VLSI DESIGN
2001, Vol. 12, No. 4, pp. 563–578
Reprints available directly from the publisher
Photocopying permitted by license only

BIST Analysis of an Embedded Memory Associated Logic*

JACOB SAVIR

ECE Dept., New Jersey Institute of Technology, University Heights, Newark, New Jersey 07102-1982

(Received 15 August 1999; In final form 11 September 2000)

Of late some interesting and useful work has been done on the problem of testing logic surrounding embedded memories. This work assumes that the logic surrounding the memory is functionally partitioned and that the different partitions are logically isolated one from the other.

This paper expands upon past work using a more flexible design rule which allows feed-forward connections between the data-path Prelogic and Postlogic. The connections are such that there is no feedback from the memory outputs to its inputs, and both the Prelogic and the Postlogic are disconnected from the Address and Control logic. Under this design rule we show the auxiliary circuits used to determine the random pattern testability of faults in the circuitry driving the address inputs and the controls of the two-port memory.

The techniques described herein are intended to be used in conjunction with the cutting algorithm for testability measurement in built-in self-test (BIST) designs [2, 11, 17], but may also be suitable for use with other detection probability tools [9, 19], and simulation tools [20].

Keywords: Signal probability; Detection probability; Exposure probability; Markov chain; Random patterns

1. INTRODUCTION

A two-port (Write and Read) memory is embedded within combinational logic as shown in Figure 1. The memory contains 2^M words of N bits each, and all 2^M addresses are distinct and accessible. The write and read address decoders have M inputs and 2^M outputs each, and are internal to the memory. The Write port is composed of N Data-In lines, M inputs to the write address decoder, and a Write Control line. The Read port consists of the N Data-Out lines, M read address decoder inputs, and a Read Control line. The primary inputs marked OUT1 in

* This work was supported by NJCST under the Center for Embedded System on a Chip Design.

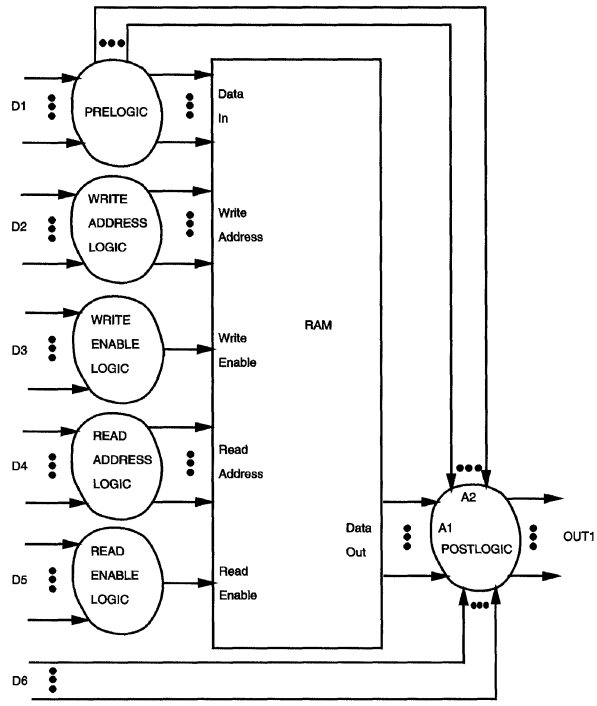

FIGURE 1 Two-port embedded memory with feed-forward connections between Prelogic and Postlogic.

Figure 1 are the only observation points in the structure.

The circuits surrounding the memory are to be tested by applying random patterns to the primary inputs and observing the responses on the primary outputs of the embedding logic. The embedding logic is purely combinational. It consists of a Prelogic driving the Data-In lines of the Write port, a Postlogic whose A1 inputs are fed by the memory and whose A2 inputs are connected directly to the Prelogic, two groups of Address logic feeding the Write and Read port address decoder inputs, and two groups of Control logic driving the port Enable line inputs. The Address and Control logic groups are disconnected from each other and from the Prelogic and Postlogic.

The first work dealing with random pattern testability of logic surrounding memories was reported in [4, 5]. In this earlier work an analytical method called TRIM (for Testability Range by Ignoring the Memory) was described for computing the exposure probabilities of faults in the combinational logic of the data-path of the memory (logic which either feeds the Data-In lines of the memory or is fed by the Data-Out lines). In this work no direct connection exists between the Prelogic and the Postlogic.

Work related to diagnosis of faults in circuits surrounding embedded memories was recently reported in [12, 14, 16]. Here also, no connections were allowed between the Prelogic and the Postlogic.

References [13, 15] describe methods of analyzing the testability of the Control logic under the design rule such that no connections exist between the Prelogic, the Postlogic, the Write and Read Address logic, or the Write or Read Enable logic except through the memory. Reference [18] describes the detection probability propagation rules of address line faults in an embedded memory with no feed-forward connections.

The present paper relaxes this rule by allowing the feed-forward connections between the Prelogic

and Postlogic as shown in Figure 1. Feed-forward connections may be found in modern microprocessor chips [1, 6, 7, 10, 21] for improving performance. With this feature, when the array is written with certain data, the same data is also available in the "same cycle" at the array output to a potential read operation intending to read this new data, thus saving one cycle.

With this relaxed design rule, a new set of methods are described which are suitable for computing the probability of random pattern detection of stuck-at faults in the memory address and control circuitry.

Section 2 briefly describes the assumed behavior of the memory Address and Control lines, and indicates the effects of faults in their upstream circuitry. Section 3 recapitulates the basic testability procedures used in past work, and Section 4 lists for convenience some of the definitions of the fault-induced memory behaviors which were given in [8, 13, 15] and adds some new definitions relating to address circuitry faults. Section 5 describes the auxiliary circuits needed to compute detection probabilities for faults in the Write Address circuitry when feed-forward connections exist between the Prelogic and the Postlogic. Section 6 does the same for Read Address faults. Section 7 gives the analysis method for Write Enable faults, and Section 8 does the same for Read Enable faults. Section 9 describes a method of reducing the complexity of the auxiliary circuits using line flagging technique. The method may be useful in connection with the cutting algorithm [2, 11, 17]. Section 10 describes some experimental results, and Section 11 concludes with a brief summary.

2. MEMORY CONTROLS

Write and/or read operations on the memory are initiated through the logic driving the Enable or clock lines of the Write and Read ports. Let these lines be called the *Enable lines* of the ports, with operational characteristics as follows:

Write Enable: When the Write Enable line is on, the word on the memory Data-In lines is written to the location programmed by the address on the Write port decoder inputs. (When the Write Enable line is off, no operation ensues).

Read Enable: When the Read Enable line is on, the word stored at the location accessed by the address on the Read port decoder inputs appears on the memory Data-Out lines. (When the Read Enable line is off, the Data-Out lines take on either W_0, the all 0's word, or W_1, the all 1's word, depending upon the hardware implementation of the Data-Out circuitry).

