Chasing Subtle Embedded RAM Defects for Nanometer Technologies

Theo Powell
Texas Instruments Inc.
PO Box 660199, MS 8645
Dallas, TX 75226-0199

Amrendra Kumar, Joseph Rayhawk, and Nilanjan Mukherjee
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070

Abstract

A design's increasing density, as well as its number of embedded memories increases its vulnerability to a variety of potential manufacturing defects. Standard March test algorithms used for obtaining good defect coverage must be augmented by new algorithms that target defects not screened by embedded BIST controllers. This paper presents our experience diagnosing address decode open faults (ADOF) using scan patterns. Subsequently, tests were added in the BIST controller to target ADOF. Other tests were added to screen potential bit/byte write-enable faults in memories with bit/byte write-enable controls.

1 Introduction

The importance of complete screening of embedded memory defects has always been high, but is increased by the move to nanometer technology. Geometries of memory cells are getting smaller, whereas, the area occupied by embedded memories in silicon is increasing. Consequently, memories are vulnerable to a variety of manufacturing defects and may require new test algorithms to be effective for screening such defects. Traditional March algorithms must be augmented by others to catch defects which would otherwise escape, thereby impacting test quality.

2 Detecting Address Decoder Open Faults

One algorithm which is well documented [1, 2, 3, 4, 5, 6] is the one for targeting address decoder open faults (ADOF). For NAND-based address decoders, these defects typically occur on the PMOS transistors behaving as a pull-up tree with one end connected to the supply voltage, V_{dd}, with the other end connected to the output of the CMOS gate. They change the combinational decoder circuit to behave as a sequential circuit and therefore cannot be detected by standard March tests. Occasionally, these defects also change the delay behavior of the circuit and therefore need to be tested at functional speed. Other open defects in the NMOS transistors affect the discharge path of the CMOS gates. They can be detected by a single pattern and are therefore covered by standard March tests [1].

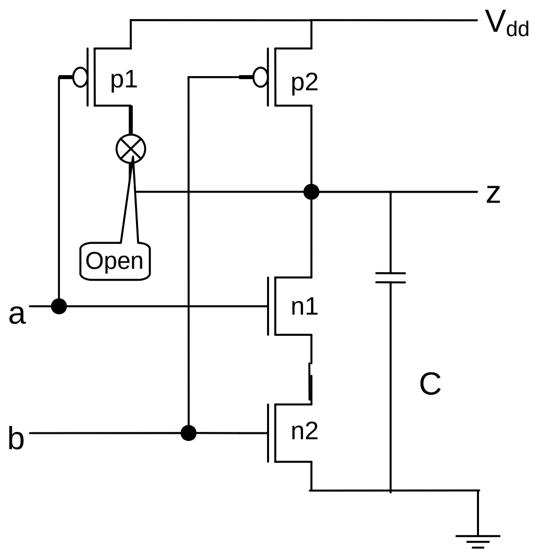

Figure 1: 2-Input CMOS NAND Gate

Figure 1 shows a 2-input CMOS gate for a NAND gate, which is used in the address decoder. If the defective PMOS transistor *p1* is open, it might not be detected unless the gate output is discharged with a previous pattern. To insure this discharge a two pattern test is required.

Table 1: Behavior of a Regular NAND Gate

a	b	p1	p2	n1	n2	Z
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Table 1 describes the state of all the transistors for a 2-input NAND gate without any ADOF. Table

2 on the other hand, describes the behavior of the NAND gate with a ADOF in transistor *p1*.

Table 2: Behavior of NAND Gate with an Open PMOS Transistor

a	b	p1	p2	n1	n2	Z
0	0	ON	ON	OFF	OFF	1
0	**1**	**OFF**	**OFF**	**OFF**	**ON**	$\mathbf{Z_{old}}$
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Referring back to Figure 1, which shows an open fault in the PMOS transistor for input *a,* the patterns that can detect all stuck-at faults for the two input gate are:

a	b	Z
1	1	0
0	1	1
1	0	1

However, this set of patterns will not screen the ADOF for input *a* since the floating output in Figure 1 will retain its 1 state even after applying a 0 input to *b* due to the output capacitance *C*. Detection of this fault requires the output line Z to be discharged with the first pattern,"1 1". Therefore, the set of patterns to detect both stuck-at faults and ADOF are as follows.

a	b	Z
1	1	0
0	1	1
1	1	0
1	0	1

Each address line uses Boolean gates to select the individual row and column of the cell, and so are susceptible to ADOF.

