

Error Localization in Test Outputs: a Generalized Analysis of Signature Compression

Serge Demidenko*, Vincenzo Piuri**, Alexander Ivaniukovich*

* Institute of Engineering Cybernetics
Belarusian Academy of Sciences
Surganov Street 6, Minsk 220012, Belarus, CIS

**Department of Electronics and Information
Politecnico di Milano
Piazza Leonardo Da Vinci 32, Milano 20133, Italy

Abstract

Signature compression is widely used to reduce the volume of information generated by testing. Localization of the faulty component in the system under test, starting from the signature analysis, may be a complex problem related to identification of the erroneous bits in the sequence being compressed and to the error model of the system. A first technique for low-multiplicity errors is based on error effects' superposition. A novel general solution is derived for any-multiplicity errors from the analytical description of signature compression.

1. Introduction

Signature analysis is a widely-known testing methodology for digital circuits. It allows substantial reduction of the amount of diagnostic data by compressing the output sequence of the circuit under test into a short word (the signature) [1]. Compression is performed by the signature registers on each datum: Single-Channel Signature Registers and Multi-Channel Signature Registers are used for an input datum composed by one or several bits, respectively. In the first case, a single-bit input stream enters into the register, while in the second case a multiple-bits input stream is considered.

An error is detected whenever the actual signature S (produced by compressing the test outputs during testing) differs from the reference value S^* (corresponding to the fault-free circuit). In this case, it is desirable to identify the position of the erroneous bits within the stream being compressed that have generated the erroneous signature. From such information and from the error model, it is possible to provide localization of the faulty component within the circuit under test. While the signature computed from a sequence of data is unique, the same signature may be associated to several data sequences; thus, aliasing may occur in error localization.

In the literature, several techniques were presented to localize single erroneous bits in the sequence being compressed by using the fault signature S' (i.e., the result of the bit-wise addition of S and S^*) [2-4]. In [2], it was

shown that there is a one-to-one correspondence between the position of a single-bit error and the error signature exists if the length L of the sequence being compressed is less than $2^n - 1$, where n is the degree of the signature register characteristic polynomial (i.e., the length of the register). A look-up table may be used for localization.

In [3, 4] the signature register is used as a pseudo random generator (i.e., the signature register operates having null inputs). In [3], signature compression is performed by means of a signature register with characteristic polynomial $G(x)$. The register for error localization has a feed-back structure which is described by the reciprocal polynomial of $G(x)$; the fault signature S' is loaded in the register and a pseudo-random generation process is executed until a given binary combination ($100...00$) appears in the register. The position of the erroneous bit is $m-q$, where m is the length of the output sequence of the circuit under test, while q is the number of time steps required to produce the combination.

To avoid the use of two different polynomials - the signature register characteristic polynomial and its reciprocal polynomial - in [4] specific binary combinations are used to initialize the signature register before starting the signature compression. After compression has been completed, pseudo-random generation is executed in the register itself until the given binary combination ($100...00$) appears. The number of time steps spent in pseudo-random generation is to the position of the erroneous bit in the sequence.

Occurrence of single-bit errors is often highly probable rather than occurrence of multi-bit errors. However, in a number of mission-critical applications or in very large and complex integrated systems, the capability of localizing also multiple-bit errors may be dramatically important to guarantee a good testing quality, a high credibility of the system, a proper behavior of the components, and a high availability. No practical solutions have been proposed by now for error localization in the case of MCSR compression.

In this paper we present some approaches to solve the problem of error localization in the case both of single

errors and of multiple ones, for SCSRs and for MCSRs. A simple solution localizes errors having rather low multiplicity with SCSR (section 2): any multiple-bit error signature is represented by bit-wise modulo-2 addition of a number of single-bit signatures; each of these single-bit localization sub-problems is solved separately by using any known algorithm.

