume $p(x)$ has a nonzero coefficient

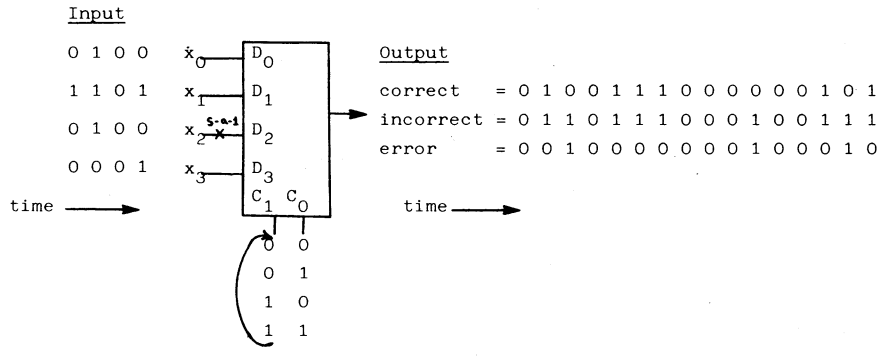

Fig. 3. The error behavior of a multiplexer with an input fault.

of x^0 for convenience and because it is now apparent that these are the most useful polynomials in practice. Some results adapted from the theory of Galois field are useful. As mentioned earlier we only consider the binary case.

Definition 2: A polynomial is irreducible if it cannot be expressed as the product of two polynomials of degree >1.

Theorem 6: Any polynomial $p(x)$ has a unique factorization:

$$p(x) = p_1(x)^{m_1} \cdot p_2(x)^{m_2} \cdot \cdots \cdot p_t(x)^{m_t}$$

where the $p_i(x)$ are irreducible.

This is analogous to the unique factorization of a positive integer into primes.

Theorem 7: $E(x^b) = E(x)^b$ when modulo 2 polynomial arithmetic is used and b is a power of 2.

We first determine the number of errors of the form $x^iE(x^b)$, $0 \le i < b$, that are multiples of $p(x)$.

Because the $p(x)$ that we are considering have no factor x^i, $i > 0$, one can show rather easily that $x^iE(x^b)$ is a multiple of $p(x)$ if and only if $E(x^b)$ is a multiple of $p(x)$. Consequently, we determine the number of errors $E(x^b)$ that are multiples of $p(x)$, and finding the number of $x^iE(x^b)$, $0 \le i < b$ is then straightforward.

First, factor $p(x)$ into its irreducible factors:

$$p(x) = p_1(x)^{m_1} \cdot p_2(x)^{m_2} \cdot \cdots \cdot p_t(x)^{m_t}.$$

Due to Theorem 7, we need to consider polynomials $E(x)^b$ that are multiples of $p(x)$. Each of these must contain each of the $p_i(x)$ as a factor some multiple of b times. Let $l_i = \lceil m_i/b \rceil$. bl_i is the minimum number of times each $p_i(x)$ must appear as a factor of $E(x)^b$. Or,

$$p_1(x)^{bl_1} \cdot p_2(x)^{bl_2} \cdot \cdots \cdot p_t(x)^{bl_t} = P(x)$$

must be a factor of $E(x)^b$. If the degree of $p_i(x)$ is d_i then the degree of $P(x)$ is $\Sigma_{i=1}^{t} bl_id_i = d$. Now, $E(x)^b = g(x) \cdot P(x)$ and $g(x)$ must be representable as $h(x)^b$. Our problem is then reduced to finding the number of possible $h(x)$. If a degree $k-1$ polynomial is being compressed, i.e., the degree of $E(x)^b < k$ then the degree of $g(x) < k - d$ and the degree of $h(x) \le \lfloor (k-d-1)/b \rfloor$. There are $2^{\lfloor (k-d-1)/b \rfloor + 1} - 1$ such nonzero polynomials, and this is the number of multiples of $p(x)$ that are of the form $E(x)^b = E(x^b)$. The total number of nonzero polynomials $E(x^b)$ is $2^{\lfloor (k-1)/b \rfloor + 1} - 1$.