Other possible failure mechanisms can also require a similar consecutive two pattern sequence for screening. All are speed-sensitive and require at-speed application of the tests. Examples of such defects are as follows - a) the pre-charge sense amp is not working correctly, and b) an unbalanced set of bit and ~bit lines at the transfer gate of column decode multiplexer.

Klaus [3] pointed out in his paper discussing DRAMs that algorithms that screen all the potential ADOF without analysis of the layout will be screened by all pairs of address sequences that are one Hamming distant from each other. The analysis is equally applicable to embedded SRAMs. Using a March algorithm does have an advantage for reducing the test time for screening ADOF defects. Otterstedt [4] showed how a LFSR address generator can be used to apply the required pattern pair to screen ADOF. The LFSR needs to be a primitive polynomial of the address size to ensure that all addresses are covered. Application of the embedded memory BIST controllers often call for one controller testing several memories of different address sizes. This makes it preferable to have the address increment or decrement by a fixed increment. It eliminates the need to have multiple address counters in the same controller for the different address sizes.

Diagnosis of memory defects would also be more difficult with LFSR counters. LFSR also make it more difficult distinguishing ADOF from other memory defects. ADOF are usually not repairable, whereas other types are likely repairable. The distinction between the two classes of faults can be important for repairable RAMs.

Embedded memory BIST is an effective way to provide the tests for embedded RAMs. Powell [7] presented a method for producing memory BIST with algorithms at compile time. In this case, the generated memory BIST has one address counter which can increment or decrement in a linear fashion. Consequently, it is unable to apply the two pattern tests required to screen ADOF. For tight geometries, the ADOF are more likely to occur. In order to lower the Defective Parts per Million (DPPM) using memory BIST, the ADOF algorithm must be included in the BIST controller.

Most of the papers [4, 5, 6] published earlier on ADOF consider them to be stuck-at faults. Azimane [8] considered the potential of resistive PMOS Open conditions using simulation of resistance inserted at the transistors. For such defects, the pattern pair must be applied at functional speed to screen the defect, since slower frequencies allow the resistive defects to escape.

3 Diagnosis with MacroTest

Designs with embedded memories using BIST can sometimes have memory failures detected by functional applications but missed by the memory BIST. In these situations, the defect type needs to be identified and additional tests need to be prepared to assure that the defects are screened from future delivered parts.

In one such case, the employed embedded memory BIST tool included a single address counter that permitted addressing either by incrementing by one, by a column, or by a fixed index factor. The pattern pairs needed to screen the ADOF could not be applied, making these defects good candidates to not be screened. A defect returned to the factory was thought to be a ADOF. The identified failure location was known to be at a specific address. A scan approach for applying patterns using MacroTest was employed to confirm that our deductions were correct.

MacroTest [9-10] is a tool that is often used to translate patterns to test a memory (or any other black-box embedded in a design) into scan patterns that can then be applied with other scan patterns targeted for random logic. There is no additional test logic nor any additional mechanisms required to deliver the patterns to the concerned memory. This is particularly useful for testing small memories, when they are performance critical in nature and BIST area overhead for a chip is a concern.

