Optimum test patterns for parity networks

by D. C. BOSSEN, D. L. OSTAPKO and A. M. PATEL
IBM Laboratories
Poughkeepsie, New York

INTRODUCTION

The logic related to the error detecting and/or correcting circuitry of digital computers often contains portions which calculate the parity of a collection of bits. A tree structure composed of Exclusive-OR gates is used to perform this calculation. Similar to any other circuitry, the operation of this parity tree is subject to malfunctions. A procedure for testing malfunctions in a parity tree is presented in this report.

Two important assumptions are maintained throughout the paper. First, it is assumed that the parity tree is realized as an interconnection of Exclusive-OR gates whose internal structure is unknown or may differ. This requires that each gate in the network receive a complete functional test. Second, it is assumed that detection of single gate failures is desired.

Since each gate must be functionally tested, an m-input Exclusive-OR gate must receive $2^m$ input patterns. It will be shown that $2^m$ test patterns are also sufficient to test the network of any size, if $m$ is the maximum number of input lines to any Exclusive-OR gate. Hence, the procedure yields the minimum number of test patterns necessary to completely test the network for any single Exclusive-OR gate failure. It will also be shown, by example, that the procedure is fast and easy to apply, even for parity trees having a large number of inputs.

GATE AND NETWORK TESTABILITY

Since the approach is to test the network by testing every gate in the network, it is primarily necessary to discuss what constitutes a test for an individual Exclusive-OR gate. Although it is assumed that the parity trees are realized as a network of Exclusive-OR gates, no internal realization is assumed for the Exclusive-OR gates. Hence, it will be presumed that all $2^k$ input patterns are necessary to diagnose a single k-input Exclusive-OR gate. Each gate, therefore, is given a complete functional test so that single error detection means that any error in one Exclusive-OR gate can be detected. The following is the definition of a gate test.

Definition 1:

A test for a k-input Exclusive-OR gate is the set of $2^k$ distinct input patterns of length $k$. Figure 1 shows a three input Exclusive-OR gate, the $2^3=8$ input test patterns, and the output sequence which must result if a complete functional test is to be performed.

If the output sequence and the sequences applied to each input are considered separately, each will be a vector of length $2^k$. Thus, the Exclusive-OR gate can be considered to operate on input vectors while producing an output vector. Figure 2 shows a three input Exclusive-OR gate when it is considered as a vector processor. In terms of vectors, a test is defined as follows.

Definition 2:

A test for a k-input Exclusive-OR gate is a set of $k$ vectors of length $2^k$ which, when considered as $k$ sequences of length $2^k$, presents all $2^k$ distinct test patterns to the gate inputs.

Theorem 1:

If $K$ is a test for a $k$-input Exclusive-OR gate, then any set $M$, $M \subseteq K$, having $m$, $2 \leq m \leq k-1$, elements forms $2^{k-m}$ tests for an $m$-input Exclusive-OR gate.

Proof:

Consider the $k$ vectors in $K$ as sequences. Arrange the sequences as a $k$ by $2^k$ matrix in which the last $m$
rows are the sequences in $M$. Code each column as a binary number with the highest order bit at the top. Since the columns are all distinct according to definition 1, each of the numbers 0 through $2^k-1$ must appear exactly once. Considering just the bottom $m$ rows, it follows that each of the binary numbers 0 through $2^m-1$ must appear exactly $2^{k-m}$ times. Since each of the possible sequences of $m$ bits appears $2^{k-m}$ times, definition 1 implies that the set $M$ forms $2^{k-m}$ tests for an $m$-input Exclusive-OR gate.

Network testability:

Two conditions are necessary for a network of Exclusive-OR gates to be completely tested. First, each gate must receive a set of input vectors that forms a test. Second, any one gate error must be detectable at the network output. For the first condition it is necessary that the set of vectors from which the tests are taken be closed under the operation performed by the $k$-input Exclusive-OR gates. The second condition requires that any erroneous output vector produce an erroneous network output vector. The structure of this set of vectors and their generation will be discussed in the following sections.

