IEEE TC '88

(83)

A Data Compression Technique for Built-In Self-Test

SUDHAKAR M. REDDY, KEWAL K. SALUJA, AND
MARK G. KARPOVSKY

Abstract—Data compression is often used to reduce the complexity of test data in the area of fault diagnosis in digital systems. A data compression technique called self-testable and error-propagating space compression is proposed and analyzed. Faults in a realization of Exclusive-OR and Exclusive-NOR gates are analyzed and the use of these gates in the design of self-testing and error-propagating space compres-

Manuscript received July 31, 1985; revised April 15, 1986 and July 17, 1987. S. M. Reddy was supported in part by U.S. Army Research Contract DAAG 29084-K-0044 and by Semiconductor Research Corporation under Contract 83-01-003. K. K. Saluja was supported by the Computer Research Board, Australia, the Graduate School of the University of Wisconsin-Madison, and the National Science Foundation under Grant DCR-8509194. M. G. Karpovsky was supported by NSF Grant DCR83-1766. This paper was presented at the International Symposium on Fault-Tolerant Computing, June 1985.

S. M. Reddy is with the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242.

K. K. Saluja is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706.

M. G. Karpovsky is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215.

IEEE Log Number 8717735.

sors (STEP SC's) are discussed. It is argued that the data compression technique proposed could reduce the hardware complexity in built-in self-test (BIST) logic design as well as in designs using external tester environments.

Index Terms—**BIST, built-in self-test, data compression, self-testing and error-propagating space compressors, space compression, testable parity trees.**

I. Introduction

Testing logic circuits in a built-in self-test (BIST) environment requires a test setup as shown in Fig. 1(a). The response of the CUT for test sequence $\langle T \rangle$ is named $\langle R^* \rangle$, which is different from the response $\langle R \rangle$ of the fault-free circuit if the CUT has a fault detectable by $\langle T \rangle$. In BIST environment. the test data analyzer includes storage for the response expected from a fault-free circuit and a compressor. To reduce the amount of data represented by $\langle R \rangle$, often data compression is used to generate a *signature* from $\langle R \rangle$ which is compared to the signature of the response $\langle R^* \rangle$ of the CUT. When a number of nodes of the CUT, monitored in driving $\langle R \rangle$, is large it may be cost effective to introduce a *space compressor (SC)*, to reduce the width of the test data [1], [2], before a time compressor is used to derive the signature. The SC reduces the *width* of the test response from N to M, as illustrated in Fig. 1(b). This allows the use of a *parallel signature analyzer (PSA)* with M shift register stages rather than N stages. Reduction in the number of stages in PSA not only reduces the hardware cost of PSA but allows one to find a proper connection polynomial for the PSA, since often it is difficult to find irreducible or primitive polynomials of large degrees that are useful as connection polynomials for PSA's. Furthermore, small signatures would require simpler checkers to compare the signature after test against the expected signatures. In environments where external test data analyzers are used, often the logic values in internal nodes are brought out of a VLSI chip by time multiplexing these values, in test mode, onto data output pins of the chip [3]. Often such efforts require several clock cycles to bring out the test data [3]. In these environments an SC would facilitate bringing out the test data in one clock and carefully designed SC's could also require smaller on-chip hardware than a time multiplexing method. We give a specific example of this in Section III. In this paper, the design of cost effective and practical linear SC's is emphasized. Linear SC's use Exclusive-OR (EOR) or Exclusive-NOR (ENOR) gates only and hence are essentially parity generator trees.

Linear SC's were first suggested by Benowitz *et al.* [1] for use in built-in test. Their suggestion was to use M parity generator trees such that the inputs to a parity generator do not depend on the same set of input variables. When such SC's can be built, the number of EOR gates used will be $N-M$, where N is the number of CUT nodes monitored and M is the number of outputs of the SC. Often it may not be possible to construct the SC's proposed in [1] because the input dependencies of the monitored nodes could not be partitioned nontrivially or it may not be cost effective, because of the limitations of the achievable compression. Additionally, such SC's may mask manifested *errors* on the monitored nodes, e.g., when two errors occur on nodes feeding the same parity generator. To circumvent these difficulties, it was proposed in [2] that the linear SC's be constructed as the parity check circuits (also called syndrome calculators) of linear error-detecting codes. This allows the construction of SC's based on the *errors* that could occur at the monitored nodes due to modeled faults in the CUT.

