IEEE IC Test Workshop
Limerick, Ireland, 13 – 14 September 2004

BUILT-IN TEST ENGINE FOR MEMORY TEST

Paul McEvoy
Department of Electronic Engineering
National University of Ireland Maynooth, Co. Kildare.
Phone: (01) 708-6384, email: pmcevoy@eeng.may.ie

Ronan Farrell
Department of Electronic Engineering
National University of Ireland Maynooth, Co. Kildare.
Phone: (01) 708-6197, email: rfarrell@eeng.may.ie

Abstract - *In this paper we will present an on-chip method for testing high performance memory devices, that occupies minimal area and retains full flexibility. This is achieved through microcode test instructions and the associated on-chip state machine. In addition, the proposed methodology will enable at-speed testing of memory devices. The relevancy of this work is placed in context with an introduction to memory testing and the techniques and algorithms generally used today.*

Keywords *BIST, memory, at-speed, DFT*

I. INTRODUCTION

Semiconductor memories are considered one of the most important aspects of modern microelectronics. Memory stores data and instructions used in almost all modern technology such as computers, telecommunications and consumer electronics. Memory is either volatile; loses data when power is removed or non-volatile; stores data indefinitely. Volatile memory can be SRAM or DRAM which stores data using a flip-flop and a capacitor respectively. ROM, EPROM, EEPROM and flash memory are examples of non-volatile memory where the data can be permanent or reprogrammable, depending on the fabrication technology used [1]. Testing nanometre integrated circuits (ICs) is expensive and can account for almost half the cost of memory chips [2]. As the transistor count on devices follows Moore's law, memory devices develop increasing storage capabilities. Manufacturing advances mean that decreasing silicon feature size enables designers to fit more memory cells per chip. With the decrease in memory cell size, however, there is an increase in sensitivity to defects. Smaller geometric features, reduced internal voltage levels, and the use of new physical processing techniques such as copper interconnect and low-K dielectrics are causing the number of resistive type defects to increase. New methods to test devices and maintain low defects per million (DPM) are required.

II. TEST METHODS AND ALGORITHMS

Digital devices are tested with an external tester that provides all the control pins and address lines to apply test patterns. As devices have become more complex and at-speed testing emerges as a vital requirement, new methods such as design-for-test (DFT) and on-chip test architecture are seen as necessary approaches to maintain quality levels while still keeping test overheads low.

One of the most common techniques for on-chip memory test is built-in self-test (BIST). BIST removes the need for many of the functions on automatic test equipment (ATE) such as high-speed external pins and off-chip memory storage thus can decrease test costs and time. A BIST core will input test patterns to the device under test (DUT), read the data back, compare it to expected responses and register a pass or fail on a given test. These test patterns can be generated deterministically or randomly [3]. Test patterns generated can detect not only modelled defects, but also non-modelled timing defects. Deterministic algorithms are usually adopted for BIST due to their simplicity. The BIST can also include diagnosis and repair capabilities depending on the design requirements.

Embedded memory on Systems On-Chip (SOCs) can create significant test issues due to its integration and proximity to logical cores. It is predicted that in Application-Specific Integrated Circuits (ASICs) embedded memory will occupy 94% of die area [4]. In this case a suitable BIST architecture is vital to production yield. The combination of a flexible memory BIST engine that allows custom variations to the test algorithms, with enhanced at-speed application provides the basis for ensuring high-quality testing. This is increasingly important considering the trend toward designs with more memory than logic.

Although BIST can provide increased test coverage in less time there are design issues that must be considered. Its implementation also adds area overhead as the consumed silicon can not be used for functional purposes. BIST may also compromise flexibility in application as it will generally be design specific - any changes would require global SOC alteration. It may also require extra pins to interface with an external tester adding further complexity. As with any logic added to ASIC the BIST will have to be tested itself. Scan based methods are normally used to ensure tester correctness. Despite this, for medium to large memories it has been shown that BIST can provide significant cost benefits when applied [2], (see figure 1).

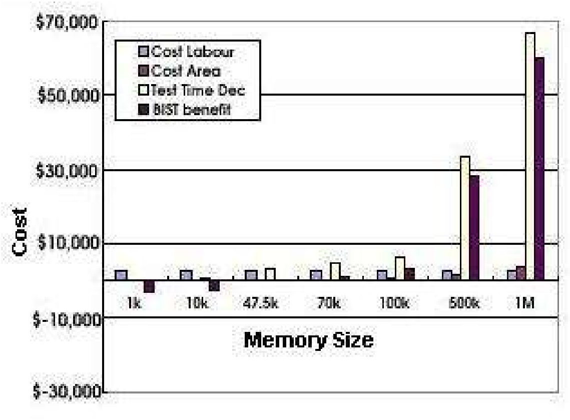

Figure 1: Cost benefit for memory BIST on embedded memory.