AN EXAMPLE

The test pattern generation procedure is so simple and easy to apply that it will be presented by way of an example before the theoretical properties of the desired sequences are discussed. The algorithm proceeds by selecting an arbitrary output sequence and then successively determining input sequences which test each gate to produce the desired output.

Figure 3 presents the seven sequences and the associated addition table that will be used in the example. Figure 4 illustrates the gate labeling procedure which will be used to determine the inputs when the output is specified. Figure 5 shows the parity tree with 57 inputs and 30 Exclusive-OR gates of two and three inputs arranged in a four level tree. The procedure generates eight test patterns which will completely test all 30 gates of the tree.

The procedure is initiated by assigning an arbitrary sequence to the output of the tree. In the example, $W_4$ is selected as the final output sequence. Employing the 3-input gate labeling procedures shown in Figure 4, the inputs are determined to be $W_1, W_2,$ and $W_4$. With these three sequences, the gate is completely tested. These inputs are then traced back to the three gates in the third level. Using the gate labeling procedure again, the inputs for the gates from left to right are $W_3, W_5; W_3, W_5;$ and $W_3, W_5$. The sequences assigned to the inputs can be determined quickly and easily by making use of tracing and labeling. Under proper operation, each gate is completely tested and a single gate failure will produce an incorrect sequence.

**Figure 1—Three input Exclusive-OR gate with test patterns**

**Figure 2—Three input Exclusive-OR gate as a vector processor**

**Figure 3—Test sequences and their addition table**

**Figure 4—Gate labeling procedures**

WHERE $a = 00010111, b = 00101011, c = 01001101, d = 01110001$

NOTE: $W_i = W_i (\text{MOD } 7)$
at the output. Above each input the required sequence is listed, and the correct output is the sequence \( W_0 \). The test patterns are obtained by reading across the sequences and noting the correct output. The test is completed by adding the all zero test pattern. This should produce a zero output.

**THEORETICAL PRELIMINARIES**

Consider the set of vectors generated by taking all mod-2 linear combinations of the \( k \) vectors of a given test set \( K \). This set is obviously closed under mod-2 vector addition. In a parity check tree network an input of any subset of vectors from this set will produce vectors in the set at all input-output nodes of the Exclusive-OR gates. Some further insight can be gained by viewing the above set as a binary group code. The generator matrix \( G \) of this code, whose rows are \( k \) vectors from \( K \), contains all possible \( k \)-tuples as columns. If we delete the column of all 0's in \( G \), the resulting code is known as a MacDonald code in which the vector length \( n \) is \( 2^k - 1 \) and the minimum distance \( d \) is \( 2^{k-1} \). The cyclic form of the MacDonald code is the code generated by a maximum length shift register.2

**Theorem 2:**

Any independent set of \( k \) vectors from the Maximum Length Shift Register Code of length \( 2^k - 1 \) forms a test set for a \( k \)-input Exclusive-OR gate, excepting the pattern of all 0's.

**Proof:**

Any independent set of \( k \)-vectors from the code forms a generator of the code. In the Maximum Length Shift Register Code as well as in the MacDonald Code, \( 2d-n=1 \). This implies that any generator matrix of the code contains one column of each non-zero type. By definition 2, this forms the test for a \( k \)-input Exclusive-OR gate excepting the test pattern of all 0's.

**Corollary:**

For an \( m \)-input gate, \( m \leq k \), any set of \( m \)-vectors from a MLSRC of length \( 2^k - 1 \) forms a sufficient test.