Suppose a stuck-at fault f_i exists in the logic driving one or the other of the Enable lines. Whenever an input pattern simultaneously sensitizes f_i and propagates its effect to the memory boundary, an *enable error* occurs on the line. If f_i is associated with the memory Read port, the enable error will either inhibit a desired read or cause an unwanted read. Similarly, if the Write port is affected, either a desired write operation will not occur or an undesired write will occur.

A random input vector applied to the primary inputs of the embedding logic may cause a simultaneous write and read operations. It is possible for both the Write and Read ports to select the same address. In this event, we assume that the word is first written to the common address and then is read out. We call this event a *write-through* operation, because for this vector the memory effectively does not "exist". The values on the Data-Out lines are precisely those on the Data-In during this event, and the memory appears to be short-circuited. We have to distinguish between write-through and non-write-through operations because each requires a different auxiliary circuit. While we assume that write-through operations are possible, it should be noted that the method of analysis described in the following can generally be modified to account for different memory behaviors.

If a fault exists in the combinational logic driving the address decoder inputs of either of the ports and a test pattern simultaneously sensitizes the fault and

propagates its effect to the inputs, an *address error* occurs on the decoder inputs. If the fault is associated with an active Write port, the word on the Data-In lines will be written to the wrong address. Conversely, if the fault affects an active Read port, an incorrect address will be read.

We will show the artifacts necessary to determine the random pattern testability of both enable errors and address errors, assuming feed-forward connections between the Prelogic and Postlogic.

3. RECAPITULATION

It was shown in [13, 15] that the exposure probability of a fault in the control circuitry of an embedded memory without feed-around logic could be determined by first computing the signal probability p_e at the output of a virtual auxiliary structure, similar to the one shown in Figure 2, consisting of various replications of the Prelogic and the Postlogic driving a "difference" engine composed of N 2-input Exclusive-Ors, each driving a common Or circuit. The replicated logics represent different epochs[1] of the test history of the original circuit and its embedded memory, and therefore the input patterns to the replications are statistically independent. The output of the auxiliary circuit assumes the value 1 whenever a difference can be observed between the signals on the outputs of the replicated logics, and therefore the signal probability p_e of the output is the probability of detecting that difference. The exposure probability Q_i of a fault f_i in the Control logic during a test consisting of a sequence of L random input vectors is then

$$Q_i(K) = 1 - (1 - p_e)^K \tag{1}$$

where K ($K \leq L$) is the number of times that the epochs represented by the auxiliary circuitry occur during the test.

[1] The word *epoch* is used to refer to points on the time axis.

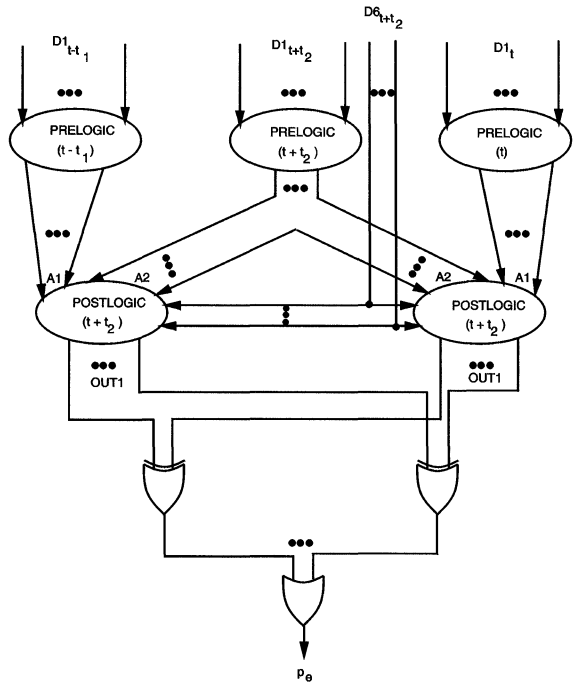

FIGURE 2 Auxiliary structure to compute p_e for write or read address faults or write enable faults under non-write-through memory operations.

The auxiliary circuits for control circuitry faults were shown in [13, 15], with methods for determining their K values, in cases for which no connections exist between the logic structures around the memory.

We now expand upon previous derivations to show the auxiliary circuits for use when connections are permitted between the Prelogic and the Postlogic. We are also extending the discussion to address circuitry faults.

4. NOTATIONS AND DEFINITIONS

Some of the terms below were defined in [13, 15]. We collect the definitions here for ease of reference, since they will be used in the sequel. New definitions are also added to ease the discussion on address circuitry faults.

4.1. Terms Which Refer to the Detection of Address Circuitry Faults

Tainted Write: A memory write to an address Z is a *tainted write* if either

1. address Z should have been selected but wasn't because the fault affected the address lines, or
2. address Z should not have been selected but was because of the fault.

Address Z is said to be tainted by the fault. Note, however, that a tainted write to an address is detectable only if it results in an error word at the address, if the error is later read out, and if a path is sensitized (during the read) through the Post-logic to the primary outputs. Notice also that taint can be removed by a correct write (one in which the fault is inactive) to a tainted address.

Fuzzy Read: A *fuzzy read* is a memory read from an address which has not been previously read and has had no intervening correct writes since the most recent tainted write to the address.

Notice that fuzzy reads are a subset of the total number of read operations on the memory.

Misdirected Read: A *misdirected read* is any read operation which accesses the wrong address.

4.2. Terms Which Refer to the Detection of Enable Circuitry Faults

Distorted Write: *A distorted write* occurs if either

1. some location should have been written but wasn't because the fault affected the Write Enable line, or
2. some location should not have been written but was because the fault struck the Write Enable line.

A distorted write may or may not create an error word, but even if it does we must read out the error word to detect the fault.

Warped Read: A *warped read* is a memory read from an address which has not been previously read and has had no intervening correct writes since the most recent distorted write to the address.

Warped reads are a subset of all of the read operations on the memory. Each warped read has a certain probability of detecting the write enable circuitry fault.

Distorted Read: A distorted read occurs if either

1. the word at location A should have been read out but wasn't because the fault affected the Read Enable line, or
2. the word at location A should not have been read out but was because the fault struck the Read Enable line.

5. ANALYSIS OF A WRITE ADDRESS CIRCUITRY FAULT

Assume that a solid stuck-at fault f_i exists in the combinational logic driving the address decoder inputs of the Write port for the embedded memory shown in Figure 1, and let the rest of the structure be fault-free.

We first focus our attention on the detection of the write address circuitry fault under the assumption that no write-through memory operations occur. We will later consider write-through operations and mixtures of the two.

5.1. No Write-through Operations

Consider a test vector which is applied at some epoch t and which causes a write operation to an arbitrary memory address A in the fault-free machine. If fault f_i is active at the decoder inputs of the faulty machine during vector t, the address word will be changed from A to some other word B. The fault causes a write to B of the data word which should have gone to A. We say that both addresses A and B have experienced a tainted write (see the definition in Section 4).