In section 3, we present the general analytical solution for localizing all possible errors in the sequence being compressed for the case both of SCSR and of MCSR compressions (by now, no other general approach is known for the case of MCSR). We derive analytically expressions that describe the mapping between the value of each signature bit and the value of bits of the sequence being compressed. Our solution is completely general: it holds for any capacity of the signature register, for any feed-back structure of the register, for any length of the sequence being compressed, and for any number of bits of datum belonging to the sequence being compressed.

2. A superposition-based algorithm for localization of low-multiplicity errors

Let $P = \{p_1, p_2, p_3, \dots, p_m\}$ the output sequence generated by the circuit under test, where m is the length of the sequence ($m=1,2,3,\dots$). It is represented by the output polynomial $P(x)$ of $(m-1)$ -th degree, whose coefficients are the bits of the sequence itself [5]. The signature polynomial $S(x)$ is the remainder of division of the polynomial $P(x)$ by the characteristic polynomial $G(x)$ over the field $GF(2)$ [6]. Let $P^*(x)$ be the output polynomial which is associated to the output sequence generated by the fault-free circuit. The corresponding signature $S^*(x)$ is assumed as the reference value. $P(x)$ and $S(x)$ are the actual output polynomial and the actual signature, respectively: they are produced by testing the circuit, possibly in the presence of faults. Whenever the actual signature $S(x)$ does not match the reference $S^*(x)$, an error is detected: diagnosis means to locate the erroneous bits of $P(x)$, i.e., the positions in which $P(x)$ differs from $P^*(x)$.

Let $P'(x)$ be the error polynomial. Each non-null coefficient in $P'(x)$ represents one erroneous bit in the output sequence in the correspondent position. The error polynomial is the bit-wise modulo-2 addition of the actual and the reference output polynomials: $P'(x) = P(x) \oplus P^*(x)$. Since addition over $GF(2)$ is a linear operation, the signature of the sum of two sequences is equal to the sum of their signatures [5]: the error signature polynomial $S'(x)$ is thus $S'(x) = S(x) \oplus S^*(x)$.

Generally, the error polynomial $P'(x)$ has k non-null coefficients: the corresponding error sequence contains k non-null bits. Due to the linearity of addition in $GF(2)$, we apply the superposition property and we view the error

sequence as a linear combination of k different sequences (the *partial error sequences*) in $GF(2)$: each of them has only one non-null bit, i.e., the error sequence is represented by the bit-wise modulo-2 addition of k partial error sequences which represent k different single-bit errors. In polynomial notation, it is:

$$P'(x) = P'_1(x) \oplus P'_2(x) \oplus \dots \oplus P'_k(x)$$

$k \in \{1, 2, \dots, m\}$; for the error signature, we have [7]:

$$S'(x) = S'_1(x) \oplus S'_2(x) \oplus \dots \oplus S'_k(x)$$

where $S'_i(x)$ is the *partial error signature polynomial* which corresponds to *partial error polynomial* $P'_i(x)$ of the i -th single-bit error, for $i=1,2,3,\dots,k$. Partitioning of the output error sequence into partial error sequences is unique; error signature polynomial, derived from the error sequence, can be uniquely partitioned into partial error signature polynomials (each of them is computed from one of partial error sequences).

Conversely, let us consider a given error signature polynomial $S'(x)$ and partition it into a set of partial error signature polynomials $S'_i(x)$ so that each partial error signature polynomial $S'_i(x)$ derives from a partial error polynomial $P'_i(x)$. It can be shown that such partitioning may be performed in several different ways, i.e., there is no unique correspondence between the error signature polynomial $S'(x)$ and the error polynomial $P'(x)$. The same error signature may thus be generated by different error distributions in the sequence being compressed.