The proof follows from Theorems 1 and 2. The maximum length shift register sequences can be generated by using a primitive polynomial \( p(X) \) of degree \( k \) in GF (2). Let \( g(X) = (X^n+1)/p(X) \) where \( n=2^k-1 \). Then the first vector \( W_0 \) of the MLSRC is the binary vector obtained by concatenating \( k-1 \) zeros to the sequence of the coefficients of \( g(X) \). The vectors \( W_1, W_2, \ldots, W_{2^k-2} \) are then obtained by shifting \( W_1 \) cyclically to the right by one digit for \( 2^k-2 \) times. The method is illustrated for \( k=3 \). A primitive polynomial of degree 3 in GF (2) can be obtained from tables, e.g., \( X^3+X+1 \) is primitive.

\[ g(X) = (X^3+1)/(X+1+1) = X^2+X+1. \]

Then \( W_0 \) is obtained from \( g(X) \) as

\[ W_0 = 1 0 1 1 1 0 0 \]

The sequences \( W_1, W_2, \ldots, W_6 \) are obtained by shifting \( W_0 \) cyclically as,

\[ W_1 = 1 0 1 1 1 1 0 \]
\[ W_2 = 0 0 1 0 1 1 1 \]
\[ W_3 = 1 0 0 1 0 1 1 \]
\[ W_4 = 1 1 0 0 1 0 1 \]
\[ W_5 = 1 1 1 0 1 0 1 \]
\[ W_6 = 0 1 1 0 0 1 1 \]

Note that when \( W_{2^k-2} \) is shifted cyclically to the right by 1 digit, the resulting vector is \( W_0 \). For the purpose of uniformity of relationship among the vectors we...
introduce the notation: \( W_i = W_{i \mod 2^k - 1} \). Now the following theorem gives a method of selecting independent vectors from a MLSRC.

**Theorem 3:**

The vectors \( W_i, W_{i+1}, \ldots, W_{i+k-1} \) in a MLSRC of length \( 2^k - 1 \) form an independent set.

**Proof:**

Suppose \( g(X) \) is given by \( g(X) = g_rX^r + g_{r-1}X^{r-1} + \ldots + g_0X + g_0 \), where \( r = (2^k - 1) - k \). Then the set of vector \( W_0, W_1, \ldots, W_{k-1} \) are given by

\[
W_0 = g_r \quad g_{r-1} \quad \cdots \quad g_0 \quad 0 \quad 0 \quad 0 \quad 0
\]

\[
W_1 = 0 \quad g_r \quad g_{r-1} \quad \cdots \quad g_0 \quad 0 \quad 0 \quad 0
\]

\[
W_2 = 0 \quad 0 \quad g_r \quad \cdots \quad \cdots \quad g_0 \quad 0 \quad 0
\]

\[
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots
\]

\[
W_{k-1} = 0 \quad 0 \quad 0 \quad \cdots \quad g_r \quad \cdots \quad g_0
\]

Clearly they are linearly independent. Because of the cyclic relationship, this implies that \( W_i, W_{i+1}, \ldots, W_{i+k-1} \) are independent.

**Corollary:**

The vectors \( W_{i+1}, W_{i+2}, \ldots, W_{i+m-1} \) and \( W_i \oplus W_{i+1} \oplus \ldots \oplus W_{i+m-1} \) \((m \leq k)\) form an independent set. With this as a test to an \( m \)-input Ex-OR gate, the correct output vector is \( W_i \).

As a direct consequence of the above theorems we have the following algorithm for the test pattern generation for a given Exclusive-OR network.

**Algorithm for test pattern generation:**

It is assumed that the Exclusive-OR network is constructed in the form of a tree by connecting \( m \)-input Ex-OR gates where \( m \) may be any number such that \( m \leq k \).

1. Select any vector \( W_i \) from a MLSRC of length \( 2^k - 1 \) as the output of the network.
2. Label the inputs to the last Ex-OR as \( W_{i+1}, W_{i+2}, \ldots, W_{i+m-1} \), and \( W_i \oplus W_{i+1} \oplus \ldots \oplus W_{i+m-1} \).
3. Trace each of the above inputs back to the driving gate with the same vector. Repeat steps (2) and (3) to determine the proper inputs to the corresponding gates.
4. The vectors at the input lines to the Ex-OR tree are then the test input vectors with the correct output as \( W_i \).
5. An additional all 0 pattern as input to the network with 0 as correct output completes the test.