These earlier works have not considered detection of faults in SC's and the design of cost-effective SC's. In this paper, we address these issues together with an analysis of the effectiveness of the linear SC's. In Section II, a new class of SC's is defined. Cost-effective designs of linear SC's based on single-error-correcting, double-error-detecting codes are given in Section III. An analysis of effectiveness of SC's is given in Section IV. In Section V, testable design of parity generator circuits is considered.

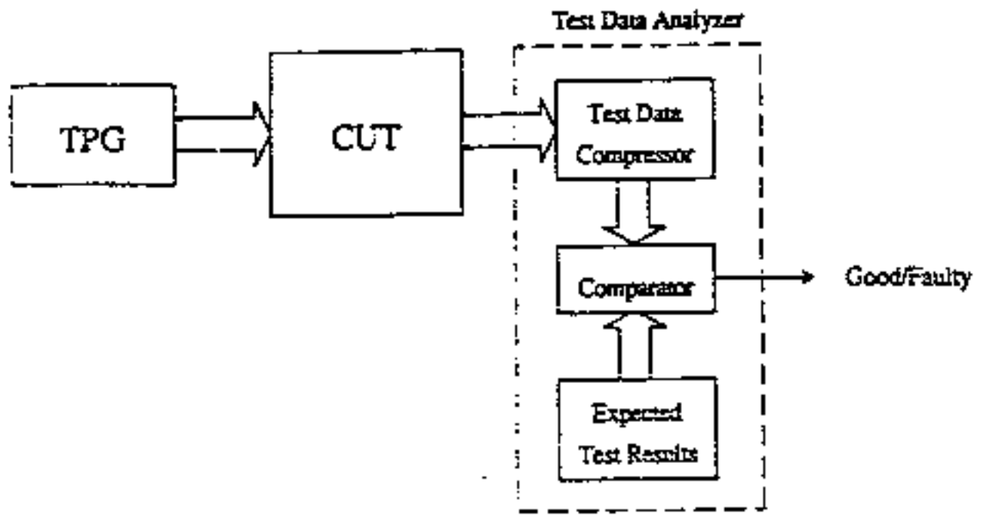


Fig. 1. Space compressor in BIST environment. (a) The BIST environment. (b) Space compressor.

II. Self-Testing and Error-Propagating Space Compressors (STEP SC's)

Let $X(t)$ and $Z(t)$ be the input to an SC and output of an SC, respectively, at time t, under the assumption that both the circuit under test as well as the SC are fault-free. Let T be the test sequence. Let $X^*(t)$ and $Z^*(t)$ be the corresponding entities when the CUT is being tested and let $\langle R \rangle$ and $\langle R^* \rangle$ be sequence of $X(t)$'s and $X^*(t)$ produced when the test sequence T is applied. The ideal requirements on SC's are given in [4]. A relaxed set of requirements for practical reasons [4] is given below. *It is assumed that either the CUT or the SC, and not both, is faulty at a given time.*

Definition 1: Let $\langle R_f^* \rangle$ denote the response of the CUT for test sequence $\langle T \rangle$ in the presence of a fault f_i in the CUT. An SC is called an *error-propagating (EP)* SC if in the absence of any fault in the SC, for at least one $X^*(t)$ in each $\langle R_f^* \rangle$, $X^*(t) \neq X(t)$, implies $Z^*(t) \neq Z(t)$.

Definition 2: An SC is called a *self-testing (ST)* SC if for each modeled fault, say f_j, in the SC there is at least one $X(t)$ in $\langle R \rangle$ that is a test for f_j.

Definition 3: An SC is called an *STEP SC* if it is self-testing and error propagating.

Definition 4: The compression ratio of an SC is M/N. (Note that smaller the compression ratio of an SC, the more *efficient* it is in reducing the number of nodes to be monitored directly or by a time compressor.)

Definition 5: $E(t) = X(t) \oplus X^*(t)$, where $\oplus$ is the bit-by-bit Exclusive-OR operator, is called an *error pattern*. Note that $X^*(t) = X(t) \oplus E(t)$ if $E(t) = X(t) \oplus X^*(t)$.