From the above remarks, we derive the following general methodology for localizing multiple-bit errors:

1. Error signature is computed by bit-wise modulo-2 adding the actual signature and the reference one.
2. Error signature is represented as the sum of partial error signatures. The number of terms in such a summation is equal to the multiplicity of the occurred error: it can be known *a priori* or it may be presumed. Since partitioning is not unique, sets of partial error signatures identify the possible error locations.
3. The sets of partial error sequences are derived from the sets of partial error signatures obtained in step 2.
4. The set of possible error sequences of the given multiplicity is generated by bit-wise modulo-2 adding the partial error sequences.

Steps 1 and 4 are trivial boolean operations. Several algorithms have been proposed in the literature for single-bit error localization [2-4]: step 3 may be solved by using one of them. The main focus of our approach is on the implementation of the partitioning step.

Direct analysis and generation of all possible partitioning is an NP-complete problem; thus, it is not practical when the number of alternative combinations becomes high. A more efficient approach should consider the following problem-oriented, implementation-specific requirements.

One set of partial error signatures is generated and treated at a time to avoid massive storage requirements and to allow pipelined execution of the subsequent steps of the algorithm. To reduce the computational efforts, each set of partial error signatures must be generated only once, i.e., the partial signatures are taken into account without any regard to the ordering position of the signatures within the set. A set of partial error signatures should not contain any of the following cases: the null signature, or several identical signatures, or the given error signature.

2.1 Partitioning 2-bits errors into two single-bit errors

Let us consider the case of two-bits in the error signature different from the expected ones. The block diagram of the simplified algorithm for partitioning two-bit errors into pairs of partial single-error signatures is shown in fig. 1. The first step is generation of the complete list of all possible partial signatures as candidates for the pairs of single-bit error signatures.

The second critical step is selection of the pairs (A, B) of partial signatures from the list: for each partial signature A , the second partial signature B , that can be used to achieve the complete signature by superposition with the first one, is identified by adding (bit-wise modulo-2) the first partial signature and the complete signature.

To avoid multiple selection of the same pairs, partial signatures of each generated pair are removed from the list of candidates. To trace the use of partial signatures and to avoid multiple re-generation of the same pairs of partial signatures, we introduce the vector $M1$. Partial signatures are used to label the entries of $M1$ as they were memory addresses. Each element of $M1$ is the counter of the number of times that the corresponding partial signature can still be used to create pairs of partial signatures leading to the given complete error signature (for two-bit errors, the maximum value is 1 since each signature can be used only in one pair).

As example, we consider the case of the two-bit error signature $S'=101$ and $L=7$. Results generated in $M1$ at the different iterations are shown in table 1: signatures label $M1$ elements (the columns), while different states of the vector are given in rows. In operations from 1 to 5, $M1$ is created. The initial value of each element of $M1$ is set to 1, but the element corresponding to the error signature S' (see state 1); the error signature itself cannot be a member of any pair of partial signatures.

The first partial signature $A=001$ is selected from the vector by operations 6, 7, and 10. The second partial signature B is computed by bit-wise modulo-2 addition with the given signature S' ; it is $B=100$ (operation 11). The content of $M1$ is now given by state 2. The next allowed partial signature A is selected in $M1$: it is $A=010$

(operation 8, 9, 7, and 10). The second partial signature of the second pair is $B=111$ (operation 11). $M1$ is in state 3. The next value of the partial signature is $A=011$, while the second partial signature of the third pair is $B=110$ (see state 4). Since there are no allowed values for the first partial signature, the generation of the signatures' pairs ends. The generated pairs are given in table 2.

$M1$	001	010	011	100	101	110	111	Pair	A	B
state 1	1	1	1	1	0	1	1	1	001	100
state 2	0	1	1	0	0	1	1	2	010	111
state 3	0	0	1	0	0	1	0	3	011	110
state 4	0	0	0	0	0	0	0			

Table 1 - The states of the counters' vector $M1$

Table 2 - The pairs of partial signatures

2.2 Partitioning 3-bits errors into 3 single-bit errors

The algorithm presented in section 2.1 can be generalized to three-bits errors: it must identify triplets of partial signatures leading to S' . At first, it generates all unique pairs of the partial signatures A and B ; then, it computes the third partial signature C as the sum of A , B and S' . The block diagram is presented in fig. 2.