III. FAULTS AND DEFECTS FOR MEMORY

A defect is a difference between a manufactured product and its design. Defects in memory arrays are generally due to shorts and opens in memory cells, address decoder and read/write logic.

There are two types of defect; *global* and *local*.

- Global defects

  Can be caused by too thick gate oxide, or too thin polysilicon, mask misalignments and process variations. They can affect many chips on a wafer and are the main cause of dynamic faults such as delay and timing faults.
- Local defects

  These (also called spot defects) are caused by extra, missing or inappropriate material, such as dust particles. A spot defect will only affect a single chip and causes a functional fault.

A fault model is an attempt to give a representation to the possible manufacturing and design defects that can occur. Below are some of the most common fault models for RAM. [5]

- A *stuck-at fault* (SAF) occurs when the value of a cell or line is always 0 (a stuck-at-0 fault) or always 1 (a stuck-at-1 fault).
- A *stuck-open fault* (SOF) occurs when the cell can not be accessed due to a broken word line. A read to this cell will produce the previously read value.
- If a cell has a *transition fault* (TF), then it fails to transit from 0 to 1 (a $\uparrow$/0 TF) or from 1 to 0 (a $\downarrow$/1 TF).
- A *data retention fault* (DRF) occurs when a cell cannot store its contents for a specific amount of time. In SRAM this is caused by a broken pull-up resistor.
- An *address decoder fault* (AF) is a functional fault in the address decoder that results in one of four kinds of abnormal behaviour:

1. With a certain address, no cell will be accessed.
2. A certain cell is never accessed.
3. With a certain address, multiple cells are accessed.
4. A certain cell can be accessed with multiple addresses.

- A *coupling fault* (CF) between two cells occurs when the logic value of a cell is influenced by the content of, or operation on, another cell. There are several types of coupling fault.
  - An *inversion coupling fault* (CFin) occurs if a transition in one cell inverts the logic value of another.
  - There is an *idempotent coupling fault* (CFid) if a transition in one cell forces a fixed logic value into another.
  - Finally, there is a *state coupling fault* (CFst) if a cell or line is forced to a fixed logic value only if the coupling cell or line is in a given state.

Linked faults affect the same cell. In linked CFs, two or more CFs exist with the same coupled cell. Linked faults may occur between faults of the same type or between faults of different type. Test algorithms must be constructed to prevent faults from being masked by a linked fault.

Different memory designs may need additional fault models to cover all possible defects. For DRAM, fault models may be needed to cover failures caused by all kinds of leakage and noise. Flash memory has read, erase and program disturbance faults due to its design that also need to be covered if tested.

IV. MEMORY TEST ALGORITHMS

An algorithm is a set of operations performed on a device under test (DUT) to determine functionality. Numerous algorithms exist that target specific faults such as zero-one, checkerboard, GALPAT and walking 1/0s [1]. However, many of these test patterns have limited fault coverage or have considerable complexity limiting their use in modern testing.

One family of tests that has proven to be particularly effective in test time and complexity is the March algorithm [6]. A March test consists of a sequence of March elements. A March element consists of a sequence of operations applied to each cell in the memory, before proceeding to the next cell. An operation can consist of writing a 0 into a cell (w0), writing a 1 into a cell (w1), reading a cell with an expected value 0 (r0) and reading a cell with an expected value 1 (r1). After all operations of a March element have been completed they are applied to the next cell. The address of the next cell is given by the address order either ascending or descending. An ascending order moves from address 0 to n-1 denoted by $\uparrow$ and a descending order, from address n-1 down to 0 denoted by $\downarrow$. When the order is irrelevant $\updownarrow$ is used. In an ascending order any address sequence may be used as long as the descending order uses the exact inverse sequence. There are a range of March algorithms that exist that test for a wide number of faults. For instance, to test for a stuck-at fault (SAF) the algorithm must contain a sequence that verifies that a 0 and a 1 can be read from every cell. MATS+ is an example of a simple March test.

$$\updownarrow(w0)\ \uparrow(r0,w1)\ \downarrow(r1,w0) \qquad (1)$$

It will detect all AFs and SAFs as a 0 and 1 are read and written from each cell.

Common to all memory BIST implementations is an address generator, a test pattern generator and BIST control logic. The BIST controller can be implemented by either hard-wired logic or microcode. The most common type of hard-wired memory BIST consists of a finite state machine (FSM) performing three basic operations: writing patterns to the memory, reading them back and comparing them to the expected results.