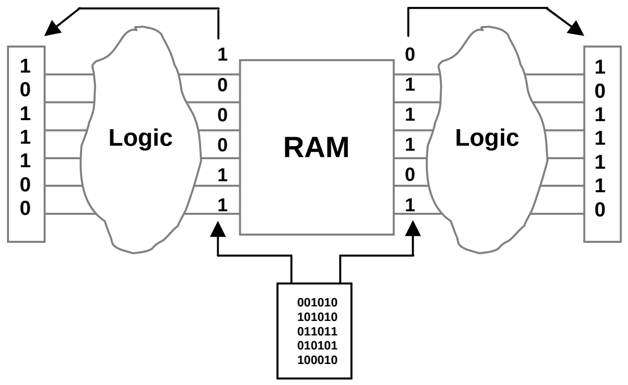

Figure 2: Overall Concept of MacroTest

With MacroTest, memories are treated as black-boxes, and therefore, all that is required by the tool is a file specifying the input stimuli, the expected output response, and the clocking mechanism. The tool takes the input and output values and converts them into corresponding scan patterns. Figure 2 illustrates the overall concept of MacroTest. The tool can be also used to generate at-speed patterns, in which case a sequence of patterns that need to be applied at-speed are specified. The tool then converts the sequence of patterns to a corresponding scan pattern having a depth of more than 1. Within TI memory testing, MacroTest was only used in verifying the at-speed algorithm for detecting the missing ADOF defects in the returned parts.

Additional information revealed that the memory did not fail when the test was applied at 20 MHz below the functional speed whereas it did fail at the functional operating speed. The applied test that screened the defect was a functional memory test which exercised the memory at the location of the defect. This information helped determine that the defect was not an open transistor but rather a resistive connection at the PMOS transistor. The required test must then be a transition test at functional speeds to be able to screen this defect.

A launch-off-capture (LOC) type transition test was employed with two patterns applied at-speed. The documented ADOF test [11] applies a write to a neighbor address followed by a read at the base address. The test applied using the scan approach used two read patterns instead. The scan patterns applied the first read of the RAM at the neighboring address followed by the second read at the base address. The consecutive reads were then used for the transition test.

The algorithm's first step loaded the entire RAM with 0's. A pattern of 1's was written into the base cell, the location at which the failure was detected. A two-read transition pattern was then applied with the first read at the neighboring address one Hamming distance away from the base cell. The second read was at the failed cell. A third functional pattern was applied to allow a scan chain capture the results of the second read prior to scan out. Both read patterns were provided to MacroTest so that a LOC transition test could be created via ATPG. The patterns were repeated for all addresses at a Hamming distance of one away from the failing address.

Figure 3 shows the sequence of clocks that were used to apply the transition scan test between the scan operations. The first vertical line is at the edge of the last scan shift pattern. The second vertical line is at the application of the first read of the neighbor Hamming address. The expected read value was all 0's; however, this result was not scanned out or compared. The third vertical line is at the capture by a scan register so that the string of 1's results could be scanned out.

The ATPG scan-based transition tests were applied to the device in question. The test screened the defect, thereby confirming on the tester that the defect screened was indeed an resistive ADOF.

Table 3: RAM Testing Coverage Measure

Algorithms	Column Rd R~d	Column Wd R~d	Row Rd R~d	Row Wd R~d	ADOF defects	Background pattern
Column March A	X	X				Checkerboard
Column March B	X	X				~Checkerboard
March 13N			X			0/F, 3/C, 0F/F0, 69/96
March 11N			X	X		5/A
ADOF					X	

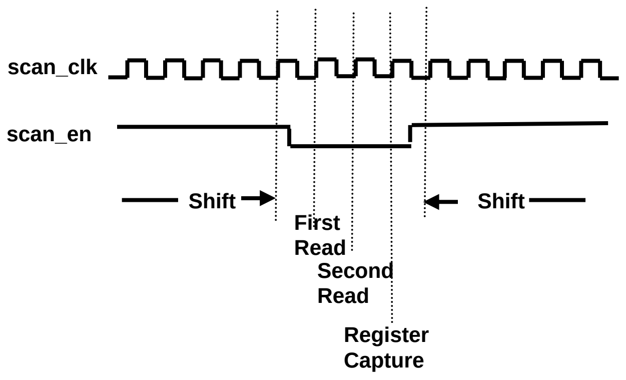

Figure 3: Scan Pattern Transition Test

The scan approach for LOC transition tests is an effective method for debugging the defects. This approach can be applied as long as the design supports the requirements for applying such LOC transition type tests. However, attempting to test all possible ADOFs in this fashion is prohibitive due to the excessive test time required for each pattern pair. Adding a test to the memory BIST controller was necessary for screening these types of defects in production.

