A Data Compression Technique for Built-in Self Test

Sudhakar M. Reddy*
Department of Electrical
and Computer Engineering
University of Iowa
Iowa City, Iowa 52242

Kewal K. Saluja**
Department of
Electrical Engineering
University of New Castle
New Castle, Australia

Mark Karpovsky***
Department of
Electrical Engineering
Boston University
Boston, Massachusetts 02215

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 an exclusive OR and exclusive NOR gate are analyzed and the use of these gates in the design of self-testing and error-propagating space compressors (STEP SCs) are discussed. It is argued that the data compression technique proposed could reduce the hardware complexity in built-in self test (BIST) logic designs as well as in external tester environments.

I. Introduction

Testing logic circuits requires a test set-up as shown in Figure 1. The response of the CUT for test sequence <T> is named <R*>, which is different from the response <R> of the fault-free circuit if the CUT has a fault detectable by <T>. In built-in self test environments the test data analyzer includes storage for response expected from a fault-free circuit and a compressor. To reduce the amount of data represented by <R>, often data compression is used to generate a "signature" from <R> and is compared to the signature of the response <R*> of the CUT. When the number of nodes, of the CUT, monitored in deriving <R> is large, it is cost effective to introduce a 'space compressor' to reduce the 'width' of the test data [1,2], before a time compressor is used to derive the signature. In this paper the design of linear space compressors is emphasized. Linear space compressors (SCs) use exclusive OR or exclusive NOR gates only and hence are essentially parity generator trees.

* The research reported was supported in part by U.S. Army Research Office, Contract-No. DAAG 29084-K-0044 and by Semiconductor Research Corporation, Contract No. 83-01-003.

** Research supported by Computer Research Board and Radio Research Board Australia and IRAC of the Univ. of New Castle, Australia.

***Research supported by NSF Grant No. DCR83-1766.

Linear SCs 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 SCs 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 SCs proposed in [1] (because the input dependencies of the monitored nodes could not be partitioned non-trivially) or it may not be cost effective, because of the possible compression. Additionally such SCs may mask manifested 'errors' on the monitored nodes; e.g. when two errors occur in outputs feeding the same parity generator. To circumvent these difficulties, in [2], it was proposed that the linear SCs be constructed as the parity check circuits (also called syndrome calculators) of linear error-detecting codes. This allows the construction of SCs based on the 'errors' that could occur at the monitored nodes due to modeled faults in the CUT.

These earlier works have not considered the detection of faults in SCs and the construction of cost effective designs for SCs. In this paper, we address these issues together with an analysis of the effectiveness of the linear SCs. In Section 2 STEP SCs are defined. Cost effective designs of linear SCs based on single-error-correcting and double-error-detecting codes are given in Section 3. An analysis of effectiveness of SCs is given in Section 4. In Section 5 testable design of parity generator circuits are considered.

II. Self-Testing and Error-Propagating Space Compressors (STEP SCs)

Let X(t) and Z(t) be the input to a SC and the output of the 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 <R> and <R*> be the sequence of X(t)s and X*(t)s produced when the test sequence T is applied. The ideal requirements on SCs are stated below.

REQ 1: $X(t) \neq X^*(t)$ implies $Z(t) \neq Z^*(t)$.

REQ 2: For every modeled fault in SC there is a X*(t) in every <R*> such that the corresponding Z*(t) is not equal to Z(t) (note that in this case Z*(t) is the output of the faulty SC when X*(t) (which may be different from X(t)) is the input.

The requirements stated above may often be much stronger than desired and/or difficult to meet. For example often a modeled fault in the CUT will be manifested by more than one X*(t) being different from the corresponding X(t) (i.e. the fault may lead to erroneous values on the monitored CUT nodes for several tests). It is desirable that for at least one such erroneous X*(t) the output of the SC be different than the expected output. The second requirement, stated above, essentially requires that every possible response sequence of the CUT be a test for faults in SC. Clearly, satisfaction of this requirement is often difficult to analyze and satisfy. Furthermore, it is meaningful only if faults are assumed to occur simultaneously in the CUT and the SC. For this reason a relaxed set of requirements are given below in definitions 1 and 2. It is assumed that either the CUT or the SC and not both, is faulty at a given time.

Definition 1: A SC is called an Error-Propagating (EP) SC iff when the SC is fault-free, for at least one X*(t) in each $\langle R_1^* \rangle$, such that X*(t) ≠ X(t), it is the case that Z*(t) ≠ Z(t).