Each partial single-bit signature can be used more than once in different partitioning. The number P of allowed occurrences of each partial signature in the partitioning sets is derived by applying the combinatorial analysis [8]. For three single-bit error signatures, P is $2^{n-1} - 2$, where n is the length of the signature register. The counters' vector $M1$ holds the actual number of times that each partial signature can be used to generate new partitioning sets. To take into account all compatibility constraints, we introduce the vector $M2$ of forbidden signatures: it points out the partial signatures that cannot be used together with the firstly-selected partial signature. Each element of $M2$ is associated to a partial signature, as the elements of $M1$. For a given partial signature, an element of $M2$ is 0 if the corresponding partial signature cannot be used in the partitioning sets; otherwise, it is 1.

We consider the same example discussed in section 2.1. We create the list of candidate partial signatures (operations from 1 to 5); the elements of the vector $M1$ are set equal to $P=2$, while the element corresponding to the error signature S' is set equal to 0 (state 1 in table 3).

The first partial signature $A=001$ is selected (operations 6, 7, and 10). Then, we compute the intermediate sum $X=A \oplus S'=100$ (operation 11) and we generate the vector of the forbidden signatures $M2$ (operations from 12 to 19 - see state 1 in table 4). The second partial signature B of the first partitioning set is selected among those which are

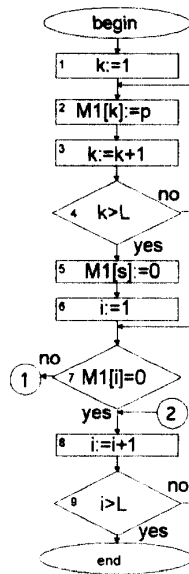
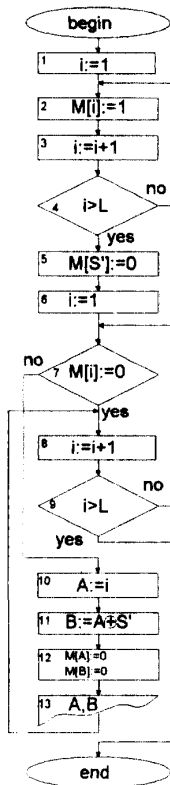