It is easy to see that the test patterns generated by the above algorithm provide a complete test for each Ex-OR gate in the parity check tree. Furthermore, any single gate failure will generate an erroneous word which will propagate to the output. This is due to the linearity of an Ex-OR gate. Suppose one of its inputs is the sequence \( W_i \) with a corresponding correct output sequence \( W_i \). If the input \( W_i \) is changed by an error vector to \( W_i + e \), then the corresponding output is \( W_i + e \). Clearly, the error will appear superimposed on the observed network output.

**TEST MECHANIZATION**

We have shown that the necessary test patterns for a parity tree can be determined by a simple procedure using a set of \( k \) independent vectors or code words \( W_0, W_1, \ldots, W_{k-1} \) from a MLSRC as the input to each gate of \( k \) inputs. The result of applying this procedure to a network is an input sequence \( W_i \) for each network input and each network output. Testing is accomplished by applying the determined sequences simultaneously to each input and then comparing the expected network outputs with the observed network outputs.

Let the gate having the greatest number of inputs in the network show \( k \) inputs. The entire test can be mechanized using a single \((2^k - 1)\)-stage feedback shift register. To do this a unique property of the MLSR codes is used. From this property it follows that the entire set of non-zero code words is given by the

---

**Figure 6—Shift register for generating test patterns**

From the collection of the Computer History Museum (www.computerhistory.org)
2^k - 2 cyclic shifts of any non-zero code word together with the code word itself.

If a (2^k-1)-stage shift register is loaded with a particular code word \( W_0 \) as in Figure 6, then the sequence of bits observed at position 1 during 2^k - 1 shifts of the register is the code word \( W_0 \). Similarly for every other position \( i \), a different code word \( W_{i-1} \) is observed, so that the entire set of 2^k - 1 sequences is available. Since the correct output of the network is one of the code words, it is also available at one of the stage outputs for comparison. The general test configuration is given by Figure 7.

**SELF-CHECKING PARITY TREE**

Let us suppose that the test sequences and the shift register connections for a parity network have been determined as in Figure 7. A modification of this mechanization can be used to produce a self-testing parity network under its normal operation. The key idea is to monitor the normal randomly (assumed) occurring inputs to the network and to compare them with the present outputs of the shift register. When and only when a match occurs, the comparison of the outputs of the parity networks with the appropriate code words is used to indicate either correct or incorrect operation, and the shift register is shifted once. This brings a new test pattern for comparison with the normal inputs. Every 2^k - 1 shifts of the register means that a complete test for all single failures has been performed on the network.

The mechanization of the self-checking parity tree is shown in Figure 8. The inputs to the AND gate \( A_{w_0} \) are the set of input lines of the parity tree which receive the test sequence \( W_i \). The inputs to the AND gates \( A_{w_{i-1}} \) are the inverse of the input lines of the parity tree which receive the test sequence \( W_i \).

An alternate approach to self-checking is to use the testing circuit of Figure 7 as a permanent part of the parity tree. The testing is performed on a time-sharing or periodic basis while the circuit is not used in its normal mode. This is easily accomplished by having the clock, which controls the shift register, gated by a signal which indicates the parity tree is not being used. This could be a major portion of the memory cycle when the parity tree under consideration is used for memory ECC.

**CONCLUSION**

We have shown that a very low and predictable number of test patterns are necessary and sufficient for the complete testing of a parity tree under the single failure assumption. The required tests are easily and rapidly determined by an algorithm which is presented. (An application of this technique is also given for a self-checking parity tree.) Since the effect of the input test patterns is a complete functional test of each gate, the tests are independent of any particular failure mode.
REFERENCES

1 J E MacDONALD
   Design methods for maximum minimum-distance error correcting codes
   IBM J of R & D Vol 4 pp 43-47 1960

2 W W PETERSON
   Error correcting codes
   MIT Press Cambridge Massachusetts 1961

3 A M PATEL
   Maximal group codes with specified minimum distance
   IBM J of R&D Vol 14 pp 434-443 1970