Assume some later test vector $t+t_2$ causes the first read of address A after the tainted write and further assume there were no intervening correct writes to A. This read operation is called a fuzzy

read, fuzzy because it is not certain that the read will detect the fault (see definition in Section 4). (Address B was also affected at epoch t by the fault. The first read of B without intervening correct writes is also fuzzy). Because the memory and its read circuitry is fault-free, the fuzzy read unloads the word at address A normally. But because of the tainted write at epoch t, the word unloaded is that which was written into A at some epoch prior to t, say at $t-t_1$. The time line shown in Figure 3 (which proceeds from left to write) illustrates the three-epoch sequence.

The probability of detecting f_i during the fuzzy read at epoch $t+t_2$ is the probability that a difference can be discerned at the primary outputs between the word written to A at $t-t_1$ and the word which should have been written to A at t. The auxiliary circuit suitable for computing this probability is shown in Figure 2.

The Prelogic marked "$t-t_1$" in Figure 2 generated the word written to address A at epoch $t-t_1$ from the random D1 inputs applied at that time. Similarly, the Prelogic marked "t" created the word which should have been written to A at epoch t. At epoch $t+t_2$ we read out either the word from epoch $t-t_1$ under the fault condition, or the word from epoch t without the fault. By creating two copies of the Postlogic as they existed during epoch $t+t_2$, as shown in Figure 2, and driving their A1 inputs with the epoch $t-t_1$ and the epoch t Prelogics, we may observe at their respective OUT1 outputs a difference between the faulty and the fault-free conditions. Notice, however, that the A2 inputs of both Prelogics must be driven in parallel by a single copy of the Prelogic existing at epoch $t+t_2$. The probability of observing a difference at the two OUT1 outputs between the faulty and the fault-free machines is just the signal probability p_e on the output of the difference circuit shown in Figure 2. In computing the value of p_e note that the inputs D1 to the three Prelogic copies are mutually independent by virtue of their occurrence at three different epochs.

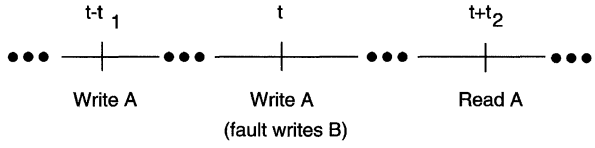

FIGURE 3 Three-epoch time line of a tainted write and fuzzy read during a write address fault under non-write-through operations.

Let K be the number of fuzzy reads which occur during the test sequence. Note that K is a function of the actual fault under analysis [13, 15]. Given K for a particular fault f_i, the lower bound probability Q_i of detecting f_i, on the assumption that there are no write-through operations, is given by

$$Q_i(K) = 1 - (1 - p_e)^K \tag{2}$$

It is important to note that p_e need be computed only once by using the structure of Figure 2, but that K must be computed for each fault. $Q_i(K)$ in Eq. (2) is a lower bound since the K fuzzy reads are just a subset of the total read operations on the memory. A second (non-fuzzy) read of a tainted address may contribute to the exposure probability of f_i, but is not considered in Eq. (2) because of the difficulty of computing its contribution.

5.2. Write-through Operations

On a vector which causes a write-through operation, the word on the memory Data-In lines appears instantly at the Data-Out lines (only for that vector, of course). The write-through occurs because the same address A is simultaneously selected by both the write and read address circuitry. If our assumed fault affects the write address decoder inputs during this vector, the port will write to some address other than A. The simultaneous read then unloads address A correctly, but observes a word which was actually written to A at some earlier vector. The time line for write-through operations is shown in Figure 4.

The appropriate auxiliary structure for write-through operations is given in Figure 5. Notice that the only difference between Figure 5 and Figure 2 is that the Prelogic driving the A2 inputs

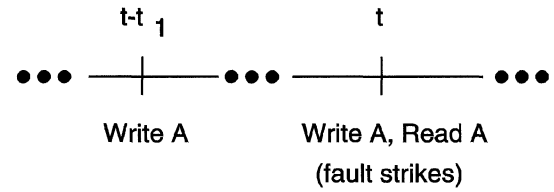

FIGURE 4 Time line during a write-through operation of a write address fault.

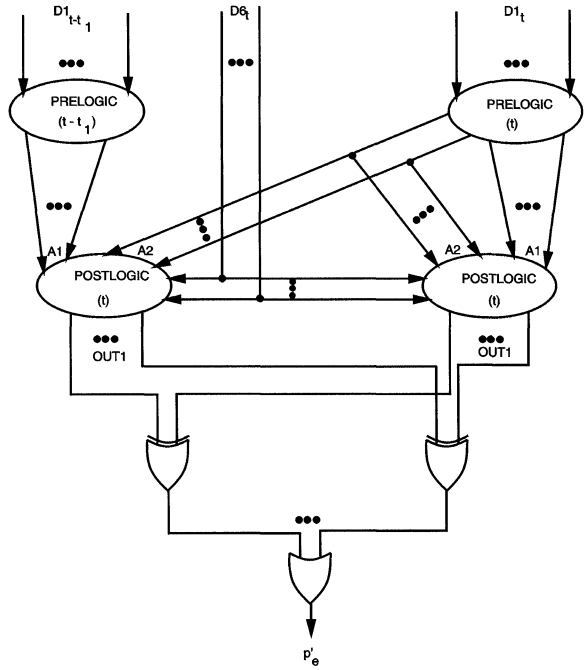

FIGURE 5 Auxiliary structure to compute p'_e for write or read address faults or write enable faults under write-through operations.

of the Postlogic copies in Figure 5 exists at epoch t. The signal probability p'_e computed from Figure 5 is the probability that a fuzzy read during a write-through operation will detect the write address circuitry fault. The number of fuzzy reads K' during write-through operations can be computed in a fashion similar to that used for computing K.

Suppose in a test sequence of length L there occurs a mixture of fuzzy reads, some during write-through and some during non-write-through operations. Then the lower bound on the probability of detecting the write address circuitry fault f_i is given by

$$Q_i(K, K') = 1 - (1 - p_e)^K (1 - p'_e)^{K'} \qquad (3)$$

Note that for large memories (*e.g.*, $M \geq 4$) write-through operations are statistically infrequent. For such memories it may be acceptable to ignore the possibility of write-through fuzzy reads, and to rely upon Eq. (2) alone.

6. ANALYSIS OF A READ ADDRESS CIRCUITRY FAULT

Let some fault f_j be present in the combinational circuitry driving the inputs to the read address decoder of the memory shown in Figure 1, and assume that the rest of the structure is fault-free. Let the test vector applied at epoch t select an arbitrary address A in the fault-free machine. Assume that test vector t in the faulty machine sensitizes f_j and propagates its effect to the decoder inputs. Then in the faulty machine some address other than A, say address B, will be unloaded to the Data-Out lines. Detection of f_j occurs if the word at B is different from that at A and if the effects of such difference can be propagated through the Postlogic to one or more primary outputs.