Fig. 1 - Partitioning two-bit error signatures

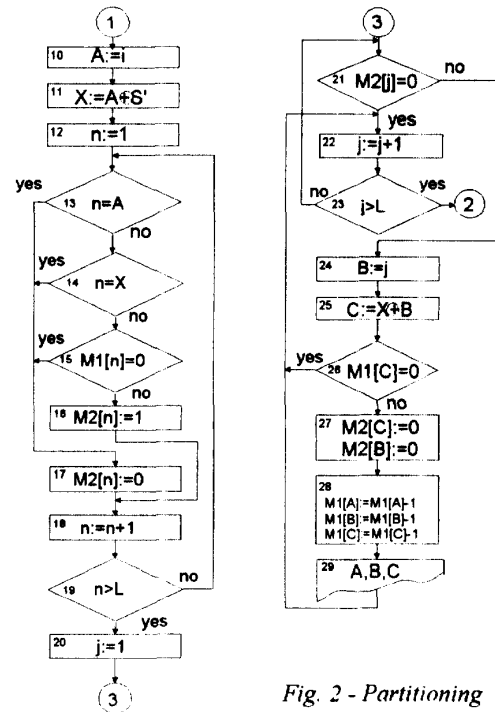


Fig. 2 - Partitioning three-bit error signatures

compatible with the first one according to the vector $M2$: it is $B=010$ (operations from 20 to 24). The third partial signature C of the first partitioning set is computed (operation 25) by $C=A \oplus B \oplus S' = B \oplus X = 110$; operation 26 checks compatibility with the other partial signatures. The vector $M2$ is then updated (operation 27) as it is shown in state 2 of table 4. The $M1$ vector is updated by decreasing the elements corresponding to the signatures A , B and C by one, since these signatures has been considered once (operation 28 - see state 2 - table 3). Generation of triplets of partial signatures is repeated updating the vectors $M1$ and $M2$ at each step, as discussed above; procedure stops when all elements of $M1$ become null. Results of each step are shown in rows from state 3 to state 7 in table 4 for the $M1$ vector and in the rows from state 3 to state 5 of table 3 for the $M1$ vector; the final results of this procedure are then summarized in table 5.

This algorithm was implemented and experimented to verify efficiency and effectiveness of our approach. The experimental version was implemented in Borland TurboPascal 5.5. It was proved highly efficient: for the case of $m=7$ and $S'=0000101$ (i.e., $L=127$ and $P=62$), computation of all signatures' triplets (2604 sets) required less than 0.3 seconds on an IBM PC AT/286 (12 MHz), excluding time for printing results.

M1	001	010	011	100	101	110	111
state 1	2	2	2	2	0	2	2
state 2	1	1	2	2	0	1	2
state 3	0	1	1	2	0	1	1
state 4	0	0	0	1	0	1	1
state 5	0	0	0	0	0	0	0

Table 3 - The states of the counters' vector $M1$

M2	001 010 011 100 101 110 111										
state 1	0	1	1	0	0	1	1				
state 2	0	0	1	0	0	0	1	Triplet A B C			
state 3	0	0	0	0	0	0	0				
state 4	0	0	1	1	0	1	0	1	001	010	110
state 5	0	0	0	0	0	1	0	2	001	011	111
state 6	0	0	0	0	0	1	1	3	010	011	100
state 7	0	0	0	0	0	0	0	4	100	110	111

Table 4 - States of forbidden signatures' vector $M2$

Table 5 - The triplets of partial signatures

The algorithm for partitioning the three-bits error signatures includes the algorithm for partitioning the two-bits error signatures. The only main difference is the use of the $M2$ vector of forbidden signatures.

This algorithm can be easily extended to the case of four-bits error signatures: it will include the algorithm for three-bits error signatures and it will introduce another vector $M3$ to take into account the forbidden signatures for the current first pair of partial signatures as the vector $M2$ stores the forbidden signatures.

The same approach can be extended also to deal with a higher value of the bit multiplicity in the error signature. By means of the general approach of partitioning and superposition, we can create an algorithm (and the corresponding software) for any given multiplicity, i.e., for any given number of single-bit error signatures. This implies that we can implement a computer-aided software generator which is automatically capable to create the partitioning procedure for any multiplicity which has been specified by the user.

3. The general methodology for error localization

At first, we review the basic operation performed by SCSR and by MCSR. For the case of the input into the first bit of the register, the SCSR operation is described by:

$$s_i(k+1) = s_i(k)\alpha_1 \oplus s_2(k)\alpha_2 \oplus s_3(k)\alpha_3 \oplus \dots \oplus s_n(k)\alpha_n \oplus b_i(k) \\ s_i(k+1) = s_{i-1}(k) \quad i = 2, 3, 4, \dots, n$$

where α_h are the coefficients of the characteristic polynomial ($\alpha_h \in \{0, 1\}$, $h \in \{1, 2, 3, \dots, n\}$), $s_h(k)$ is the state (value) of the h -th bit of signature register at the k -th time step ($k=0, 1, 2, 3, \dots$). $b_i(k)$ is the value of the bit belonging to sequence being compressed that enters at the k -th time step. By mathematical induction and iterative substitution, we derive for any j -th time step [9]:

$$s_i(j) = \left[\sum_{m=1}^n \oplus s_m(0) \sum_{t=m}^n \oplus \alpha_t C_{(j+m-t-1)} \right] \oplus \\ \oplus \left[\sum_{z=0}^{j-2} \oplus b_i(z) \sum_{f=1}^n \oplus \alpha_f C_{(j-f-z-1)} \oplus b_i(j-1) \right] \quad j \geq i \\ s_i(j) = s_{i-j}(0) \quad j < i, \quad i = 1, 2, \dots, n$$

where $C_{(w)}$ is the sum of products of the characteristic polynomial coefficients of the signature register. It is:

$$C_{(w)} = \begin{cases} 0 & \forall w < 0 \\ 1 & \forall w = 0 \\ \sum_{d=1}^{D(w)} \oplus \prod_{q=1}^{N(w,d)} \alpha_{p(d,q)} & \forall w > 0 \end{cases}$$

$$\text{where} \quad \sum_{q=1}^{N(w,d)} p(d,q) = w \quad \forall d \in \{1, 2, 3, \dots, D(w)\};$$

$$p(d,q) \in \{1, 2, 3, \dots, n\}; \left\{ p(d,q) \mid q = \overline{1, N(w,d)} \right\} \neq \left\{ p(d,q) \mid q = \overline{1, N(w,h)} \right\}$$

$\forall g \neq h$; and $g, h \in \{1, 2, 3, \dots, D(w)\}$. $D(w)$ equals the number of all of composition of the number w , while $N(w,d)$ is the number of summation terms of the d -th variant of composition.

The first part of equation (1) is dependent on the initial state of the register only, i.e., it describes a pseudo-random generator; the second part is dependent on the values of the elements of the sequence being compressed, i.e., it corresponds to a SCSR having null initial state.

In the same way, MCSRs are analytically described by:

$$s_i(k+1) = s_i(k)\alpha_1 \oplus s_2(k)\alpha_2 \oplus s_3(k)\alpha_3 \oplus \dots \oplus s_n(k)\alpha_n \oplus b_i(k) \\ s_i(k+1) = s_{i-1}(k) \oplus b_i(k) \quad i = 2, 3, 4, \dots, n$$

By mathematical induction and iterative substitution, it is:

$$s_i(j) = \left[\sum_{m=1}^n \oplus s_m(0) \sum_{t=m}^n \oplus \alpha_t C_{(j+m-t-1)} \right] \oplus \\ \oplus \left[\sum_{z=0}^{j-2} \oplus \sum_{t=1}^n \oplus b_t(z) \sum_{f=t}^n \oplus \alpha_f C_{(j+t-f-z-1)} \oplus \sum_{r=1}^i \oplus b_r(j-i+r-1) \right] \quad j \geq i$$

$$s_i(j) = [s_{i-j}(0)] \oplus \left[\sum_{r=1}^i \oplus b_r(j-i+r-1) \right] \quad j < i, \quad i = 1, 2, \dots, n$$

These equations have two parts: the first one describes a pseudo random sequence generator, the second one describes the multi-channel signature register having zeroed initial state.

From the above descriptions we can derive the theoretical foundations of our general localization methodology. Consider first the case of SCSR; the error signature $S'(x) = S(x) \oplus S^*(x)$ is given (for the practical case of $j > i$) by:

$$s'_1(j) = \sum_{z=0}^{j-2} \oplus b'_1(z) \sum_{f=1}^n \oplus \alpha_f C_{(j-f-z-1)} \oplus b'_1(j-1) \\ s'_2(j) = \sum_{z=0}^{j-2} \oplus b'_1(z) \sum_{f=1}^n \oplus \alpha_f C_{(j-f-z-2)} \oplus b'_1(j-2) \\ \dots \\ s'_n(j) = \sum_{z=0}^{j-2} \oplus b'_1(z) \sum_{f=1}^n \oplus \alpha_f C_{(j-f-z-n)} \oplus b'_1(j-n)$$