Design of an EP SC is simplified when the SC is a linear circuit. A linear circuit F has the property that $F(X^*(t)) = F(X(t) \oplus E(t)) =$

$F(X(t)) \oplus F(E(t))$. That is when the SC is fault-free, $Z^*(t) = Z(t) \ominus F(E(t))$. Note that when $X^*(t) \neq X(t)$, $E(t)$ is nonzero and, furthermore, $Z^*(t) \neq Z(t)$ if and only if $F(E(t)) \neq 0$. Therefore, a linear SC realizing a multiple output function F is an EP SC if and only if for every modeled fault in the CUT for at least one error pattern $E(t)$ produced by the CUT, $F(E(t)) \neq 0$. That is, the EP property of an SC can be analyzed by knowing the error patterns produced by the modeled faults in the CUT. This is analogous to the error-detecting codes. This analogy is further clarified by defining an M row, N column binary matrix H, in which each row corresponds to an output of the linear SC and there is a 1 in the ith column of the jth row of H if the ith input is included in the jth output. If H is taken as the parity check matrix of a linear code of length N, then H describes a linear code that detects all errors E, such that $HE^T = 0$, where E^T is the transpose of E. The parity check circuit or the syndrome calculator of a linear error-detecting code is therefore a linear SC, whose compression ratio is one minus the rate of the code (the rate of a code is the ratio of $\log_2$ of the number of codewords and the length of the codewords). Therefore, given a CUT and the nature of errors generated due to modeled faults and a chosen test sequence, the design of a linear EP SC can be achieved by the following steps.

1. Pick a subset, say EE, of error patterns created at the inputs of the SC, such that for each modeled fault in the CUT, EE contains *at least one error pattern* that occurs when the test sequence T exercises the CUT.
2. Pick a linear code, say LC, such that LC can detect all error patterns in EE.
3. The syndrome calculator circuit of LC is an EP SC for the CUT with the chosen test sequence T.

If only single faults in the CUT are of concern, a simple method to derive EE, the set of representative errors, is to determine the number of monitored nodes that depends on the logic value of a line in the CUT. The maximum of such numbers will indicate the maximum weight (i.e., the number of errors) of any pattern due to a *single fault* in the CUT. If this number is t, then the syndrome calculator of a t-error-detecting linear code will be an EP SC. In the next section, cost-effective designs for SC's that propagate error patterns with weight less than or equal to 2 or 3 are considered.

It should be recalled that linear combinational functions are easily realized by trees of EOR gates (such circuits are also called parity trees). A linear SC would therefore be self-testing if the response (R) of the fault-free CUT contains tests for the modeled faults in parity trees. One can augment T to force inclusion of tests for the parity. However, it may not be cost effective to augment T to include tests for parity trees constructed from arbitrarily realized EOR gates. For this reason, in Section V, testable realization of parity trees is considered.

III. Design of Error-Propagating Space Compressors (EP SC's)

In this section, linear EP SC's that propagate error patterns with up to 2 or 3 errors are studied. Consider single-error-correcting, double-error-detecting (SEC–DED) Hamming codes [5]. These codes detect error patterns with up to three errors and have length 2^k, $k \geq 2$. The parity check matrix of these codes have $(k + 1)$ rows and 2^k columns. Different specifications for the parity check matrix H of SEC–DED have been used. We use a particular specification to realize cost-effective linear SC's from these codes. This specification of the H matrix for length 16 SEC–DED Hamming code is illustrated in Fig. 2(a). If we number the rows 0, 1, $\cdots$, k, then it can be seen that the ith row is formed by concatenating 2^i alternating subsequences of solid 1's and 0's. The length of each subsequence is 2^{k-i} and each row starts with a subsequence of 1's. In general, it can be shown that the proposed specification of the length 2^k SEC–DED Hamming code's parity check matrix corresponds to the generator matrix of first-order Reed–Muller codes which is the dual of he SEC–DED Hamming code [5].

If the $(k + 1)$ parity trees for the given H matrix are constructed without using shared logic, the resulting SC would require $\{(N \log_2 N)/2 + N - \log_2 N\} \approx (N \log_2 N)/2$ EOR gates where $N = 2^k$ is the number of inputs to the SC. However, by sharing logic, it is possible to construct these SC's with less than $2N$ EOR gates. This is illustrated for the length 16 Hamming code by the shared logic equations given in Fig. 2(b). In this figure, $\oplus$ is mode 2 sum or EOR operation. The numbers summed are the indexes of the columns of the H matrix [cf. Fig. 2(a)]. The basic idea is to realize the parity trees corresponding to rows k and $k - 1$ (i.e., parity check P_k and P_{k-1}) without sharing logic and for a row i, $k - 2 \geq i \geq 0$, by sharing the logic used to realize the parity trees of rows with higher row indexes. For example, referring to Fig. 2(b), the parity tree for P_2 is obtained by using the outputs of the EOR gates computing $(1 \oplus 2)$ and $(9 \oplus 10)$ used to compute P_3. The number of EOR gates in realizing each output for the SC based in the length 16 SEC-DED code is also given in Fig. 2(b). The shared logic realization proposed is given in Fig. 2(c). The total number of EOR gates for this SC is 26. In general, for an SC realized from the H matrix of length $N = 2^k$ SEC–DED Hamming code, it can be shown that no more than $2(N - 1) - \log_2 N$ EOR gates are needed. These SC's propagate all errors of weight ≤ 3.