Definition 2: A SC is called a Self-Testing (ST) SC iff for each modeled fault, say ft, in the SC there is at least one input X(t) in <R> (the fault-free response of CUT under test input T) that is a test for ft.

Definition 3: A SC is called a STEP SC if it is self-testing and error-propagating.

Definition 4: Compression Ratio of a SC is M/N.

The design of linear SCs based on linear error-detecting codes and minimum compression ratios achievable were considered in [2].

Definition 5: E(t) = X(t) ⊕ X*(t) is called an error pattern.

Design of error-propagating SC is simplified when the SC is a linear circuit. A linear circuit F (possibly realizing multiple outputs) has the property that F(X*(t)) = F(X(t) ⊕ E(t)) = F(X(t)) ⊕ F(E(t)). That is, when the SC is fault-free, Z*(t) = Z(t) ⊕ F(E(t)). Note that when X*(t) ≠ X(t), E(t) is non-zero and furthermore Z*(t) ≠ Z(t) if and only if F(E(t)) ≠ 0. Therefore a linear SC is an EP SC iff for every modeled fault in the CUT for at least one error pattern E(t) produced by the CUT F(E(t)) ≠ 0. That is, the EP property of an SC CUT 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 a M row, N column binary matrix H, in which each row of H corresponds to an output of the linear SC and there is a one in ith column of the jth row 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 $HE^T \neq 0$, 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 (rate of a code is the ratio of $\log_2$ of the number of code words and the length of the code words). Therefore given a CUT and the nature of errors generated due to modeled faults and a chosen test sequence, the design of a linear error-propagating SC can be achieved by the following steps:

1. Pick a subset, say EE, of errors created at the monitored nodes 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 all erorrs in EE are detected by LC.
3. The syndrome calculator circuit of LC is an error-propagating SC for the CUT with the chosen test sequence T.

A simple procedure to derive EE, the set of representative errors, is to determine the number of monitored nodes that depend on the logic value of a line in the CUT. The maximum of such numbers will indicate the maximum weight or number of errors in any error 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 SCs that propagate error patterns with weight less than or equal to 2 or 3 are considered.

It should be recalled that the linear combinational functions are easily realized by trees of exclusive OR 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 trees. However it may not be cost effective to augment T or to augment it to include tests for parity trees constructed from arbitrarily realized EOR gates. For this reason in Section 5, testable realization of parity trees are considered.

III. Design of Error-Propagating SCs

In this section linear error-propagating SCs that propagate error-patterns with up to 2 or 3 errors are considered. Consider single-error-correcting, double-error-detecting (SEC-DED) Hamming codes [3]. These codes detect error patterns with up to three errors and have length 2^k, k > 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 SCs from these codes. This specification of the H matrix for length 16 SEC-DED Hamming code is shown in Figure 2. If we number the rows 0,1, ..., k, then it can be seen that ith row is formed by concatenating 2^i subsequences of solid ones and zeros. The length of each subsequence is 2^{k-i} and each row starts with a subsequence of ones. In general it can be shown that length 2^k SEC-DED Hamming code's parity check matrix can be constructed in this fashion.

If the (k+1) parity trees for the given H matrix are constructed without using shared logic, the resulting SC would require $\approx (N \log_2 N)/2$ EOR gates. However by sharing logic it is possible to construct these SCs with only $< 2N$ EOR gates. This is illustrated for the length 16 Hamming code by the shared logic equations given in Figure 3. In this Figure '$\oplus$' is mod 2 sum or EOR operation. The numbers summed are the indices of the columns of the H matrix. The basic idea is to realize the parity trees corresponding to row i, $0 \leq i \leq k-2$, by sharing the logic used to realize the parity trees of rows with higher row index. For example the parity tree for P_3 is obtained by using the ouputs of the EOR gates computing (1+2) and (9+10) used to compute P_2. The number of EOR gates in realizing each output for the SC based on the length 16 SEC-DED code are also given in Figure 3. The total number of EOR gates for this SC is 26. In general for a 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 SCs propagate all errors of weight ≤ 3. The H matrix of a length (2^k-1) SEC Hamming code can be obtained from the H matrix of a length 2^k SEC-DED Hamming code by deleting row o and column N. The SC corresponding to the SEC Hamming code can be designed using no more than $2(N-1-\log_2 N)$ EOR gates. The compression ratios of SCs derived from the Hamming codes is $\approx (\log_2 N)/N$. The complexity of the SCs derived from the Hamming codes was shown to be $\approx 2N$ EOR gates if shared logic is allowed. Another SEC code that requires $\approx 2N$ 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 ones in each one of the r rows and the corresponding SC requires $r(r-1)/2 = 2(n-r) \approx 2N$ EOR gates. This SC does not use shared logic. An example of these SEC codes is given in Figure 4.