6.1. No Write-through Operations

Let us first consider a non-write-through operation. To focus our attention we will use the time line shown in Figure 6. (The sequence of write A followed by write B on the time line can be reversed without affecting the following results).

At $t+t_2$ the fault strikes and the word at B is read instead of the word at A. This is a misdirected

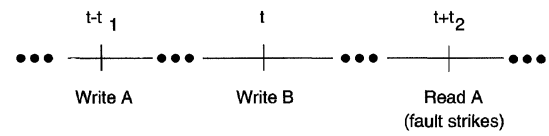

FIGURE 6 Three-epoch time line of a non-write-through operation during a read address fault.

read (see the definition in Section 4). During this misdirected read, the word unloaded to the Data-Out lines is from B instead of A. Obviously we want to know whether we can distinguish between the two words at the primary outputs.

From the time line we see that at $t+t_2$ we want to compare the words written at $t-t_1$ and t. Except for the nature of the fault, this is exactly what the auxiliary structure of Figure 2 does. Hence it is easy to see that the signal probability p_e on the output of Figure 2 is exactly the probability of detecting the read address circuitry fault on one misdirected read. If R is the number of (non-write-through) misdirected reads in the test sequence, then the exposure probability of the read address circuitry fault f_j is

$$Q_j(R) = 1 - (1 - p_e)^R \tag{4}$$

R may be computed using the methods discussed in [13, 15].

6.2. Write-through Operations

The similarity between write and read address faults and the use of Figure 2 for non-write-through operations carries over into the case of write-through and the use of Figure 5. The read address circuitry fault time line for write-through operations is shown in Figure 7.

At epoch t we want to compare the word just written to A with the word read from B because of the fault. This is just what auxiliary circuit in Figure 5 does. Hence p'_e from Figure 5 is the probability of detecting fault f_j from a single misdirected read under write-through operation. If

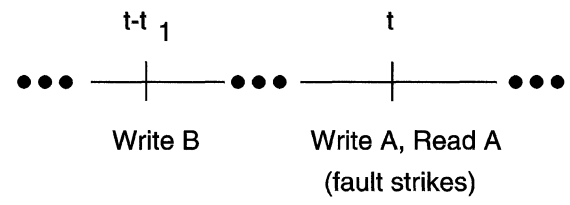

FIGURE 7 Time line during a write-through operation of a read address fault.

the total number of misdirected reads during the test sequence consists of R non-write-through misdirected reads, and R' write-through misdirected reads, the exposure probability of a read address circuit fault f_j is

$$Q_j(R, R') = 1 - (1 - p_e)^R (1 - p'_e)^{R'} \tag{5}$$

Notice that Q_j is an exact exposure probability for f_j since no read operations other than misdirected reads can contribute any information about the fault.

Since write-through operations are infrequent for large memories under random stimuli, R' is small and Eq. (5) is closely approximated by Eq. (4).

7. ANALYSIS OF A WRITE PORT ENABLE CIRCUITRY FAULT

We now consider faults in the memory control or enable circuitry. References [13, 15] described the auxiliary circuits for a memory without feed-around connections. The circuits used where Prelogic to Postlogic connections exist are described in this and the following section for Write and Read port enable faults.

Assume some stuck-at type fault f_i in the combinational circuitry driving the Enable Line of the Write port of the memory shown in Figure 1, and further assume the remainder of the structure, including the memory, is fault-free. During an input vector for which the fault affects the Enable Line input to the memory, one of two things will happen: either the Write port should be on but is incorrectly turned off, or it should be off and is incorrectly turned on. We call either occurrence a distorted write (see definition in Section 4). Notice that a distorted write affects only one address.

It was shown earlier that different auxiliary circuits were used for the same address fault, depending upon whether or not a write-through operation was occurring when the fault hit the address. When we consider Enable Line faults,

however, a curious situation can occur. The fault can actually cause a write-through when none is intended, or conversely can inhibit an intended write-through. Consider, for example, a test vector during which the fault affects the Write port Enable Line. For this vector we define six possible behaviors, listed below along with their transforms under the Write Enable Line fault.

Case	Fault-free	Fault behavior
1.	no operation	write
2.	write	no operation
3.	read	write and read different addresses
3a.	read	write and read same address
4.	write and read different addresses	read
4a.	write and read same address	read

Cases 1, 2, 3, and 4 cannot be detected during the epoch in which they occur, and are not write-through events. Cases 3a and 4a, on the other hand, are write-throughs in the sense that Case 3a will cause an unintended write-through and Case 4a will inhibit an intended write-through. We will treat these two sets of cases separately.

7.1. Cases 1, 2, 3, and 4 (No Write-through)

In Cases 1 and 2 the Read port is off when the fault strikes, and in Cases 3 and 4 the Read port is on but selecting some address other than that on the Write port. There are four time lines associated with detection of the fault, and they are shown in Figure 8. A and B are two arbitrary but different addresses. The fault strikes the time line at epoch t.

Notice that the top line of Figure 8 also represents Cases 1 and 3 in which the Write port should be off at epoch t. If the vector applied at t causes f_i to affect the Write Enable input, the port is incorrectly turned on. Since address A is on the write decoder inputs at epoch t, f_i causes the correct word at A to be overwritten by the word on the Data-In lines.

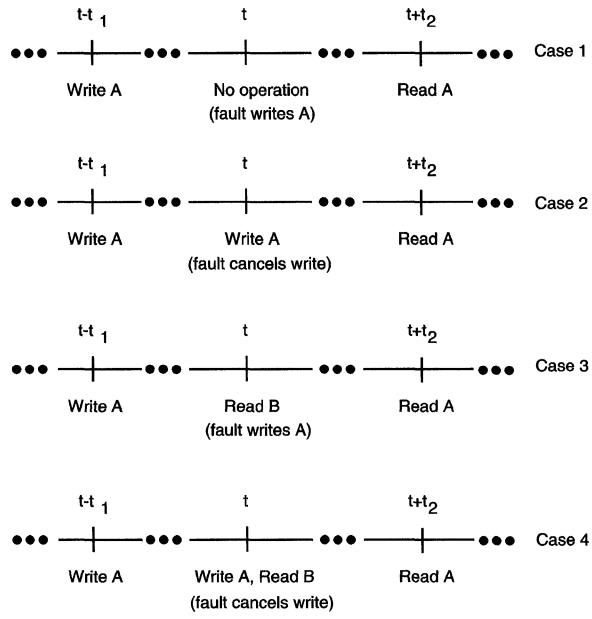

FIGURE 8 Time lines for the four cases of a write enable fault under non-write-through operations.

Notice that the bottom line of Figure 8 also represents Cases 2 and 4 where the Write port should be on at epoch t. When f_i affects the Write Enable at t, it turns the port off, and the word which should have been written to A is not.