It is rearranged in the general canonical form:

$$a_{11}b'_1(0) \oplus a_{12}b'_1(1) \oplus a_{13}b'_1(2) \oplus \dots \oplus a_{1j}b'_1(j-1) = s'_1(j) \\ a_{21}b'_1(0) \oplus a_{22}b'_1(1) \oplus a_{23}b'_1(2) \oplus \dots \oplus a_{2j}b'_1(j-1) = s'_2(j) \\ \dots \\ a_{n1}b'_1(0) \oplus a_{n2}b'_1(1) \oplus a_{n3}b'_1(2) \oplus \dots \oplus a_{nj}b'_1(j-1) = s'_n(j)$$

where the coefficients a_{ve} ($v = \overline{1, n}$, $e = \overline{1, j}$) depend only on the coefficients $\{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n\}$.

To identify the erroneous bits we need to solve this system of linear Diophantine equations over the field $GF(2)$, with respect to the set $\{b'_1(0), b'_1(1), b'_1(2), \dots, b'_1(j-1)\}$. Non-

null elements of the set correspond to erroneous bits in the sequence being compressed.

The same approach can be used for MCSRs. The system describing the error signature is:

$$s'_1(j) = \sum_{z=0}^{j-2} \oplus \sum_{t=1}^n \oplus b'_t(z) \sum_{f=t}^n \oplus \alpha_f C_{(j+t-f-z-2)} \oplus b'_1(j-1)$$

$$s'_2(j) = \sum_{z=0}^{j-2} \oplus \sum_{t=1}^n \oplus b'_t(z) \sum_{f=t}^n \oplus \alpha_f C_{(j+t-f-z-3)} \oplus \oplus b'_1(j-2) \oplus b'_2(j-1)$$

$$s'_n(j) = \sum_{z=0}^{j-2} \oplus \sum_{t=1}^n \oplus b'_t(z) \sum_{f=t}^n \oplus \alpha_f C_{(j+t-f-z-n-1)} \oplus \oplus \sum_{r=1}^n \oplus b'_r(j-n+r-1)$$

and its canonical form (composed by linear Diophantine equations) is:

$$a_{11}b'_1(0) \oplus a_{12}b'_1(1) \oplus a_{13}b'_1(2) \oplus \dots \oplus a_{1j}b'_1(j-1) \oplus \oplus a_{1(j+1)}b'_2(0) \oplus a_{1(j+2)}b'_2(1) \oplus a_{1(j+3)}b'_2(2) \oplus \dots \oplus a_{12j}b'_2(j-1) \oplus \oplus \dots \oplus a_{1(n-1)}b'_n(0) \oplus a_{1(n-1)}b'_n(1) \oplus a_{1(n-1)}b'_n(2) \oplus \dots \oplus a_{1ny}b'_n(j-1) = s'_1(j)$$

$$a_{n1}b'_1(0) \oplus a_{n2}b'_1(1) \oplus a_{n3}b'_1(2) \oplus \dots \oplus a_{nj}b'_1(j-1) \oplus \oplus a_{n(j+1)}b'_2(0) \oplus a_{n(j+2)}b'_2(1) \oplus a_{n(j+3)}b'_2(2) \oplus \dots \oplus a_{n2j}b'_2(j-1) \oplus \oplus \dots \oplus a_{n(n-1)}b'_n(0) \oplus a_{n(n-1)}b'_n(1) \oplus a_{n(n-1)}b'_n(2) \oplus \dots \oplus a_{nny}b'_n(j-1) = s'_n(j)$$

From these results, we derive the general technique for localizing erroneous bits in the output sequence of the circuit under test both for SCSR and for MCSR. The localization procedure can be summarized as follows:

1. the error signature is computed by adding (bit-wise modulo 2) the reference signature and the signature generated by testing;
2. a system of linear Diophantine equations is derived for given j and $\{\alpha_i\}$, $i=1, 2, 3, \dots, n$;
3. this system is solved in the canonical form; it generates all possible error patterns that are related to the error signature.