In this section we have given SCs that propagate all error patterns of weight ≤ 3 (derived from SEC-DED codes) or ≤ 2 (derived from SEC codes). The SCs require less than 2N EOR gates. Results similar to SEC-DED codes given here can be obtained for the equivalent class of codes considered in [10]. Details are not given here.

IV. Effectiveness Analysis of SCs

Even though it is desirable to construct SCs 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 error-propagating 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 non-zero binary N-tuples. Earlier it was shown that an error E is not propagated by an SC if and only if $HE^T \neq 0$, where H is the (M X N) parity check matrix corresponding to the SC. From matrix theory it is known that the number of non-zero error patterns E for which $HE^T = 0$ is $\leq 2^{N-M} - 1$. Hence the proportion of errors not propagated by a 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 t is the minimum number of different error patterns caused by a fault in the CUT, then the probability of not propagating errors is 2^{-tM}. Therefore if for a given CUT with a chosen test sequence T, the parameter t is large, small values of M (e.g. M=1) may be sufficient, leading to SCs with small compression ratios.

V. Design of Testable Parity Trees

The design of self-testing linear SCs is essentially the same as 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 [4]. The following fault models were used for the faults in the two input exclusive OR (EOR) gate used in the construction of the parity trees:

(i) the two inputs and the output of the EOR gate may be stuck-at-0 and/or stuck-at-1
(ii) 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 sufficient conditions, were given in [4,5]. 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.

Next we consider certain NMOS circuits to realize EOR and exclusive NOR (ENOR) or EQUIVALENCE (EQN) 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 NFETs. The general form of an NMOS gate is shown in Figure 5. 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 I (cf. Figure 5), the output can be set to 1. The latter case is possible only if there is a path through some pass transistors [6] in the NFET network 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 a NMOS ENOR circuit that uses pass transistors. The problem of detecting the depletion mode FET stuck-open fault will be discussed.

In Figure 6(a), a three transistor NMOS ENOR gate is given. This gate appears in [6] but, to the best of our knowledge, no test analysis of this gate has appeared in open literature. In Figure 7 tests for all possible single faults are given. The notation used in Figure 7, to designate a fault is x/y, where x is a circuit lead or a FET identifier and y is the stuck-at value; for FETs it is stuck-on or stuck-open (or stuck-op) and for lines it is stuck-at-0 or stuck-at-1. Other possible fault sites are shown in Figure 6(a). It can be readily verified that stuck-at or line-open faults on these lines are equivalent to one of the faults given in the table. From the table it can be readily seen that the tests for single faults in the ENOR gate of Figure 6(a) 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 the faulty gate and if that input has been applied prior to the application of test input 00 and before a 01 or 10 input has been applied, the test could fail to detect the fault FET 1 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 SCs for some of the cases for which the results in [5] would indicate that a self-testing linear SC is not realizable for given <R>. In the remainder of this section we pursue this problem and we apply these results to design STEP SCs to compress test data from decoders.

To design parity trees using the ENOR gate of Figure 6(a), 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 Figure 6(a) would also detect faults in the two-input parity functions shown in Figure 6(b), since the additional gates included are standard NMOS inverters. A parity tree can be constructed either by using EOR gates or ENOR gates. The following readily provable theorem provides the basis for this claim.

Theorem 1: The output function of a parity tree constructed of two input ENOR gates driven by inputs $x_1, x_2, \ldots, x_n$ is $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ if n is an odd integer and is $\overline{x_1 \oplus x_2 \oplus \ldots \oplus x_n}$ if n is an even integer.

Earlier several authors have studied the problem of test generation to detect faults in the parity trees [7-9]. The parity trees constructed by the EOR and ENOR gates of Figure 6(b) 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. In a manner entirely analogous to the procedure for trees of normal 'EOR gates' [7-8], the test sequences for trees of EOR and ENOR gates of Figure 6(b) can be constructed from the length 5 and length 4 sequences given in Figure 8, as sets S_E and S_N, respectively. Thus it is possible to derive test sequences of length 5 (4) to detect single faulty gates in trees of EOR (ENOR) gates of Figure 6(b). This result should be compared with the earlier result for trees of 'normal' EOR gates in which single faulty gates can be detected by four tests.