The event at epoch t on both time lines is a distorted write. Note that a distorted write can either eliminate a desired write operation to some address, or cause an undesired write. A single occurrence of a distorted write has the possibility of causing a single incorrect word in memory. Now assume that the vector applied at epoch $t+t_2$ causes the first read of A since the distorted write, and further assume that there were no intervening correct writes to A. Then there is a possibility of detecting f_i at $t+t_2$. We refer to this read operation as a warped read (see definition in Section 4). Warped reads are a subset of the read operations on the memory, and each warped read has a certain probability of detecting the write enable fault.

Determination of the probability of detection of f_i during the warped read at $t+t_2$ is simplified if we notice the essential similarity between the two time lines for the fault. For Cases 1 and 3 we want to

compute the probability of detecting a difference between the word W_{t-t_1} which should be stored in A and the word W_t which was inadvertently written to A by f_i at epoch t. And for Cases 2 and 4 the desired probability of detecting a difference between the word W_t which should be stored in A and the word W_{t-t_1} which is actually in A because f_i inhibited the write operation at t. These two events are statistically similar in that we want the probability of detecting the difference between two words generated by the Prelogic at two different epochs. But this is exactly the situation reflected by the auxiliary structure shown in Figure 2. We conclude that the signal probability p_e on the output of Figure 2 is exactly the probability of detection of the Write port enable circuitry fault from a single warped read. Thus the exposure probability of the fault during a test sequence in which S warped reads occur is

$$Q_i(S) = 1 - (1 - p_e)^S \tag{6}$$

Again we note that p_e is a constant for a particular memory but that S must be computed for each fault in the enable circuitry. Methods for computing S are given in [13, 15]. Q_i in Eq. (6) is a lower bound since a second (and subsequent) read of a distorted write address may contribute to the detection of f_i.

7.2. Cases 3a and 4a (Pseudo Write-through)

In both of these cases, the Read port is on during an input vector during which the fault affects the Write port Enable line, and the same address is selected by both ports. The time lines associated with these two cases are shown in Figure 9.

The top line in Figure 9 refers to Case 3a in which the Write port is supposed to be off at vector t during which the fault strikes. The fault turns on the Write port and "writes-through" the word currently on the Data-In lines. The read operation during this vector then "reads" the write-through word instead of the word written at $t-t_1$.

Similarly, the bottom line of Figure 9 refers to Case 4a. At vector t the memory should be doing a write-through, but because the fault inhibits the

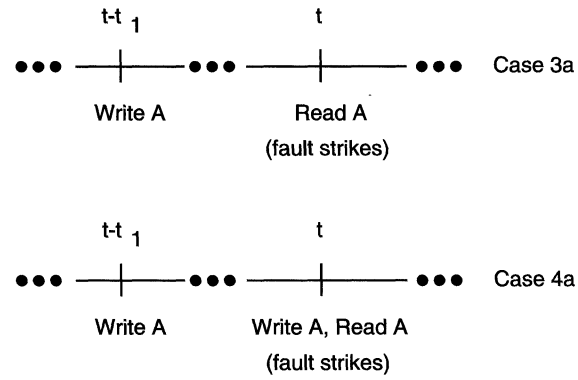

FIGURE 9 Time lines for Cases 3a and 4a of a write enable fault.

Write port the word unloaded to the Data-Outs is the word written at $t-t_1$ instead of the (correct) write-through word.

For both events, we want to know the probability of detecting a difference between a write-through word and a word which was written at some earlier epoch. Consideration of the auxiliary structure shown in Figure 5 clearly shows that it computes this required probability p'_e on its output.

If we let S' be the number of warped reads which occur under Cases 3a and 4a, then the exposure probability of the Write port enable circuitry fault f_j under a mixture of all possible cases is

$$Q_i(S, S') = 1 - (1 - p_e)^S (1 - p'_e)^{S'} \tag{7}$$

We notice that for reasonably large memories the probability of encountering a Case 3a or 4a situation is low (since there is a small probability that both the Write and the Read ports randomly select the same address). Hence for large memories we may ignore these "write-through" enable fault events.

The computation of S and S' may be done by simulation or with a suitable Markov chain [13, 15].

8. ANALYSIS OF A READ PORT ENABLE CIRCUITRY FAULT

Assume a solid fault f_j in the Read port enable circuitry, and let the remainder of the structure be

fault-free. A test vector which propagates the effect of f_j to the Enable line input of the memory will either cause an unintended read or will inhibit a desired read. We call either event a distorted read (see the definition given in Section 4).

An enable fault on the Read port has a slightly different behavior pattern than any of the previous faults we have considered. Recall that in Section 2 we assumed that a fixed word appeared on the memory Data-Out lines when the read enable was turned off. The actual value of the word depends upon the implementation of the Data-Out circuitry. In the subsequent analysis of Read port enable faults, let W_0 be the actual word on the memory Data-Out lines when the Read port is turned off. A convenient mental image is to think of W_0 as the all 0's word, but it is important that the actual "non-read" word for the specific memory under analysis be used for W_0 in the auxiliary circuits to be described.

Just as with the earlier write enable analysis, a distorted read can cause an unintended write-through operation, or can destroy an intended write-through. The following table shows the six cases of fault-free operation and the corresponding operations when the fault affects the Read port Enable.

Case	Fault-free	Fault behavior
1.	no operation	read
2.	read	no operation
3.	write	write and read different addresses
3a.	write	write and read same address
4.	write and read different addresses	write
4a.	write and read same address	write

(Note that this table is *not* the same one shown in Section 7 on write enable faults).

Note that Cases 1, 2, 3, and 4 behave like non-write-through operations and therefore can be analyzed together. Cases 3a and 4a can also be considered together since they both have aspects of write-through operation in either their fault-free or their distorted read behaviors.

8.1. Cases 1, 2, 3, and 4 (No Write-through)

The time line for Cases 1, 2, 3 and 4 are shown in Figure 10, where A and B are arbitrary but different addresses in memory.

In all cases a distorted read occurs at epoch t. For Cases 1 and 3, W_0 is the expected output word but instead the fault causes a read of the word at A. For Cases 2 and 4 the word at address A is expected, but the fault cancels the read so the actual output word is W_0.