The H matrix of length $(2^k - 1)$ SEC Hamming code can be obtained from the H matrix of length 2^k SEC–DED Hamming code by deleting row 0 and column N. The SC corresponding to the SEC Hamming code propagated all errors of weight ≤ 2 and can be designed using no more than $2(N - 1 - \log_2 N)$ EOR gates. The compression ratios of SC's derived from the Hamming codes is $\approx (\log_2 N)/N$.

The complexity of the SC's derived from the Hamming codes was shown to be $\approx 2N$ EOR gates *if shared logic is allowed*. Another SEC code that required $\approx 2N$ EOR gates to realize the corresponding SC is obtained by letting only weight 1 and weight 2 binary r-tuples as the columns of the H matrix. The length N of such a code is $r(r + 1)/2$ and the compression ratio of the corresponding SC is $2/(r + 1)$. This H matrix has r 1's in each one of the r rows and the corresponding SC requires $r(r - 1) = 2(N - r) \approx 2N$ EOR gates and *does not use shared logic*, thus may require smaller area in integrated circuit realizations. An example of these SEC codes is given below. This code has column weight ≤ 2.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Formal proofs and formal procedures to design SC's of given complexity for the three classes of codes considered above are not given, since the construction of such SC's for Hamming codes is easily understood from the example given for the length 16 SEC–DED Hamming code. For the codes whose H matrix is given above, since no logic sharing is possible in the construction of the corresponding SC's, the design is straightforward.

In this section, we have given SC's that propagate all error patterns of weight ≤ 3 (derived from SEC–DED codes) or ≤ 2 (derived from SEC codes). The SC's require less than $2N$ EOR gates. Results similar to SEC–DED codes given here can be obtained for the class of equivalent codes considered in [6]. Details are not given here.

As an example of possible application of SC's, consider the built-in testing facilities in the Motorola 32 bit microprocessor MC68020 [3]. One of the testability features added to MC68020 is to bring out the 116 outputs of the microprogram ROM's by time multiplexing onto 32 data lines (it takes 4 clock cycles to get data from the 116 outputs). Instead, one may use a space compressor that will compress the 116 outputs onto 32 outputs. This way one can reduce the test time and additionally the area complexity of the required SC can be seen to be

$$H_{16} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} \\ ROW\ 0 \\ ROW\ 1 \\ ROW\ 2 \\ ROW\ 3 \\ ROW\ 4 \end{matrix}$$

(a)

$P_4 = ((1 \oplus 3) \oplus (5 \oplus 7)) \oplus ((9 \oplus 11) \oplus (13 \oplus 15))$;7 GATES

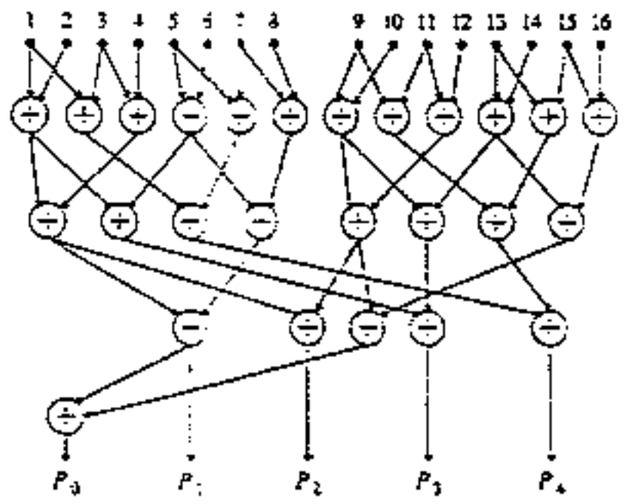
$P_3 = ((1 \oplus 2) \oplus (5 \oplus 6)) \oplus ((9 \oplus 10) \oplus (13 \oplus 14))$;7 GATES