A point is to be made regarding the parity trees constructed in this section. First point to be made is that the device count in a ENOR tree would normally be higher than in an EOR tree, unless it is possible to cascade two ENOR gates of Figure 6(a) before level restoring inverters are needed.

Next we want to demonstrate, through a non-trivial example, that a STEP SC with a single output using a EOR tree can be constructed from the gates given in Figure 6, 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 input combinations to detect faults. Consider a decoder that is used to select one of 8 registers in a microprocessor. Normally such decoders have a disable input to de-select all lines. That is, such a decoder's outputs can only be the nine binary eight tuples shown in Figure 9. In the presence of single faults in the decoder the outputs of the decoder can be expected to contain at most one error. Hence a single bit parity tree would be an SC with error propagating property. 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 Figure 9 are 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 the SC shown will be a STEP SC. The example shown can be generalized to all decoders with disable inputs.

If the decoder does not have a disable input then a 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 .... 00, should be different from its output for input 100 ... 0 as well as for input 0100 ... 0. This is possible if and only if the single output SC realizes a function whose value is identical for inputs 100 ... 0 and 010 ... 0. By applying this argument iteratively to the following (N-1) inputs with two ones, 1100 ... 0, 01100 ... 0, ...00 ...110 and 00 ... 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 SCs with two outputs to compress the outputs from decoders, without disable inputs, that are subject to single faults. An example of such a SC for a 8 output decoder can be obtained by generating the parity of the first four outputs and the last four outputs, respectively.

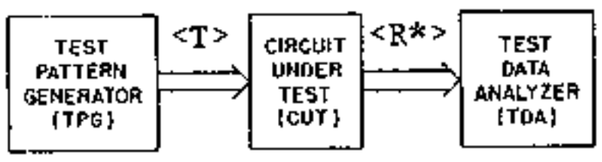
It is interesting to note that the example given above indicates that the number of outputs of a STEP SC are not only dictated by the errors it is to propagate but also by the desired self-testing property. It is worth pointing out that the STEP SCs designed for decoders without disable inputs is also a TSC checker if single errors are assumed in the output of the decoder.

VI. Concluding Remarks

In this paper 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 a SC is to be followed by a time compressor [2]. It was argued that linear space compressors (SCs) are effective and amenable to simple analysis. The concept of self-testing error-propagating SCs was introduced. Design and tests for EOR and ENOR gates that appear to provide reduced device count and lead to STEP SCs were presented. Finally an example to illustrate where the proposed EOR and ENOR designs would lead to STEP SCs, while earlier designs of EOR and ENOR gates will not provide STEP SCs was given.

References

1. N. Benowitz, D.F. Calhoun, G.E. Alderson, J.E. Bauer and C.T. Joeckel, "An Advanced Fault Isolation Systems for Digital Logic," IEEE Trans. on Comp., Vol. C-24, No. 5, pp. 489-497, May 1975.
2. K.K. Saluja and M. Karpovsky, "Testing Computer Hardware Through Data Compression in Space and Time," Proceedings of 1983 International Test Conference, pp. 83-88, October 1983.
3. W.W. Peterson and E.J. Weldon, Jr., "Error-Correcting Codes," Second Edition, The MIT Press, Cambridge, Massachusetts.
4. J. Khakbaz, "Self-Testing Embedded Parity Trees," Digest of Papers International Symposium on Fault-Tolerant Computing, pp. 109-116, June 1982.
5. J. Khakbaz and E.J. McCluskey, "Self-Testing Embedded Parity Checkers," IEEE Trans. on Comp., pp. 753-756, August 1984.
6. J. Mavor, M.A. Jack and P.B. Denyer, "Introduction to MOS LSI Design," Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.
7. D.C. Bossen, D.L. Ostapko and A.M. Patel, "Optimum Test Patterns for Parity Networks," Proc. AFIPS 1970 Fall Joint Computer Conference, pp. 63-68, November 1970.
8. J.P. Hayes, "On Realization of Boolean Functions Requiring a Minimal or Near-Minimal Number of Tests," IEEE Trans. on Comp., pp. 1506-1513, December 1971.
9. S.C. Seth and K.L. Kodandapani, "Diagnosis of Faults in Linear Tree Networks," IEEE Trans. on Comp., pp. 29-33, January 1977.
10. M.Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DEC Codes," IBM J. Res. Develop., pp. 395-401, July 1970.