4 Improving Defect Coverage for ADOF and Other Defects

The key for screening ADOF is an algorithm including the two pattern pairs one Hamming distance apart. [3] The Galloping pattern (GALPAT) test includes all such pairs and so will also screen ADOF. GALPAT is very costly in terms of test application time since the complexity is $O(N^2)$, *N* being the address size. In order to reduce the test time it is frequently limited to applying the GALROW test which tests for address decode defects across rows or the GALCOL test which tests for address decode defects across columns. The order for these tests are $4*N*R$ and $4*N*C$, where *R* is the number of rows, *C* is the number of columns, and N is the number of address cells. The test is applied twice; once with the background of 0's and the second time with the background of 1's. The complexity of the ADOF algorithm is $O(N*\log_2 N)$. Its advantage is that it covers both row and column address decoder defects, but is less test time than both tests.

Table 3 shows a covering measure first introduced by Powell [7]. The table identifies boundary conditions of the RAM for testing the read/write operations. Table 3 above includes one additional boundary condition, namely the ADOF.

To illustrate the boundary condition, consider the March 13N algorithm (also called Partial MOVI test in [1]) which has an X under Row Rd R~d. To see how this tests two consecutive reads of opposite data, the operation of two consecutive addresses need to be compared:

$$\begin{array}{c} |\leftarrow \text{Address A}i \rightarrow|\leftarrow \text{Address A}i{+}1\rightarrow| \\ \text{Rd W\textasciitilde d R\textasciitilde d Rd W\textasciitilde d R\textasciitilde d} \\ |\leftarrow \text{R\textasciitilde d Rd} \rightarrow| \end{array}$$

The addressing used during the March 13N algorithm is row specific since it is incremented/decremented by 1. With the March 13N, every row is checked with consecutive reads of opposite data. All the algorithms detect some unique faults, and therefore, each one of them in Table 3 is required. Some defects are only screened by algorithms similar to the ADOF algorithm but applied to the devices using an existing CPU-based BIST approach. Although the number of defects screened exclusively with ADOF defects is small, it indicates that an additional boundary condition for testing them is necessary.

The table keeps track of addressing by row and column increments for defect screening. If a LFSR address counter is used for the controller addressing as suggested by Klaus [3], keeping track of both row and column addressing can be difficult to check.

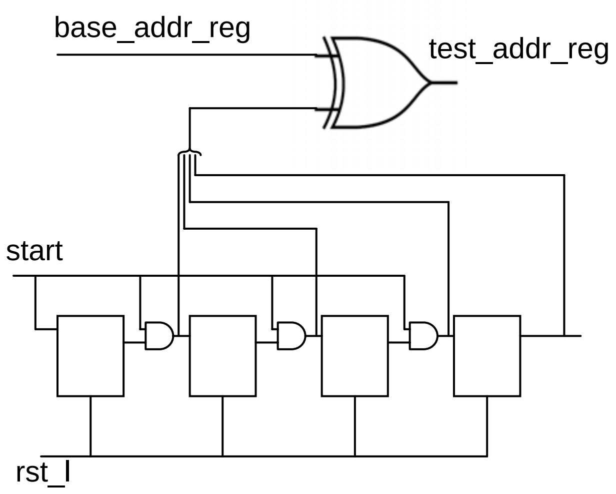

Figure 4: Schematic for Test Address Generator

5 Adding ADOF Screening to the BIST Controller

As described in Section 2, the ADOF algorithm is a series of two pattern vectors. The pseudo-code for the algorithm that is implemented by the BIST controller is as follows:

```
// ADOF Algorithm
    1. Initialize memory with data
    2. Write data at base address
    3. For all neighboring (test) addresses
        a. Write ~data at test address
        b. Read data at base address
    4. Write data at base address
```