In all four cases, detection will occur if a difference can be observed at the primary outputs between the word W_0 and the word from address A on the Data-Outs. Figure 11 shows the appropriate auxiliary structure for these four cases. It is easily seen that the signal probability p_d from Figure 11 is

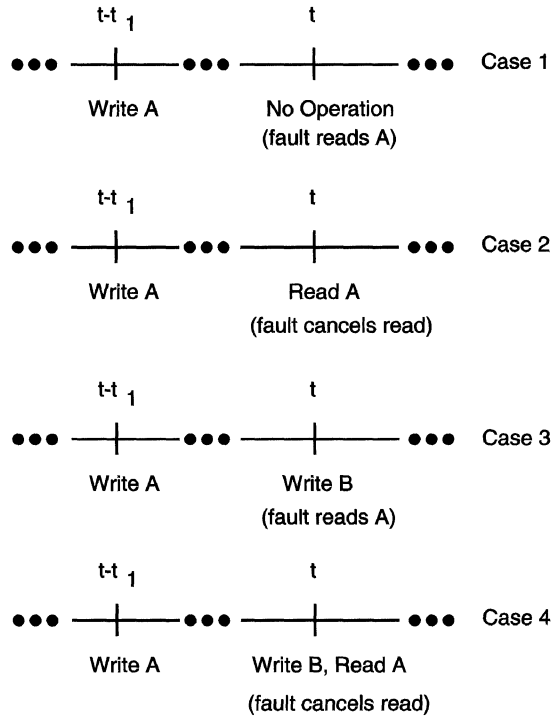

FIGURE 10 Time lines for the four cases of a read enable fault under non-write-through operations.

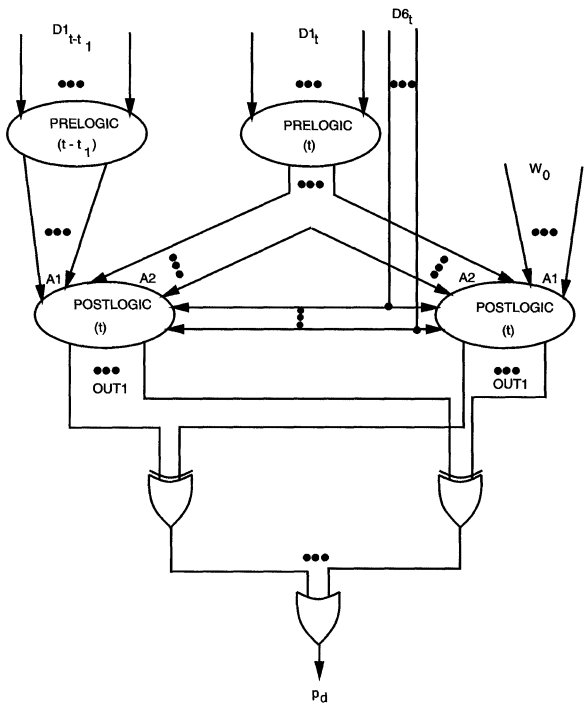

FIGURE 11 Auxiliary structure to compute p_d for read enable faults under non-write-through memory operations.

the probability of detecting the read enable fault on one distorted read occurring from any of these cases. Given that V distorted reads occur in the non-write-through mode, the exposure probability of the read enable fault f_j is

$$Q_j(V) = 1 - (1 - p_d)^V \tag{8}$$

8.2. Cases 3a and 4a (Pseudo Write-through)

There are two time lines for these cases as shown in Figure 12. Notice that, because all write operations are fault-free, both lines involve only a single epoch.

For Case 3a, a write but not a read should occur at epoch t, so W_0 is expected on Data-Out lines. The fault causes a distorted read at epoch t which in turn causes a write-through so that the actual Data-Out word is the word written to address A.

In Case 4a a write-through operation is expected at epoch t. The distorted read inhibits the read

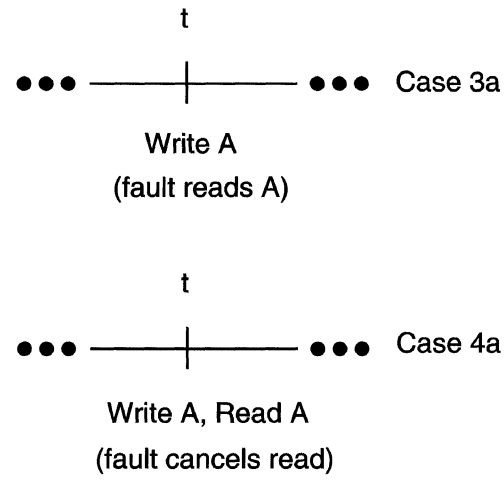

FIGURE 12 Time lines for read enable fault under pseudo write-through.

portion of the write-through, and causes W_0 to appear on the Data-Outs instead of the written-through word.

In both cases, we want to know the probability of detecting the difference between W_0 and the

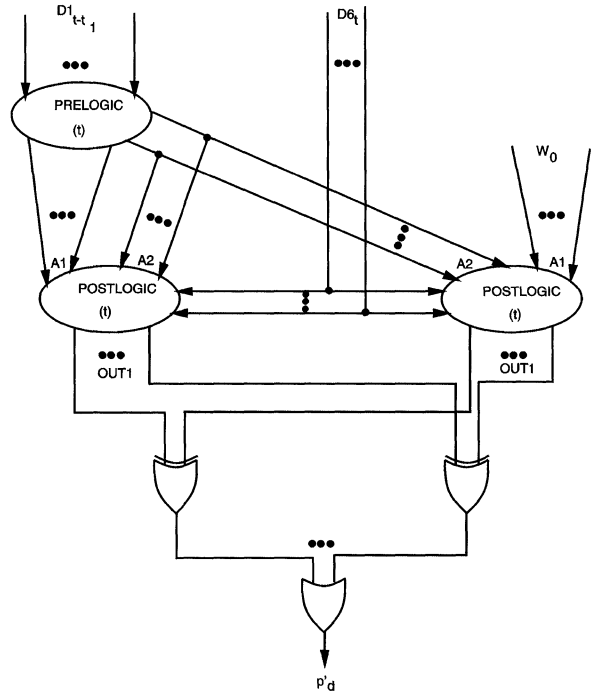

FIGURE 13 Auxiliary structure to compute p'_d for read enable faults under write-through memory operations.

word written to A at epoch t. The auxiliary structure of Figure 13 computes this probability as its output signal probability p'_d.

Let V' be the number of distorted reads that occur during the test sequence in the write-through mode. If there are also V distorted reads that occur in non-write-through mode, the exposure probability of the read enable fault f_j for this mixture of distorted reads is

$$Q_j(V, V') = 1 - (1 - p_d)^V (1 - p'_d)^{V'} \qquad (9)$$

9. AUXILIARY CIRCUIT REDUCTION USING FLAGS

This section discusses a topic which is not crucial to the analysis, but will serve to simplify its implementation in applications of the cutting algorithm [2, 11, 17].

The auxiliary circuit signal probabilities required by the preceding analysis are computed on a software model of the circuits. Since all of the auxiliary circuits require the replication of either prelogics or postlogics or both, the size of the model may possibly exceed the available storage. Furthermore, computation time grows significantly as the model size increases. For both reasons, then, it is useful to consider techniques for reducing the degree of replication in the auxiliary circuits. If the cutting algorithm is used to compute the signal probabilities, a suitable technique is suggested here.