<T> Test Sequence
<R*> Response

Figure 1:
The components of logic circuit testing

COLUMN

	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	ROW 0
	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	ROW 1
H_{16} =	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	ROW 2
	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	ROW 3
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	ROW 4

FIGURE 2:
A PARITY CHECK MATRIX OF LENGTH 16 SEC-DED CODE

P_1 = ((1+3)+(5+7))+((9+1)+(13+15))
7 GATES
P_2 = ((1+2)+(5+6))+((9+10)+(13+14))
7 GATES
P_3 = ((1+2)+(3+4))+((9+10)+(11+12))
5 GATES
P_4 = ((1+2)+(3+4))+((5+6)+(7+8))
3 GATES
P_5 = (((1+2)+(3+4))+((5+6)+(7+8)))
+ (((9+10)+(11+12))
+ ((13+14)+(15+16)))
4 GATES

FIGURE 3:
THE PARITY CHECK EQUATIONS FOR THE H MATRIX OF FIGURE 2

$$H = \begin{matrix} 1&0&0&0&1&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&1&0 \\ 0&0&1&0&0&1&0&1&0&1 \\ 0&0&0&1&0&0&1&0&1&1 \end{matrix}$$

FIGURE 4:
A SEC CODE WITH COLUMN WEIGHT ≤ 2

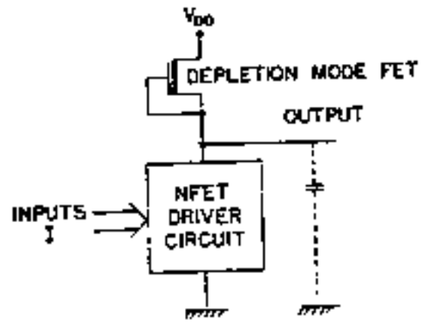


Figure 5:
General diagram of a NMOS gate

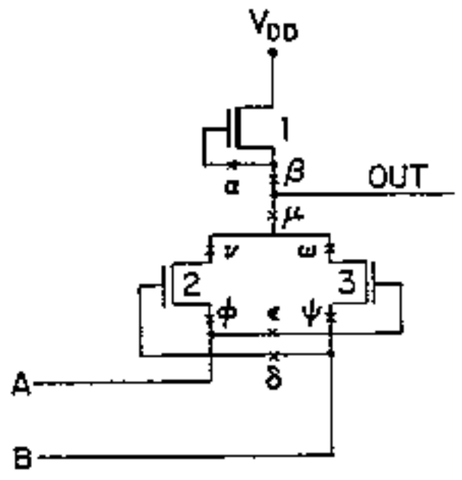
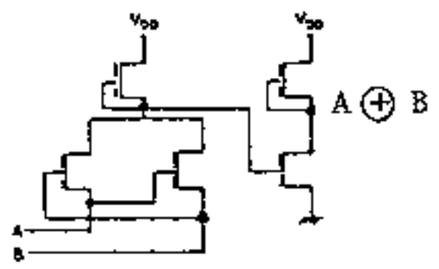
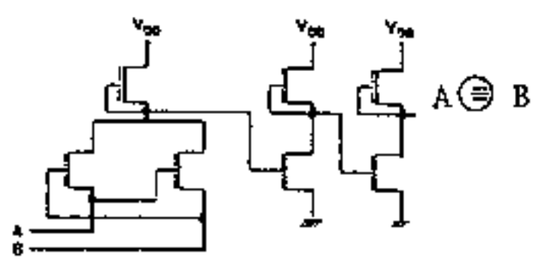


Figure 6(a): A NMOS ENOR gate



(i) EOR gate that can be cascaded



(ii) ENOR gate that can be cascaded

Figure 6(b):

EOR and ENOR gates derived from Figure 6(a) to generate parity trees.

TEST A	TEST B	FAULT 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 B or ε a δ or φ a ψ
0	0		X	X		X				X	X	X		
0	1	X			X				X	X			X	X
1	0	X					X	X			X		X	X

Figure 7:
Tests for the ENOR gate of Figure 6 (a)

1	1	0
0	0	0
S_1 = 1	S_2 = 0	S_3 = 1
0	0	0
0	1	1

(A) SET S_E

1	0	0
0	1	0
S_0 = 0	0	1
1	0	0

(B) SET S_H

FIGURE 8:
THE SEQUENCES TO DESIGN TESTS FOR PARITY TREES

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

FIGURE 9:
THE OUTPUTS OF A DECODER WITH DISABLE INPUTS