Unlike March algorithms, the ADOF algorithm has two loops – the first one involving all addresses within the memory (a.k.a. the *base loop*), and the second involving all the neighboring addresses that are generated for every base address (a.k.a. *local loop*). The addressing scheme for the base loop is the same as any March algorithm and can be implemented using a conventional up/down counter. Whereas, the test address loops through all addresses that are a Hamming distance of one away from the base address. Instead of having another address counter and complex logic, the test address is generated using the following equation:

$$\text{test\_addr} = (1 << j) \text{ XOR } \text{base\_addr}; \; j \in 1,N$$

where, N is the width of the data word of the memory. In essence, the test address generation requires an additional shift register that can be stored with an initial value of 1, which can then be logically shifted left by one bit to obtain the test addresses. Figure 4 illustrates the circuit employed to calculate the test addresses for a given base address.

The notation for describing the ADOF algorithm is:

$\Uparrow$ indicates addressing in ascending order

Wd, Rd indicate writing value d or reading value d

$\Uparrow_H$ indicates accessing each address one Hamming distance away from base address

W_Hd indicates writing data, d at the address one Hamming distant away from base address.

( ) indicates looping within the entire address space.

The ADOF algorithm can thus be denoted by:

$\{ \Uparrow (W0); \Uparrow (W1, \Uparrow_H (W_H0, R1),W0) \}$

One of the other changes in the controller is related to the diagnostics monitor (referred to as data-logger in the remaining of the paper) that is employed to scan out failing information of the memory whenever the controller detects a failure. For March algorithms, this is accomplished by reporting the state of the controller including the memory number, port number, failing address, algorithm step, and the fail data actually read from the memory or the fail map value (which is the error vector). Whenever the ADOF test is executed, an additional address value (test address) can be reported in order to pinpoint the location of the failure. This is dependent on the targeted resolution during the data-logger operation. The data-logger can be modified to report the shift register value along with the base address register when a fault is detected. Consequently, one can easily determine which NAND/NOR gate in the address decoder is failing.

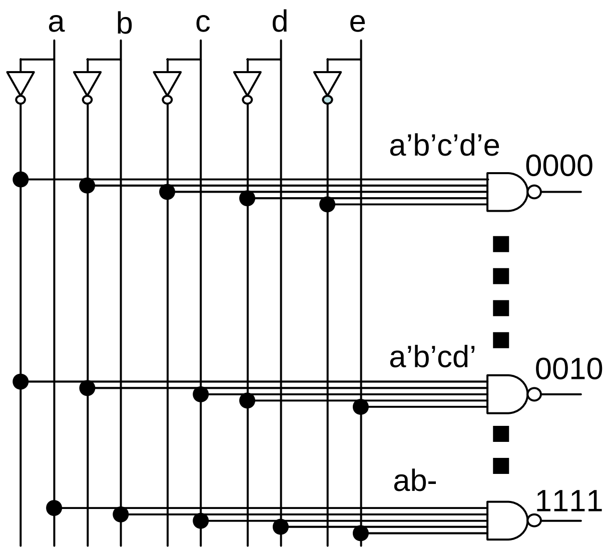

Figure 5: 5-Bit NAND-Based Address Decoder

Figure 5 illustrates the diagram of a 5-bit address decoder. Assume that the illustrated address decoder is used for a 32x16 RAM. Further consider that the second PMOS transistor of the NAND gate (line *b*) generating the address 00101 is faulty. When the base address is 00101 while applying the ADOF algorithm, the neighboring test addresses that are considered are as follows: 00100, 00111, 00001, 01101, 10101. Since the PMOS transistor corresponding to line b is open, when the controller tries to write to test address location 01101 the output of the NAND gate doesn't change and therefore the base address location 00101 is overwritten. Consequently, when the controller goes back and reads the base address location, it sees that the value has changed and concludes that the address decoder corresponding to the base address location 00101 is faulty. When the fault is detected, the value in the shift register is 01000. Further, since the data-logger also scans out the value in the shift register needed to generate the test address, one can determine the address line corresponding to failing PMOS transistor.