When the cutting algorithm encounters a fanout node which re-converges at some down-stream circuit it cuts all except one of the fanout branches and assigns the probability range (0, 1) to each of the cut branches (this is the *full-range cutting algorithm*). The uncut branch continues the signal probability of the stem of the fanout. This cutting turns the structure into a tree, thus permitting use of the relatively simple tree equations for computing signal probabilities [2, 17].

Consider now the side prelogic replicas (marked $t - t_1$ and t) in the auxiliary circuit of Figure 2. Their D1 inputs, and hence their A1 outputs, are independent by virtue of their occurrence at two different epochs. Imagine that these two prelogics are removed and replaced by a single prelogic whose A1 outputs fan out to the two postlogics. These fanouts would now re-converge at Exclusive-Ors of the difference engine. The cutting algorithm, when running against this modified circuit, would (incorrectly) force a cut on one branch of each of the A1 fanouts. Suppose now that the replacement of a single prelogic has its A1 outputs appropriately flagged so that the cutting algorithm *does not* cut any of the fanout branches. As far as the subsequent operation of the algorithm, then, the signals on both branches of each A1 fanout are considered independent. Specifically, each pair of A1 inputs to the

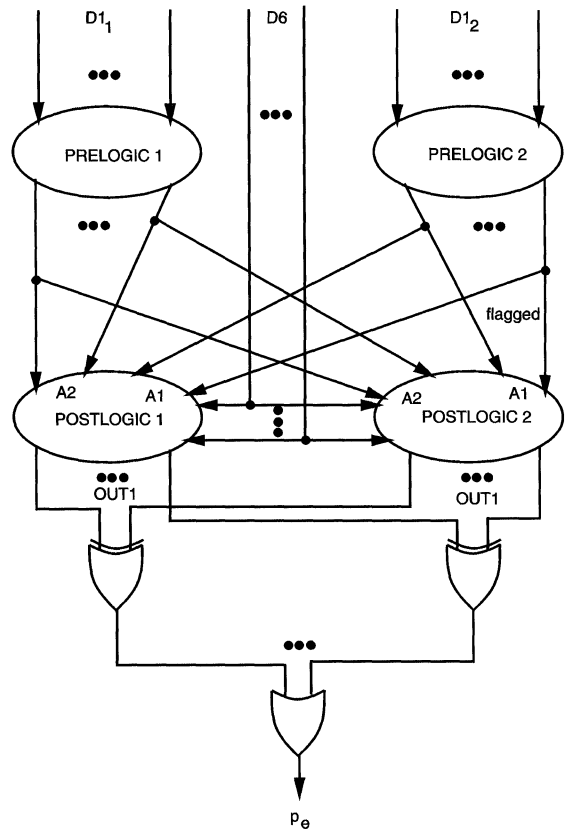

FIGURE 14 Simplified auxiliary structure derived from Figure 2.

two postlogics would be considered independent. But this is precisely why the two side prelogic replicas of Figure 2 were employed, to insure pairwise independence between the corresponding A1 inputs to the postlogics. Hence, by modifications to the cutting algorithm, and by appropriate flagging of fanout branches, we can replace the two prelogic copies by one. This modification of Figure 2 is shown in Figure 14 in which the flagged A1 outputs are marked.

Similarly, the auxiliary circuit of Figure 5 can be simplified to its flagged equivalent shown in Figure 15. The auxiliary circuits for enable faults shown in Figures 11 and 13 cannot be significantly simplified.

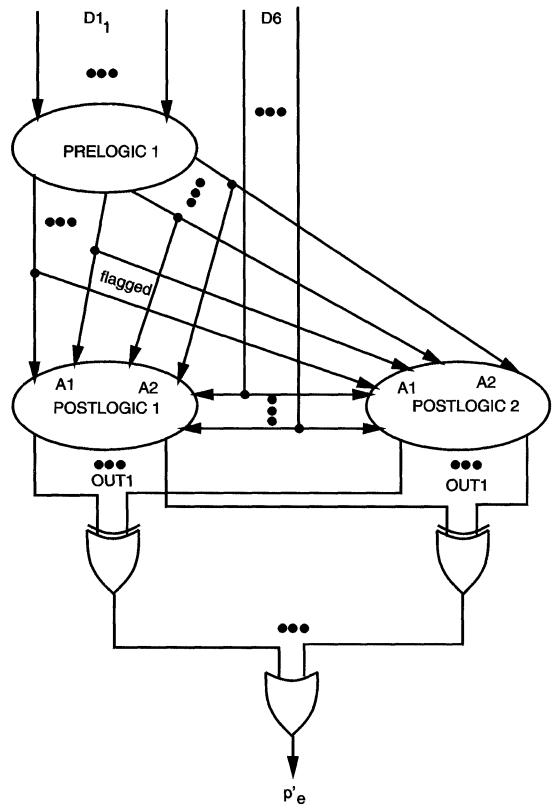

FIGURE 15 Simplified auxiliary structure derived from Figure 5.

10. EXPERIMENTAL RESULTS

In order to test the correctness of all our auxiliary structures we have taken the entire set of ISCAS'85 circuits [3] and modified them by embedding a two-port memory in them at the netlist level. We have added the necessary controls, and verified that all modified circuits have feed-forward connections between their Prelogic and Postlogic. The embedded memories ranged in size from 4 words by 8 bits, to 1024 words by 32 bits.

In order to verify the correctness of our auxiliary structure it was necessary to compare the detection probability curves as predicted by our equations to those measured by simulation. To this end we have injected 100 random single stuck-at faults in each circuit. For each injected fault we have divided the test length of 100,000 patterns to 100 windows of length 1,000. At the end of each window we have measured the average cumulative detection probability over 30 different sets of random stimuli. Every such average constituted a point on the experimental detection probability curve.

To create the theoretical curve we have computed all the statistical parameters as suggested by this paper. We then used the formulas we have developed to compute the theoretical cumulative detection probability after each 1,000 vector window.

For the cases were our formulas predicted exact values there was a complete match between the theory and the measured values. For cases were our formulas only predicted lower bounds, the theoretical curve was below the measured curve (within 5% distance).[2]

[2]Lower bounds come in very handy in computing a **conservative** test length to detect the fault in question. A conservative test length figure **guarantees** the targeted test quality (unlike an upper bound that does not guarantee it).

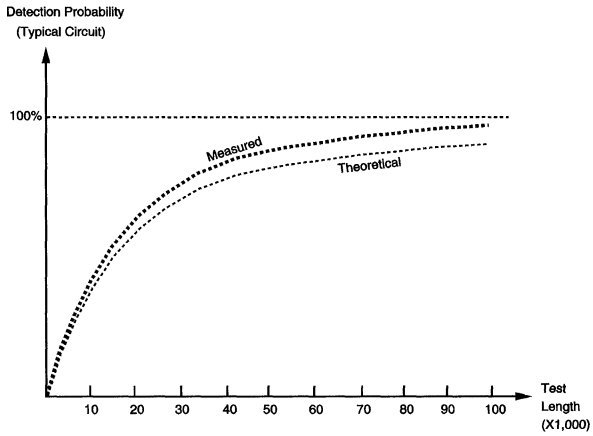

FIGURE 16 Detection probability of a typical ISCAS'85 circuit.