$P_2 = ((1 \oplus 2) \oplus (3 \oplus 4)) \oplus ((9 \oplus 10) \oplus (11 \oplus 12))$;5 GATES

$P_1 = ((1 \oplus 2) \oplus (3 \oplus 4)) \oplus ((5 \oplus 6) \oplus (7 \oplus 8))$;3 GATES

$P_0 = ((1 \oplus 2) \oplus (3 \oplus 4)) \oplus ((5 \oplus 6) \oplus (7 \oplus 8)) \oplus ((9 \oplus 10) \oplus (11 \oplus 12)) \oplus ((13 \oplus 14) \oplus (15 \oplus 16))$;4 GATES

(b)



(c)

Fig. 2. Design of an SC based on a length 16 SEC-DED code. (a) Parity check matrix of length 16 SEC-DED code. (b) Parity check equations for the tree realization of the H matrix. (c) SC based on length 16 SEC-DED code.

comparable to that of a 4 to 1 multiplexer needed to bring out the outputs of the control ROM's.

IV. Analysis of Effectiveness of Space Compressors

Even though it is desirable to construct SC's that will propagate all errors caused by the modeled faults in the CUT, often it may not be cost effective to determine the set of all possible error patterns and then choose an appropriate EP SC. In such cases, it may be appropriate to use an SC that provides adequate error propagation for random errors. To derive a figure of merit for such choices, assume that the error patterns are uniformly distributed over the space of nonzero binary N-tuples. Earlier it was pointed out that an error E is not propagated by a linear SC if and only if $HE^T = 0$, where H is the $M \times N$ parity check matrix corresponding to the SC. From matrix theory, it is known that the number of nonzero error patterns E for which $HE^T = 0$ is $2^{N-M} - 1$. Hence, the proportion of errors not propagated by an N input M output SC is $\leq (2^{N-M})/(2^N - 1) \approx 2^{-M}$.

The simple analysis given above indicates that the probability that an SC does not propagate an error decreases exponentially with M, the number of its outputs. In deriving this measure, we have implicitly assumed that it is necessary to propagate each error created by the faulty CUT. However, a faulty CUT often produces several different error patterns when exercised by the test sequence T. If p is the minimum number of *different error patterns* caused by a fault in the CUT, then the probability of not propagating errors is 2^{-pM}. Therefore, if for a given CUT with a chosen test sequence T, the parameter p is large, small values of M (e.g., $M = 1$) may be sufficient, leading to SC's with minimum compression ratios.

V. Design of Testable Parity Trees

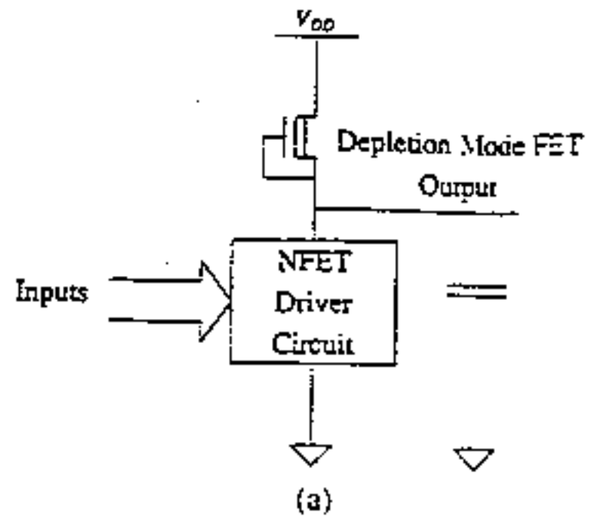
The design of self-testing linear SC's can be seen as equivalent to the design of embedded self-testing parity trees. This problem was studied earlier in the context of designing totally self-checking (TSC) checkers for single parity codes and certain linear block codes [7]. The following fault models were used for the faults in the two-input EOR gate used in the construction of the parity trees.

1) The two inputs and the output of the EOR gate may be stuck-at-0 and/or stuck-at-1.

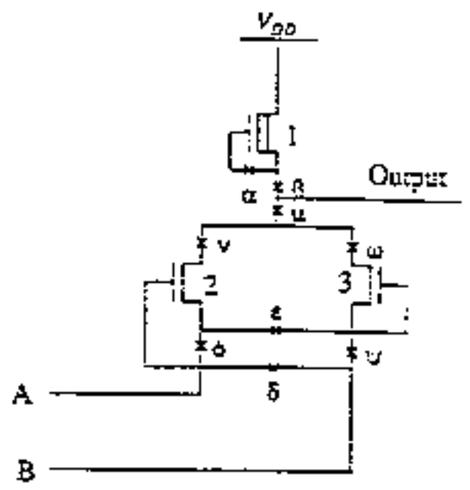
2) The faults in the EOR gates are such that all four input combinations are necessary to detect the faults in it.