The ADOF algorithm described in this section will not help in isolating the transistor if there are multiple PMOS opens in a NAND gate. This is because once there is an error, the memory location corresponding to the base address register will be corrupted. For all subsequent neighboring addresses the controller will report a failure irrespective of whether the transistor corresponding to that line actually failed or not. This will increase the volume of failing data from the data-logger. This can be prevented by slightly modifying the algorithm step to be:

$$\{ \Uparrow (W0); \Uparrow (W1, \Uparrow_H (W_H0, R1, W1), W0) \}$$

Essentially, an extra write operation to the base address after writing to every test address is added in the inner loop to reset the value that may have been changed if the transistor actually failed. This increases the test application time but helps in pinpointing the location of the failing transistor. The basic ADOF algorithm can be easily modified to better match the test goals using a language that communicates to the BIST compiler the algorithm..

5.1 Area Overhead

As with any algorithm, one of the major concerns when including support for a new algorithm into a BIST controller is the additional area overhead that is incurred. Within TI, numerous memories are assigned to a BIST controller depending on their proximity of location and similarity in behavior. To characterize the area overhead, two memories of sizes 128x32 and 512x32 were used. In the first case, two of the TI standard set of algorithms was used, whereas, in the second case, the ADOF algorithm was added to the standard set. The ADOF algorithm was used with data background and its inverse.

Table 4: Area Overhead for ADOF Algorithm (Sequential Interleaved Configuration)

Sequential Interleaved Configuration			
	Standard Set of Algorithms	Standard Set + ADOF	Additional gates
2 Memories	1849	2010	161
2 Memories + Diagnosis	4522	4810	288
8 Memories	2770	2941	171
8 Memories + Diagnosis	5523	5806	283

Table 5: Area Overhead for ADOF Algorithm (Sequential Non-Interleaved Configuration)

Sequential Non- Interleaved Configuration			
	Standard Set of Algorithms	Standard Set + ADOF	Additional gates
2 Memories	1847	2012	165
2 Memories + Diagnosis	4496	4808	312
8 Memories	2746	2912	166
8 Memories + Diagnosis	5495	5776	281

Table 4 and

Table 5 show the area overhead incurred for adding the ADOF algorithm for sequential interleaved and non-interleaved configurations. The area overhead numbers were calculated based on basic 2-input NAND/NOR gates used in the design. Two different configuration were used – two memories (Configuration 1) and eight memories per controller (Configuration 2).

The area overheads were presented for controllers with and without the data-logger. As is evident from the tables, the additional area overhead for Configuration 1 was around 9% of the area of the BIST controller, whereas for Configuration 2 it was approximately 7% of the area of the BIST controller. The area overhead for adding the ADOF algorithm didn't increase substantially as the number of memories per controller was increased. This is true for sequential configurations as the BIST controller hardware was shared across multiple memories assigned to a controller.

Note, there was not much difference in the area overhead when the data-logger was added because the tool didn't by default add an extra register needed to store the shifted value required for identifying the failing PMOS transistor. The conventional diagnostic data that was scanned out provides sufficient information about the algorithm step, address location, memory number, etc., from which one can diagnose the failing gate in the address decoder easily.

6 Bit/Byte Write-Enable Mask Tests

Several ASIC RAMs have a vector input signal that enables or disables the writing circuitry of associated bits in the data words. There are two frequently seen versions of this. The first uses one-to-one association with *bit[i]* of the mask controlling whether *bit[i]* of data is changed during a write. Several TI RAMs have this type of write masking. The second type includes each bit of the mask controlling whether a byte of the data word is writable.

These mask signals, which often are routed close to each other, are subject to the same manufacturing defects as the memory arrays. Among these are stuck-at faults and shorts between two neighboring lines. It is often the case that memories are built out of blocks. The write enable signals may be driven to the blocks after buffer stages. In such a case, defects may only be detected by testing against all blocks.