A typical comparison between a theoretical and experimental curve is shown in Figure 16.

11. CONCLUSIONS

We have described a method of computing the random pattern testability of stuck-at type faults in the address and control circuitry of an embedded two-port which has feed-forward connections between its data path Prelogic and Postlogic.

The first step is to determine the single-vector error probability of detecting the fault using the appropriate auxiliary circuit. There are four of these single-vector error detection probabilities: p_e and p'_e for the probabilities of detection of write address, read address, and write enable faults, for non-write-through and write-through behaviors, respectively; and p_d and p'_d for read enable faults under the same two behaviors. These detection probabilities need be computed just once for each memory under analysis.

The next step is to compute the number of times that each fault creates an error vector. This has to be done for each fault separately, and may be done most conveniently by using an appropriate Markov chain as demonstrated in [13, 15] (but may also be done using simulation).

The final step is to compute the exposure probability Q_i of the fault f_i using the equation

$$Q_i(T) = 1 - (1 - p_m)^T \tag{10}$$

where p_m is the appropriate detection probability for the specific memory and T is the number of typed vectors which occur during the test.

We have also shown means for reducing the complexity of the auxiliary circuits by appropriate flagging of the prelogic outputs.

The analysis is specifically designed for use with the cutting algorithm in assessing the testability of faults for built-in self-test. It may, however, be useful in connection with other methods of computing fault exposure probabilities.

References

[1] Bardell, P. H. and Lapointe, M. J., Production experience with built-in self-test in the IBM ES/9000 system. In: *Proc. Int. Test Conf.*, pp. 28–36, October, 1991.

[2] Bardell, P. H., McAnney, W. H. and Savir, J., *Built-In Test for* VLSI: *Pseudorandom Techniques*. Wiley Interscience, New York, 1987.

[3] Brglez, F. and Fujiwara, H., Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN. In: *Special session on ATPG and fault simulation, Proc. IEEE Int. Symp. Circuits and Systems*, pp. 663–698, June, 1985.

[4] Carter, L., Huisman, L. M. and Williams, T. W., TRIM: Testability range by ignoring the memory. In: *Proc. Int. Test Conf.*, pp. 474–479, September, 1986.

[5] Carter, L., Huisman, L. M. and Williams, T. W., TRIM: Testability range by ignoring the memory. *IEEE Trans. Computer-Aided Design*, **7**, 38–49, January, 1988.

[6] Paul Reed *et al.*, A 250 MHz 5W RISC microprocessor with on-chip L2 cache controller. In: *Proc. IEEE International Solid-State Circuits Conference*, pp. 412–413, Feb., 1997.

[7] Hunter, C., Slaton, J., Eno, J., Jessani, R. and Dietz, C. (1994). PowerPC 603**(TM) microprocessor: an array built-in self test. In: *Proc. Int. Test Conf.*, pp. 388–394.

[8] McAnney, W. H., Savir, J. and Vecchio, S., Random pattern testing for data-line faults in an embedded multiport memory. In: *Proc. Int. Test Conf.*, pp. 100–105, November, 1985.

[9] Parker, K. P. and McCluskey, E. J. (1975). Probabilistic treatment of general combinational networks. *IEEE Trans. Computers*, **C-24**(6), 668–670.

[10] Raina, R., *PowerPC Design Center, personal communication*, Aug., 1998.

[11] Savir, J., Improved Cutting Algorithm. *IBM J. Res. Dev.*, **34**(2/3), 381–388, March/May, 1990.

[12] Savir, J., BIST-based fault diagnosis in the presence of embedded memories. In: *Proc. Int'l Conf. on Comp. Des.*, pp. 37–47, Oct., 1997.
[13] Savir, J., Random pattern testability of memory control logic. *In Proc. VLSI Test Symp.*, pp. 399–407, Apr., 1997.
[14] Savir, J., Salvaging test windows in BIST diagnostics. In: *Proc. VLSI Test Symp.*, pp. 416–425, April, 1997.
[15] Savir, J., Random pattern testability of memory control logic. *IEEE Trans. Computers*, **47**(3), 305–312, March, 1998.
[16] Savir, J., Salvaging test windows in BIST diagnostics. *IEEE Trans. Computers*, **47**(4), 486–491, Apr., 1998.
[17] Savir, J., Ditlow, G. S. and Bardell, P. H., Random Pattern Testability. *IEEE Trans. Computers*, **C-33**(1), 79–90, Jan., 1984.
[18] Savir, J., McAnney, W. H. and Vecchio, S., Random pattern testing for address-line faults in an embedded multiport memory. In: *Proc. Int. Test Conf.*, pp. 106–114, November, 1985.
[19] Seth, S. C. and Agrawal, V. D., PREDICT-probabilistic estimation of digital circuit testability. In: *Proc. Fault-Tolerant Computing Symposium*, pp. 220–225, June, 1985.
[20] Waicukauski, J. A., Gupta, V. P. and Patel, S. T., Diagnosis of BIST Failures by PPSFP Simulation. In: *Proc. Int. Test Conf.*, pp. 480–484, 1987.
[21] Yen, J. T., Sullivan, M., Montemayor, C., Pete, W. and Evers, R., Overview of PowerPC 620 multiprocessor verification strategy. In: *Proc. Int. Test Conf.*, pp. 167–174, October, 1995.

Author's Biography

Dr. Savir holds a B.Sc. and an M.Sc. degree in Electrical Engineering from the Technion, Israel Institute of Technology, and an M.S. in Statistics and a Ph.D. in Electrical Engineering from Stanford University. He is currently a Distinguished Professor at New Jersey Institute of Technology, where he held the position of Director of computer engineering (1996–2000), and Newark College of Engineering Associate Dean for research (1999–2000). Previously with IBM, Dr. Savir was a Senior Engineer/Scientist at the IBM PowerPC Development Center in Austin, TX; at IBM Micro electronics Division in Hudson Valley Research Park; at IBM Enterprise Systems in Poughkeepsie, NY, and a Research Staff Member at the IBM T. J. Watson Research Center, Yorktown Heights, N.Y. He was also an Adjunct Professor of Computer Science and Information Systems at Pace University, N.Y, and SUNY Purchase, N.Y.

Dr. Savir's research interests lie primarily in the testing field, where he has published numerous papers and coauthor-ed the text "Built-In Test for VLSI: Pseudorandom Techniques" (Wiley, 1987). Other research interests include design automation, design verification, design for testability, statistical methods in design and test, fault simulation, fault diagnosis, and manufacturing quality.

Dr. Savir has received four IBM Invention Achievement Awards, six IBM Publication Achievement Awards, and four IBM Patent Application Awards. He is a member of Sigma Xi, and a fellow of the Institute of Electrical and Electronics Engineers.