By similar reasoning, the total number of nonzero polynomials of the form $x^iE(x^b)$ is $2^{\lfloor (k-1-i)/b \rfloor + 1} - 1$, and the number of nonzero multiples of $p(x)$ of the form $x^iE(x^b)$ is $2^{\lfloor (k-d-1-i)/b \rfloor + 1} - 1$.

This discussion gives us the following theorem. Recall that

- b byte length (errors are separated by multiples of b),
- k length of data sequence being compressed,
- d degree of $P(x)$ $\Sigma_{i=1}^{t} bl_id_i$.

Theorem 8: If the errors $x^iE(x^b)$ are equally likely, the probability that a polynomial $p(x)$ will not detect such an error in a length k data stream is

$$\frac{\sum_{i=0}^{b-1} 2^{\lfloor (k-d-1-i)/b \rfloor + 1} - 1}{\sum_{i=0}^{b-1} 2^{\lfloor (k-1-i)/b \rfloor + 1} - 1}.$$

As $k \to \infty$, this probability becomes $2^{-d/b}$. Furthermore, bit errors remain dependent, and the choice of a polynomial can lead to a significant difference in the detection probability. This is due to the dependence of d on the factorization of the polynomial used.

Corollary 1: If each of the irreducible factors of $p(x)$ appears only once, as $k \to \infty$, the probability of not detecting an error of the form $x^iE(x^b)$ is minimized, and the probability is 2^{-r}.

Proof: To minimize the probability of an undetected error, we need to maximize d:

$$d = \sum_{i=1}^{t} \left\lceil \frac{m_i}{b} \right\rceil d_i.$$

Furthermore, $\Sigma_{i=1}^{t} m_id_i = r$ which we assume is fixed. $\lceil m_i/b \rceil \le m_i$ so

$$b \cdot \sum_{i=1}^{t} \left\lceil \frac{m_i}{b} \right\rceil d_i \le b \cdot \sum_{i=1}^{t} m_id_i.$$

$\lceil m_i/b \rceil = m_i$ when $m_i = 1$ (or in the degenerate case $b = 1$). Hence,

$$b \sum_{i=1}^{t} \left\lceil \frac{m_i}{b} \right\rceil d_i$$

is maximized when all the $m_i = 1$ and

$$d = b \sum_{i=1}^{t} d_i = b \cdot r.$$

Consequently, $2^{-d/b} = 2^{-b \cdot r/b} = 2^{-r}$. □

Corollary 2: If each of the irreducible factors of $p(x)$ appears an integral multiple of b times, then as $k \to \infty$, the probability of not detecting an error of the form $x^iE(x^b)$ is maximized, and the probability is $2^{-r/b}$.

Proof: $l_i = \lceil m_i/b \rceil$ so $bl_i \ge m_i$ and

$$d = \sum_{i=1}^{t} bl_id_i \ge \sum_{i=1}^{t} m_id_i = r.$$

When $m_i = bl_i$ then

$$d = \sum_{i=1}^{t} m_id_i = r.$$

Here, the probability of an undetected error is $2^{-d/b} = 2^{-r/b}$. □

We now use the above measures to compare the four polynomials given earlier. We let $b = 8$, as might be the case in a microprocessor system. Then $P_A(x)$ and $P_B(x)$ have the minimum probability of missing an error, $2^{-r} = 2^{-16} \simeq 0.000015$, because each has its irreducible factors appearing only once ($P_A(x)$ is itself irreducible). On the other hand, $P_C(x)$ and $P_D(x)$ have the maximum probability of missing an error, $2^{-r/b} = 2^{-8} \simeq 0.004$, because each has one factor appearing 16 times ($x + 1$ and x, respectively). If we let $b = 16$, then the performance of $P_A(x)$ and $P_B(x)$ is unaffected, but $P_C(x)$ and $P_D(x)$ only detect 0.5 of the errors. It is also interesting to note that while $x^{16} + 1$ maximizes missed errors for length 16 signatures, $x^{15} + 1$ minimizes them for length 15 signatures because each irreducible factor of $x^{15} + 1$ appears only once.