It has been shown that the most costly feature to implement when using BIST is the storage space required for the algorithms and instructions. One common approach is to use an on-chip ROM to store the instruction set [7]. The area overhead is significant, however, and removing this factor could reduce cost considerably. Assuming the logic area overhead is negligible the silicon area for memory BIST, A_{MB} is calculated by

$$A_{MB} = 1 + \frac{(\log_2(M_a))}{M_a} \times 1.33 \qquad (2)$$

where M_a is the memory size used to store the test instructions. To try to overcome this area issue, attempts have been made to use microcode instructions that call on specific March operations [8]. Although this approach reduces stored memory overhead, March primitives must still be stored on-chip and the logic overhead is significant.

V. THE ON-CHIP SOLUTION

The proposed solution is to store the memory algorithms off-chip, loading them into a register file inside the BIST engine temporarily. The test is then run at-speed until the algorithm is completed. For each read operation the comparator checks the output data with the expected values and notifies the automatic test equipment (ATE) of any failure. When an algorithm has been completed another is passed in and the process is repeated.
The architecture consists of a small register containing the algorithm being used (see figure 2). A simple shift register is used to store the pattern data and pass it to the address decoder that converts the code into control signals, data and address calls to the memory. The decoder also includes an instruction counter, tracking operation progress, and an input from the comparator to detect faults that occur and their location. The order in which the memory is accessed is given by the algorithm, and is tracked by the decoder which controls the address generator accessing the column and row address lines. A multiplexer is used to switch between system and BIST control. When the memory is instructed to perform a read operation it is verified using the comparator and control is sent to the decoder to register any deviation from expected data.

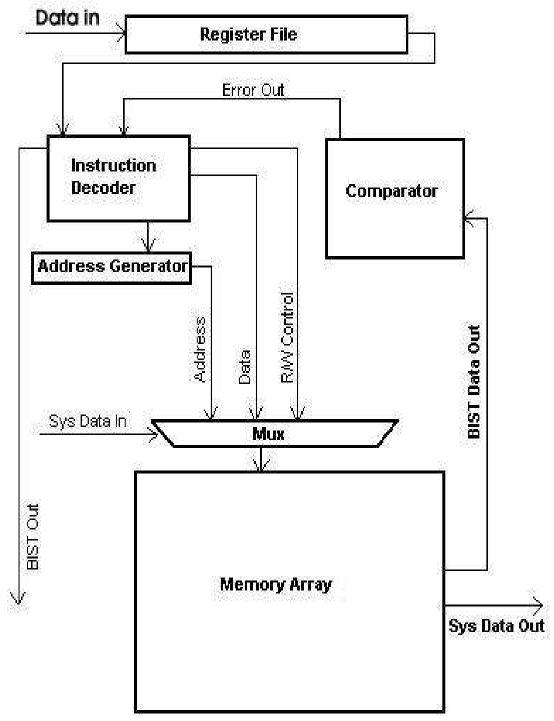

Figure 2: Proposed BIST architecture.

Required register storage must be determined when converting the algorithm to a bit pattern. To minimise the size of the register we propose using a short microcode to instruct the decoder on the required operations and control signals.

There are four possible operations when testing memory data; read-zero (r0), read-one (r1), write-zero (w0) and write-one (w1). When accessing the cells sequentially there is also an addressing order; incremental or decremental. A microcode example, the MATS+ algorithm is given in (1). Each operation is given a code (see table 1) and a complete code string is constructed.

operation	*code*
r0	*00*
r1	*01*
w0	*10*
w1	*11*
direction	*1*
delay	*00*
delay 1	*01*
delay 2	*10*
delay 3	*11*
EE	*1/0*

Table 1: Microcode operation assignments

The microcode must produce a set of elements containing operations. For each element there is an address order that will not change until the next element. To implement this a 3-bit header is used containing the address direction and a 2-bit delay assignment. The delay allows up to three clock cycles after the element is completed to check for *data retention faults* (DRFs). Following the header, the r/w operation is given and the 6th bit indicates the end-of-element signal (EE). If the element is complete, EE will be high and new header is given. If further operations are remaining EE is low, another operation is supplied and an EE check is made again. Operations are provided until the element is complete. The EE bit is included to minimise the the storage required for each code, as the address order and delay are only required once for each element.

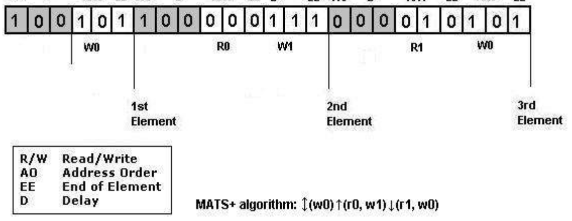

Figure 3: March algorithm microcode MATS+ microcode register contents.