Given these fault models, procedures to design parity trees, when the inputs (equivalent to (R) in the current discussion) satisfy certain necessary and sufficient conditions, were given in [7] and [8]. The first fault-model used is inadequate to represent faults in EOR gates and the second fault model may be pessimistic, as discussed later in this section.

Let us consider certain NMOS circuits to realize EOR and ENOR or EQUIVALENCE gates. The NMOS gates to be presented require only three tests to detect line stuck-at faults, FET stuck-on and stuck-open faults, line open faults, and shorts between adjacent lines. Before we give these circuits, it is important to review FET stuck-open faults in NMOS circuits. An NMOS circuit is made up of a load consisting of a depletion mode FET and a driver circuit which is an interconnection of enhancement NFET's. The general form of an NMOS gate is shown in Fig. 3(a). If the depletion mode FET is open, then the output of the gate can be assumed to be permanently at logic 0, unless due to a combination of values of inputs, the output can be set to 1. The latter case is possible only if there is a path through some pass transistor [9] in the NFET driver circuit that will set the output to 1. If this were the case, the output may retain the logic 1 value (due to the capacitive load on the circuit) for a while after an input that set it to 1 is removed. In such a case, the only way to test the depletion mode FET stuck-open fault is to initialize the output to 0 and then attempt to make it a logic 1. This implies a two-pattern test. In the sequel, we will give an NMOS ENOR circuit that uses pass transistors.



(a)



(b)

Test Input		Fault												
A	B	1/ON	1/OP	2/ON	2/OP	3/ON	3/OP	A/0	B/0	A/1	B/1	OUT/0	OUT/1	Bridge Between A & B or ε & δ or α & γ
0	0		×	×		×				×	×	×		
0	1	×			×				×	×			×	×
1	0	×					×	×			×		×	×

(c)

Fig. 3. A three-transistor ENOR gate and its tests. (a) General structure of an NMOS gate. (b) An NMOS Exclusive-NOR gate. (c) Tests for faults in the ENOR gate.

The problem of detecting the depletion mode FET stuck-open fault will be discussed.

In Fig. 3(b), a three-transistor NMOS ENOR gate is given [9]. To the best of our knowledge, no complete test analysis of this gate has appeared in open literature. In Fig. 3(c), tests for all possible single faults are given. The notation used in Fig. 3(c), to designate a fault, is x/y, where x is a circuit lead or an FET identifier given in Fig. 3(b) and y is the fault type; for FET's it is stuck-on (ON) or stuck-open (OP) and for lines it is stuck-at-0 (0) or stuck-at-1 (1). Possible fault sites are shown in Fig. 3(b). It can verified that all stuck-at or line open faults in the circuit are equivalent to one of the faults given in Fig. 3(c). From Fig. 3(c), it can be seen that the tests for single faults in the ENOR gate of Fig. 3(b) can be obtained out of the three inputs 00, 01, and 10. Actually all multiple faults can be detected by tests constructed of 00, 01, and 10. Furthermore, these three inputs must be applied to test for 1/OP, 2/OP, and 3/OP faults, respectively. That is, the input 11 is redundant with respect to fault coverage for faults in the ENOR gate. As mentioned earlier, the stuck-open fault in FET1 requires that 01 or a 10 input precede a 00 input to detect the fault. This is required since the input 11 could set the output to 1 even in a faulty gate and if that input has been applied prior to the application of test input 00 and before a 01 or 10, then the input 00 could fail to detect the fault FET1 stuck-open

The analysis given above indicates that it may be possible to construct parity trees that can be tested for all faults even when it is not possible to apply 11 input to a two-input parity gate, thus potentially making it possible to design self-testing SC's for some of the cases for which the results in [8] would indicate that a self-testing

$$S_1 = \begin{matrix}1\\0\\1\\0\\0\end{matrix} \quad S_2 = \begin{matrix}1\\0\\0\\0\\1\end{matrix} \quad S_3 = \begin{matrix}1\\1\\1\\0\\1\end{matrix}$$

(a)

$$S_4 = \begin{matrix}1\\0\\0\\0\end{matrix} \quad S_5 = \begin{matrix}1\\1\\0\\1\end{matrix} \quad S_6 = \begin{matrix}1\\1\\0\\1\end{matrix}$$

(b)

Fig. 4. Sequences to design tests for parity trees. (a) Set S_E. (b) Set S_N.

linear SC is not realizable for given (R). In the remainder of this section, we pursue this problem and we apply the derived results to design STEP SC's to compress test response from decoders.