The default action of most memory BIST controllers is to set these signals to their asserted values. In other words, all bits of the data word are enabled concurrently for writing. This is needed for tests, such as March algorithms, that are looking for cell-related defects. Consequently, if any of these signals are stuck-at their asserted values (on-state), standard March algorithms will not detect such failures.

A full test for all possible defects in the write-enable mask (WEM) signals would involve an extension to testing for pattern-sensitive faults. However, such a level of testing is usually impractical for manufacturing testing, and therefore only a subset of tests is investigated. In the following paragraphs, it is assumed that standard tests, with the WEM signals all set to write state, have been performed with no faults detected. Otherwise, any such faults are likely to lead to fault detection during write enable mask testing that is not related to the mask circuitry.

The minimal test would be to write memory with one pattern with all WEM signals asserted active, followed by a test that writes the inverted pattern with the WEM signals all set to inactive. The memory would then be read expecting the original pattern. This would be a test for stuck-at faults. This could be followed by another write with the inverted pattern with WEM all set to active. This is a minimal test to catch a transition fault where the signals are changing from their inactive state to an active one.

The above tests do not catch shorts between the WEM lines. Such tests will require that some of the WEM be set to active while others are set inactive.

There are two conceptually simple approaches to accomplish this – walking 1, walking 0, and checkerboard patterns, as described below.

a) A *walking 1/0* test involve settings each bit of the WEM signals to 1 (or 0) with a write process followed by a read test. The walking 1 test checks that nearby 0's in the WEM that are not shorted to the high signal. On the other hand, a walking 0 test checks that nearby 1's are not able to pull-up a 0. It is desirable to apply both the walking 0/1 patterns when testing for failures in the WEM signals.

b) A *checkerboard* test uses a different approach, where the WEM signal bits are set to a pattern of alternating 0's and 1's. It should be noted that both checkerboard and inverse checkerboard patterns should be used because some circuitry is more prone to pull-down, whereas, some is more prone to pull-up issues.

The description of a checkerboard algorithm for a memory with 16-bit data word and one WEM signal per bit is included in the following:

```
// Checkerboard mask
1. Initialize memory, data = 0, WEM = 16'h0000
2. Set write data = 16'hFFFF
3. Write memory with WEM = 16'h5555
4. Read memory expecting 16'hAAAA
// Inverse checkerboard mask
5. Initialize memory, data = 0, WEM = 16'h0000
6. Set write data = 16'hFFFF
7. Write memory with WEM = 16'hAAAA
8. Read memory expecting 16'h5555
```

Note that the WEM signals are effective only during the write operation. During reads, the entire memory word is read to determine if there are any issues with the WEM signals.

Since the write enable mask is common to all memory locations, only one memory cell is necessary for applying the test. Alternatively, the number of cells equal to the width of the RAM is also used.

6.1 Supporting the WEM Test in a BIST Controller

In a typical BIST controller, the actual value read from the memory is compared with the expected value that is generated within the controller at the appropriate time to determine whether the memory is functioning correctly. For standard March algorithms, the expected value has a simple relationship to the current pattern. Typically, it is either the current pattern or its inverse. In order to support a WEM test, the data that has to be written must remain the same, but the desired masks have to be applied to the WEM signals during the write process. As a result, after the write operation, the memory will be left with values prior to the write operation which are masked. When the actual data that is read from memory is compared with the expected data, the BIST controller implements the logic for generating the correct expected data.

The changes in the BIST controller primarily concern the logic performing the following tasks. First, additional logic is implemented to store and generate all the WEM that are desired by the user. Once the WEM are specified, flexibility is introduced to allow the designer to describe algorithm steps that use the WEM. The area overhead in this case directly depends on the number of WEM chosen, although certain types of masks such as walking 0/1 can be generated using a shift register and don't have to be explicitly stored within the controller. Second, the expected data for the comparator needs to be generated by employing the specified masks. Additional logic is included to distinguish whether an operation includes, or does not include a mask and the mask of interest. Finally, there is some area incurred in the BIST controller itself since the area of the BIST's finite state Machine increases as new algorithms steps are included.