VI. Conclusions

Errors due to faults in digital systems are not independent, and any measure based on independent errors is of questionable value. Consequently, to determine the effectiveness of fault signature analysis (or any other compression technique) one approach is to characterize classes of dependent errors that are likely to occur and to determine the likelihood of their detection. This approach was used here.

An approach of this type, however, does not give a complete measure because all potential errors are not considered. On the other hand, measures of this type do tend to give one confidence in LFSR signature analysis, provided the error classes studied are realistic. They are also useful for pointing out weaknesses in certain configurations. For example, the polynomial $x^{16} + 1$ was shown to perform relatively poorly against the repeated-use errors, and should probably be discarded for that reason.

It is clear that some experimental research in this area is needed. For example, the (n, d)-burst detecting properties of the polynomials $P_A(x)$ and $P_B(x)$ are different, but without experimental data a choice of one over the other is difficult.

Another area where further research is needed is in error modeling. That is, one could formulate a model, possibly similar to models for intermittent faults, and calibrate the model (i.e., adjust parameters) with experimental data. Then, the model could be used to give a more complete measure of the effectiveness of test data compression techniques.

References

[1] R. A. Frohwerk, "Signature analysis: A new digital field service method," *Hewlett-Packard J.*, pp. 2–8, May 1977.

[2] N. Benowitz *et al.*, "An advanced fault isolation system for digital logic," *IEEE Trans. Comput.*, vol. C-24, pp. 489–497, May 1975.

[3] J. P. Hayes, "Transition count testing of combinational logic circuits," *IEEE Trans. Comput.*, vol. C-25, pp. 613–620, June 1976.

[4] H. Fujiwara and K. Kinoshita, "Testing logic circuits with compressed data," in *Proc. 8th Annu. Int. Conf. on Fault-Tolerant Comput.*, June 1978, pp. 108–113.

[5] R. David, "Feedback shift register testing," in *Proc. 8th Ann. Int. Conf. on Fault-Tolerant Comput.*, June 1978, pp. 103–107.

[6] D. A. Anderson, "Design of self-checking digital networks," Coordinated Sci. Lab. Rep. R-527, University of Illinois, Urbana, Sept. 1971.

[7] Hewlett-Packard Corp., *A Designers Guide to Signature Analysis*, Appl. 222, 1978.

[8] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA: MIT Press, 1972.

[9] A. Avizienis, "A study of the effectiveness of fault-detecting codes for binary arithmetic," Jet Propulsion Lab. Tech. Rep. 32-711, Pasadena, CA, Sept. 1965.

[10] J. E. Meggitt, "Error correcting codes and their implementation for data transmission systems," *IRE Trans. Inform. Theory*, vol. IT-7, pp. 234–244, Oct. 1961.

Minimal Detecting Transition Sequences: Application to Random Testing

RENÉ DAVID AND PASCALE THÉVENOD-FOSSE

***Abstract*—This paper presents the new notion of *minimal detecting transition sequence* (MDTS). A detectable fault f in a circuit C is detected by any MDTS in a set D^f called *detection set associated with* f. From a prescribed set of faults, we obtain a list of detection sets. This list of detection sets is calculated once for all, for a given circuit C. Once this list has been obtained for a circuit, it may be used either to generate a deterministic test sequence, or to calculate *random testing* lengths within various hypothesis (input vector probabilities).**

***Index Terms*—Detection set, minimal detecting transition sequence (MDTS), random testing, sequential circuit, transition sequence.**

Manuscript received July 30, 1979; revised January 17, 1980. This work was supported in part under Contracts IRIA 74.147 and IRIA 78.195 (SURF Project).

The authors are with the Laboratoire d'Automatique, Institut National Polytechnique de Grenoble, Grenoble, France.