To design parity trees using the ENOR gate of Fig. 3(b), due to logic level shifts at the output of the ENOR gate, it is not prudent to cascade two such gates. Buffers (inverters) must be added between two stages to restore logic levels. *It should be noted that the tests given for the ENOR gate of Fig. 3(b) would also detect faults in the two-input parity functions using a buffered version of this gate.* A parity tree can be constructed either by using EOR gates or ENOR gates. The following readily provable equalities provide the basis for this claim. In these equalities, $\ominus$ and $\odot$ denote EOR and ENOR operators, respectively.

for n an odd integer:

$$x_1 \ominus x_2 \ominus x_3 \cdots \ominus x_n = x_1 \odot x_2 \odot x_3 \cdots \odot x_n$$

and for n an even integer:

$$x_1 \ominus x_2 \ominus x_3 \cdots \ominus x_n = \overline{x_1 \odot x_2 \odot x_3 \cdots \odot x_n}.$$

Earlier, several authors had studied the problem of test generation to detect faults in the parity trees [10]–[12]. The parity trees constructed by the ENOR gate of Fig. 3(b) (or its buffered variations) require reconsideration of the problem because of the need for the test sequences that include a test input 00 preceded by input 01 or 10 described earlier. Our procedure for constructing test sequences is analogous to those given in [10] and [11] for *normal ENOR* gates.

Let us consider the set $S_E = \{S_1, S_2, S_3\}$ given in Fig. 4(a). Clearly, $S_1 \ominus S_2 = S_3$; thus, the set S_E is closed under EOR operation. Furthermore, if S_i and S_j, $i \neq j$, are treated as test inputs for an EOR gate then they contain at least one input 00 which is preceded by an input 01 to 10. Elements of the set S_E can be assigned to the EOR gates forming a tree such that inputs assigned to every gate are S_i, S_j with $i \neq j$. This can be done by starting with assigning an element of S_E to the inputs of the output EOR gate. This procedure can then be repeated up the tree until inputs to the first level EOR gates are specified. Once the inputs to the first level EOR gates are specified, the tree inputs that need to be applied can be readily determined by reading across the components of set S_E assigned to each input of the first level EOR gates. Thus, a single faulty gate in the tree of EOR gates can be detected by a test sequence of length 5. From the set S_E it is also evident that not only three tests given in Fig. 3(c) are applied to each EOR gate in the tree, but all four two-tuples are applied to every EOR gate.

Similar arguments can be used for the set S_N of Fig. 4(b) which gives the tests for parity tree constructed ENOR gates. We conclude that a single faulty gate in the tree of ENOR gates can be detected by a test sequence of length 4. However, in this case each ENOR gate is applied to only those three tests which are given in Fig. 3(c). If it is also desirable that the test input 11 be applied (e.g., in functional testing environment where the actual gate type used is not known), then the additional test can be applied by having all inputs as 1's and making the length of test sequence 5. We note that the device count in an ENOR tree using buffered ENOR gates would normally be higher than in an EOR tree, unless it is possible to cascade two unbuffered

A	B	C	D	E	F	G	H
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0

Fig. 5. The output of a decoder with disable input.

ENOR gates before level restoring inverters are used. Even if two levels of ENOR gates are used before level restoring inverters are used, the tests given above for ENOR trees would remain effective. Furthermore, the logic delay in ENOR trees could be higher than that for EOR trees.

Next we want to demonstrate, through a nontrivial example, that a STEP SC with a single output can be constructed as a tree from the EOR gates discussed in this section, whereas such a construction is not possible if the normal EOR gates that do not use pass transistors are used. The gates of the latter type will need all four two-input combinations to detect faults. Consider a decoder that is used to select one of eight registers in a microprocessor. Normally such decoders have a disable input to deselect all lines. That is, such a decoder's outputs can only be the nine binary 8-tuples shown in Fig. 5. If no more than one output can be erroneous at any time, then a single output parity tree would be an EP SC for this decoder. By observing the set of possible outputs from the decoder, it is readily seen that independent of the choice of inputs to an EOR gate, a 11 input cannot be applied to the EOR gates in the top level of the EOR tree. However, if the sequence of inputs given in Fig. 5 is applied to the input of the parity tree constructed from the EOR gates proposed in this paper, every EOR gate gets a 01 and 10 input and a 00 input preceded by a 01 or 10 input. Hence, such a single output SC will be STEP SC. The example given can be generalized to all decoders with disable inputs.