6.2 Area Overhead

In order to characterize the area overhead incurred by including the WEM tests experiments were conducted based on two memory models with the following sizes – 32x32 and 46x32. Two configurations were used: sequential non-interleaved and sequential interleaved, for a 2 memory and 8 memory per controller. The controllers had four algorithms after which the algorithms to test the WEM signals were added. Checkerboard and inverse checkerboard patterns were used for the masks.

Table 6: Area Overhead for WEM Algorithm (Sequential Interleaved Configuration)

Sequential Interleaved Configuration			
	Standard Set of 4	Standard Set +	Additional gates

	Algorithms	Write mask test	
2 Memories	1849	2112	263
2 Memories + Diagnosis	4522	4829	307
8 Memories	2770	3044	274
8 Memories + Diagnosis	5523	5854	331

Table 7: Area Overhead for WEM Algorithm (Sequential Non-Interleaved Configuration)

Sequential Non-Interleaved Configuration			
	Standard Set of 4 Algorithms	Standard Set + Write mask test	Additional gates
2 Memories	1847	2113	266
2 Memories + Diagnosis	4496	4818	322
8 Memories	2746	3031	285
8 Memories + Diagnosis	5495	5841	346

Table 6 and Table 7 show the area overhead in terms of 2-input NAND gates with and without the data-logger. The controller was synthesized to run at 400 MHz. As can be seen, the area overhead is between 6-8% for controllers with the data-logger, whereas, it is between 10-14% for controllers without the data-logger.

7 Conclusions

Customers for ASIC RAMs are pleased with denser technologies to permit more function and memories per design. Low defect rates are required, which means all potential defects that can occur in memories must be screened. The TI ASIC flow now has a means for including tests for the ADOF and to test the control functions such as bit write-enable lines.

8 Acknowledgement

Appreciation is extended to Jayashree Saxena for the MacroTest test prepared to screen ADOF and to Kohichi Atsumi for the algorithm analysis. The review and feedback from Mark Chadwick was also appreciated.

9 References

1. A.J. van de Goor, “Testing Semiconductor Memories, Theory and Practice,” Publisher: A.J. van de Goor, pp 445-449

2. B. Nadeau-Dostie, et.al, “Serial Interfacing for Embedded Memories,” *IEEE Design & Test of Computers*, 1990, 7 (2), pp. 52-63

3. Matthias Klaus, et al., “Tests for Resistive and Capacitive Defects in Address Decoders,” *Proc. Asian Test Symposium*, 2001, pp. 31-36

4. Otterstedt, et al., “Detection of CMOS Address Decoder Open Faults with March and Pseudo Random Memory Test,” *Proceedings of International Test Conference,* 1998, pp. 53-62

5. L. Dilillo, et al., “March iC-:An Improved Version of March C- for ADOFs Detection,” *Proceedings of VLSI Test Symposium,* 2004

6. Sachdev, et al., “Open Defects in CMOS RAM Address Decoders,” *IEEE Design & Test of Computers,* April-June 1997, pp. 26-33

7. Theo J. Powell, et al., “BIST for Deep Submicron ASIC Memories with High Performance Application,” *Proceedings of International Test Conference,* 2003, pp 386-392

8. Mohamed Azimane, et al., “New Test Methodology for Resistive Open Defect Detection in Memory Address Decoders,” *Proceedings of VLSI Test Symposium,* 2004, pp 123-134

9. D.E. Ross, T. Wood, G. Giles, “Conversion of Small Functional Test Sets of Non-scan Blocks to Scan Patterns,” *Proceedings of International Test Conference*, October 2000, pp. 691-700

10. J. Boyer, R. Press, “New Methods Test Small Memory Arrays,” *Test and Measurement World,* February 2003

11. Benoit Nadeau-Dostie, “Design for At-Speed Test, Diagnosis and Measurement,” Kluwer Academic Publishers, 2000

12. R. Dean Adams, “High Performance Memory Testing, Design Principles, Fault Modeling and Self-Test,” Kluwer Academic Publishers