I. Introduction

Several works on random testing of sequential circuits have been done [1]–[3]. This paper presents the new notion of *minimal detecting transition sequence* (MDTS). This notion has been developed to analyze random testing but it may also be used to generate a deterministic test sequence. Roughly speaking, *a detecting transition sequence* for a fault f *is a string of branches* (*transitions*) on the flow graph of the *fault-free machine*, which ensures the detection of the fault f whatever the initial state of the faulty machine may be. There is no hypothesis concerning the number of states of the faulty machine. The set of MDTS's associated with a fault f is called the *detection set* associated with f.

Once a list of detection sets, associated with a prescribed set of faults, has been obtained for a circuit (for example a *JK* flip-flop), it may be used in several ways [4]. Examples concerning *deterministic testing* and *random testing* are given. An asynchronous version has already been used [3], [5], [6] to analyze the testing of a circuit, such as a flip-flop, when it is included in a larger circuit. Properties 2–4 and Theorem 2 in this paper have been shown in [5].

II. Background

Let a Moore-type sequential machine $M(X, Q, Z, \delta, \omega)$, according to the classical notations, become $M^f(X, Q^f, Z, \delta^f, \omega^f)$ if a fault f is present. The sets of internal states $Q = \{q_1, \cdots, q_n\}$ and $Q^f = \{q_1^f, \cdots, q_m^f\}$ are such that $m \neq n$ or $m = n$. Let X_i and J denote an input vector and an input sequence, respectively. $\delta(q_i, J)$ denotes the state reached by applying J to q_i. Any Mealy-type machine may be transformed to an equivalent Moore-type machine (see Fig. 2). Accordingly, the use of Moore-type machines is not a restriction.

The application of an input test sequence will be called a "*testing experiment*." It is assumed that the fault-free circuit M can be initialized in a preset state $q_0 \in Q$ and that the preset sequence is applied before a testing experiment.

Definition 1: Let M^1 and M^2 be two sequential machines having the same set of inputs. A state q_i^1 of M^1 is said to be *i-compatible* (input-compatible) with a state q_j^2 of M^2 if M^1 may be in the state q_i^1, knowing that 1) at least one input vector has been applied to both machines and 2) M^2 is in the state q_j^2. □

Property 1: q_i^1 is i-compatible with q_j^2 iff $\exists X_m, q_k^1, q_l^2$: $\delta^1(q_k^1, X_m) = q_i^1$, $\delta^2(q_l^2, X_m) = q_j^2$. (Then the i-compatibility is reciprocal.)

Proof: It follows from Definition 1. □

The idea of this notion is that, *at any time* during a testing experiment: if the fault-free machine M should be in the state $q_i \in Q$, a faulty machine M^f cannot be in any state in Q^f. $C_i^f \subset Q^f$ denotes the set of states i-compatible with q_i. Let us consider, for example, the flow tables of M_1 and M_1^f. Fig. 1(b): M_1 in the state **a** $\Rightarrow$ the last input vector has been X_0. Fig. 2(c): the last input vector has been $X_0 \Rightarrow M_1^f$ in the state **b** or **c**. Hence $C_{\mathbf{a}}^f = \{\mathbf{b}, \mathbf{c}\}$. This is represented in Fig. 2(d): a point is associated with a pair of i-compatible states. Another information appears in Fig. 2(d): a cross is associated with a pair of states the outputs of which are different. If two states such as **b** in M_1 and **d** in M_1^f have the same output vector, they will be called *o-compatible* (output compatible). The table in Fig. 2(d) is called the *compatibility table* of M_1^f, relative to M_1.

Definition 2: A *transition sequence* T is *a string of successive transitions* in a machine M. It is defined by a state q_i and an input sequence J. We note $T = q_iJ$ (see Fig. 1(c)). □

Definition 3: The concatenation of two transition sequences $T_1 = q_1J_1$ and $T_2 = q_2J_2$ is defined, iff $\delta(q_1, J_1) = q_2$, by $T_1 \cdot T_2 = q_1J_1 \cdot q_2J_2 = q_1J_1J_2$. □

For example, [see Fig. 1(c)] $\mathbf{a}X_1X_0 \cdot \mathbf{b}X_1 = \mathbf{a}X_1X_0X_1$. The input

0018-9340/80/0600-0514$00.75