Performances of this technique are related to the specific choice of the characteristic polynomial of the signature register. Good results can be achieved for minimal polynomials of the type $X^m \oplus X^j \oplus 1$ [9].

The following example shows application to SCSR. Let be $n=4$, $\alpha_4 = \alpha_3 = 1$, $\alpha_2 = \alpha_1 = 0$, and $j=7$. Consider $S^*=1001$ and $S=0100$. Erroneous bits are identified by performing the following steps:

1. Error signature is $S' = S^* \oplus S = 1001 \oplus 0100 = 1101$.

2. After having replaced all known coefficients with their values, the system describing the signature is:

$$S_1(7) = \sum_{z=0}^5 \oplus b'_1(0) \sum_{f=1}^4 \oplus \alpha_f C_{(6-f-z)} \oplus b'_1(6) =$$

$$= b'_1(0) \oplus b'_1(2) \oplus b'_1(3) \oplus b'_1(6) = 1$$

$$S_2(7) = \sum_{z=0}^5 \oplus b'_1(0) \sum_{f=1}^4 \oplus \alpha_f C_{(5-f-z)} \oplus b'_1(5) =$$

$$= b'_1(1) \oplus b'_1(2) \oplus b'_1(5) = 1$$

$$S_3(7) = \sum_{z=0}^5 \oplus b'_1(0) \sum_{f=1}^4 \oplus \alpha_f C_{(4-f-z)} \oplus b'_1(4) =$$

$$= b'_1(0) \oplus b'_1(1) \oplus b'_1(4) = 0$$

$$S_4(7) = \sum_{z=0}^5 \oplus b'_1(0) \sum_{f=1}^4 \oplus \alpha_f C_{(3-f-z)} \oplus b'_1(3) =$$

$$= b'_1(0) \oplus b'_1(3) = 1$$

3. The final solutions of the canonical system are:

$$N \quad b'_1(0) \quad b'_1(1) \quad b'_1(2) \quad b'_1(3) \quad b'_1(4) \quad b'_1(5) \quad b'_1(6)$$

1	0	0	0	1	0	1	0
2	0	0	1	1	0	0	1
3	0	1	0	1	1	0	0
4	0	1	1	1	1	1	1
5	1	0	0	0	1	1	0
6	1	0	1	0	1	0	1
7	1	1	0	0	0	0	0
8	1	1	1	0	0	1	1

Acknowledgements

Authors thank the National Research Council of Italy for financial support and Politecnico di Milano for the Visiting Research Fellowship (Dec.92-Mar.93) granted to S. Demidenko.

References

1. Bardell P.H., McAnney W.H., and Savir J.: Built-in Test for VLSI. - John Wiley, 1987
2. Nikolaou P.P., and Lefas C.C.: On the recovery of Error Patterns from Signatures Obtained in Digital System Testing. - BIT (Dan), 1988, vol. 28, No 2, pp. 242-252
3. McAnney W.N., and Savir J.: There is Information in Faulty Signatures. - ITC 1987, pp. 630-636
4. Diachenko O.N., and Tarasenko A.N.: Modes of Diagnostic Information Separation from Signature. - Electronic Modelling, 1990, vol. 12, No 5, pp. 64-70 (in Russian)
5. Peterson W.W., and Weldon E.J.: Error Correcting Codes. - MIT Press, 1972
6. Yarnolik V.N., and Demidenko S.N.: Generation and Application of Pseudorandom Sequences for Random Testing. - John Wiley, 1988
7. Chan J.C., and Womack B.F.: Diagnostics Based on Faulty Signature. - ITC 1989, p.935
8. Riordan J.: An Introduction to Combinatorial Analysis. - Princeton University Press, 1980
9. Demidenko S.N., and Piuri V.: A Generalized Approach to Error Localisation by Using Fault Signature Information. - Politecnico di Milano, Dept. Electronics, Report 9-93, 1993