If the decoder does not have a disable input, then an STEP SC will need at least two outputs. The need for two outputs can be established by the following argument. Note that the possible fault-free inputs to the SC (from the decoder under test) are N-tuples with a single one. If an SC with a single output is to propagate all single errors in the fault-free response, the output of it for an input with two ones, say $1100 \cdots 00$, should be different from its output for input $100 \cdots 0$ as well as for input $0100 \cdots 0$. This is possible if and only if the single output SC realizes a function whose value is identical for inputs $100 \cdots 0$ and $010 \cdots 0$. By applying this argument iteratively to the following $(N-1)$ inputs with two ones, $1100 \cdots 0, 011 \cdots 0, \cdots, 0 \cdots 110$, and $0 \cdots 011$, we note that the single output SC must realize a function which takes identical values for all inputs with a single one in them. This implies that under normal operation the output of the SC is a constant and hence such an SC is not self-testable. (Note that this conclusion is independent of how the SC is constructed. That is, any logic circuit, not necessarily linear, would have this property.) It is readily possible to design STEP SC's with two outputs to compress the outputs from decoders, without disable inputs, that are subject to single faults. An example of such an SC for an eight-output decoder can be obtained by generating the parity of the first four outputs and the last four outputs, respectively.

It is interesting to note that the example given immediately above indicates that the number of outputs of an STEP SC are not only dictated by the errors it is to propagate but also by the desired self-testing property. It is worth pointing out that the STEP SC designed for decoders without disable inputs is also a TSC checker if single errors are assumed in the output of the decoder. This is another example of the construction of testable linear trees that could not have been constructed from earlier results [7], [8].

VI. Concluding Remarks

In this correspondence, a compression technique, called space compression, that is applicable to compress test data in both external test environment and BIST environment was proposed and analyzed. In BIST environment an SC is to be followed by a time compressor [2]. It was argued that linear space compressors are effective and amenable to simple analysis. The concept of self-testing error-propagating SC's was introduced. Finally, an example to illustrate where the proposed EOR and ENOR designs would lead to STEP SC's, while earlier designs of EOR and ENOR gates will not provide STEP SC's, was given.

References

[1] N. Benowitz, D. F. Calhoun, G. E. Anderson, J. E. Bauer, and C. T. Joeckel, "An advanced fault isolation system for digital logic," *IEEE Trans. Comput.*, vol. C-24, pp. 489–497, May 1975.

[2] K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time," in *Proc. 1983 Int. Test Conf.*, Oct. 1983, pp. 83–88.

[3] J. Kuban and J. Salick, "Testability features of the MC68020," in *Proc. 1984 Int. Test Conf.*, Oct. 1984, pp. 821–826.

[4] S. M. Reddy, K. K. Saluja, and M. Karpovsky, "A data compression technique for built-in self-test," in *Dig. Int. Symp. Fault-Tolerant Comput.*, Ann Arbor, MI, June 1985, pp. 294–299.

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

[6] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Develop.*, July 1970, pp. 395–401.

[7] J. Khakbaz, "Self-testing embedded parity trees," in *Dig. Int. Symp. Fault-Tolerant Comput.*, Santa Monica, CA, June 1982, pp. 109–116.

[8] J. Khakbaz and E. J. McCluskey, "Self-testing embedded parity checkers," *IEEE Trans. Comput.*, vol. C-33, pp. 753–756, Aug. 1984.

[9] J. Mavor, M. A. Jack, and P. B. Denyer, *Introduction to MOS LSI Design*. Reading, MA: Addison-Wesley, 1982.

[10] D. C. Bossen, D. L. Ostapko, and A. M. Patel, "Optimal test patterns for parity networks," in *Proc. AFIP 1970 Fall Joint Comput. Conf.*, Nov. 1970, pp. 63–68.

[11] J. P. Hayes, "On realization of Boolean functions requiring a minimal or near-minimal number of tests," *IEEE Trans. Comput.*, vol. C-20, pp. 1506–1513, Dec. 1971.

[12] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE Trans. Comput.*, vol. C-26, pp. 29–33, Jan. 1977.