Each algorithm will require a different number of bits to code, and obviously more complex patterns need more storage. The register is such that it can hold the longest algorithm that will be used and possibly longer, depending on whether future patterns will be included or previous ones are supplemented to. From the above example the MATS+ algorithm requires a total of 24 bits to code completely. Equation (3) is used to determine the number of bits needed where n_e is the number of elements and n_o is the number of operations. An example is shown illustrating a few of the most common March tests (table 2).

$$3(n_e + n_o) \tag{3}$$

The instruction decoder consists of a finite state machine (FSM) that enters a specific state based on the input from the microcode register. The decoder increments or decrements the address counter and controls specific instructions to the memory array such as read, write, hold and test end. Using an FSM minimises logic overhead and simplifies design.

Algorithm	*bits required*
MATS+	*24*
March C-	*48*
March B	*63*
March G	*90*

Table 2: March algorithm bit requirements

VI. COMPARISON WITH EXISTING SOLUTIONS

The proposed BIST architecture provides a number of advantages over existing approaches. One main benefit of the design is that it can be applied to any type of memory architecture with minimal design alteration. A different memory design will merely require different test patterns that can be changed externally. Using the proposed microcode, memory test algorithms can be highly compacted, reducing storage requirements (table 2). By allowing for the largest possible tests, a suitably sized register could facilitate very complex and thorough test patterns. This method significantly reduces the memory required on-chip compared to previous methods [7] [8], while also adding increased flexibility to alter algorithms when necessary. This flexibility is critical in modern designs since advances in technology and design demand higher performance tests to maintain quality.

Another significant advantage is the ability to test the memory chip at its functional speed. This is critical to detecting faults that will only occur during operation such as *data retention faults* (DRFs). Current ATEs may only function at a fraction of the device operating speeds; high speed RDRAM may run at 800MHz while the ATE may be limited to 100MHz.

At-speed testing can also result in lower test times as data can be compared on-chip avoiding the need to pass all the processed information to the ATE. In addition, it is possible to take advantage of the parallel internal structure of memories to run concurrent tests on different memory banks.

The reduction in requirements from ATEs will demand less capital expenditure and upgrade costs involved in testing each type of memory device. It may be possible to support complete evaluation with a basic desktop setup; memory storage, data scan outputs and a pass/fail input.

A major concern with any BIST implementation is the requirement for extra silicon area. This area adds to the cost of the device, as this represents parts of memory and logic that are not added to the chip for functional purposes. However, using the proposed approach, memory overhead is minimal. The logic area requirement should also be negligible as only a short finite state machine and comparator are used.

Further investigation will be needed when testing flash memory. Not only is the architecture quite different but the operating speeds are much lower than RAM. A typical program operation on flash memory may take a long time (typically 100µs), with block erase taking even longer. In this case, the benefits of testing on-chip for speed-related gains are minimal.

VII. CONCLUSIONS

In this paper, we have proposed a method to test memory on-chip using microcode-based test algorithms. This approach applies patterns from an external tester that are stored into a register in the BIST engine. We have outlined the importance of testing devices at-speed with minimal area overhead. The BIST design allows at-speed testing with the added flexibility for the design to be used in a range of different memory types such as SRAM, DRAM and ROM.

ACKNOWLEDGEMENTS

The authors would like to thank Pat Brady and Desmond FitzGerald at Intel Ireland for their help in the design and test technology involved.

REFERENCES

[1] A.J. van de Goor, *Testing Semiconductor Memories: Theory and Practice.* Chichester, England: John Wiley and Sons, 1991.

[2] Juin-Ming Lu, Cheng-Wen Wu, *Cost and benefit models for logic and memory BIST* Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings, 27-30 March 2000

[3] Sying-Jyan Wang, Chen-Jung Wei, *Efficient built-in self-test algorithm for memory,* Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian, 4-6 Dec. 2000 Pages:66-70

[4] Venkatesh, R. Kumar, S. Philip, J. Shukla, *A fault modeling technique to test memory BIST algorithms.* Memory Technology, Design and Testing, 2002. (MTDT 2002).

[5] A.J. van de Goor, *Using March tests to test SRAMs.* Design & Test of Computers, IEEE, Volume: 10, Issue: 1 ,March 1993 Pages: 8-14

[6] M. Marinescu. *Simple and efficient algorithms for functional RAM testing.* In Proc. Int. Test Conf pages 572-576, 1982.

[7] Dreibelbis, J. Barth, J. Kalter, H. Kho, R. *Processor-based built-in self-test for embedded DRAM.* Solid-State Circuits, IEEE Journal of, Volume: 33 ,Issue: 11, Nov. 1998

[8] Dongkyu Youn, Taehyung Kim, Sungju Park; *A microcode-based memory BIST implementing modified March algorithm.* Test Symposium, 2001. Proceedings. 10th Asian, 19-21 Nov. 2001