ISSN:0975-9646

Design of Fault Detection Module for Embedded Ram Memory

Mohanish P. Gadge , S.P.Karmore
CSE, GHRCE, Nagpur

Abstract- **Embedded RAMs are those whose address, data, and read/write controls cannot be directly controlled or observed through the chip's 1/0 pins. Testing these memories, which are incorporated on a large percentage of VLSI devices are harder just because of the lack of controllability of its inputs and observe ability of its outputs. Testing such RAMs is the main objective of this paper. It is challenging to test embedded RAMs, and hence we will discuss techniques - design for testability (DFT) and built-in self test (BIST), which help in improving the testability of these RAMs.**

Keywords- Built-In Self Test (BIST), embedded memory fault, Modified March algorithm, Microcode, Transition fault, neighbourhood pattern sensitive faults(NPSF).

I.INTRODUCTION

Embedded RAMS are widely used in digital integrated Systems such as telecommunications ASICs and broadband ISDN, in processors for implementing cache memories and registers, and in DSP chips for storage of weights.

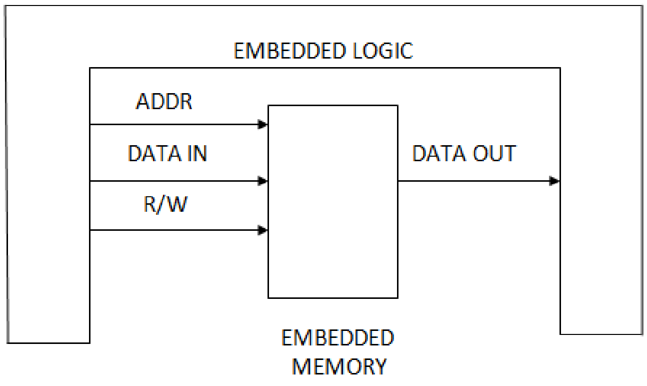

Fig.1 Embedded RAM

It is not possible to test an embedded RAM simply by applying test patterns directly to the chip's 1/0 pins, because the embedded RAM'S address, data, and control signals are not directly accessible through the 1/0 pins (c.f. Figure 1). Instead, their inputs and outputs have to be accessed through the logic in which the memories are embedded. Many of the RAM test algorithms involve applying test vectors in a particular sequence, and it would be difficult to apply the test vectors in the correct sequence through the embedding logic, let alone generate all of the required test vectors.

II.FAULT MODEL

Following are different fault models that are used to detect fault-

A. Stuck-at Fault Model

The logic value of memory is stuck- at '0' or 'l', and not changed.

B. Transition Fault Model

The logic value of memory is not changed from '0' to 'l' (upward transition), or '1' to 'O' (downward transition).

C. Coupling Fault Model

The transition of logic value in one cell changes logic value of related cells.

III. MICROCODE MBIST CONTROLLER

As shown in the above section, the need of developing new fault models increases with the new memory technologies. In addition, the shrinking technology will be a source of previously unknown defects/faults [1]. In the late 1990's, experimental results based on DPM of a more number of tests applied to a large number of memory chips indicated that many detected faults could not be explained with the well known fault models, suggesting the existence of additional faults. This gives the introduction of new fault models, based on fault injection and SPICE simulation: Read Destructive Fault (RDF), Write Disturb Fault (WDF), Transition Coupling Fault (Cft), Deceptive Read Disturb Coupling Fault (Cfdrd) etc. [1] Another class of faults called Dynamic faults which require more than one operation to be performed sequentially in time in order to be sensitized have also defined. [4-5]. Traditional tests, such as March C-, are becoming insufficient/inadequate for today's and the future high speed memories. Therefore, more appropriate test algorithms have been developed to deal with these new fault models. Examples of such tests are March BLC [2] and March RAW [4]. March BLC covers some of the new fault models like Deceptive Read Destructive fault, Write disturb fault, etc., whereas March RAW covers some of the Dynamic faults.

These new test algorithms have as many as six or seven operations per march element, and thus some of the recently modelled and simulated architectures are inadequate to implement these test algorithms, as they have been developed to make space for only up to two test operations per March element [3]. This architecture is capable of implementing the newly developed March algorithms, because of its ability to execute algorithms with unlimited number of operations per March element. many of the recently developed March algorithms can be applied using this architecture. In this paper we present March BLC [2], an optimized test that detects all static faults in the presence of BL coupling using only the need CBs, with a complexity of test time equal to 46n. Compared to March m-MSS (108n) [16], which applies all possible CBs, the test time is significantly reduced by over 50%.

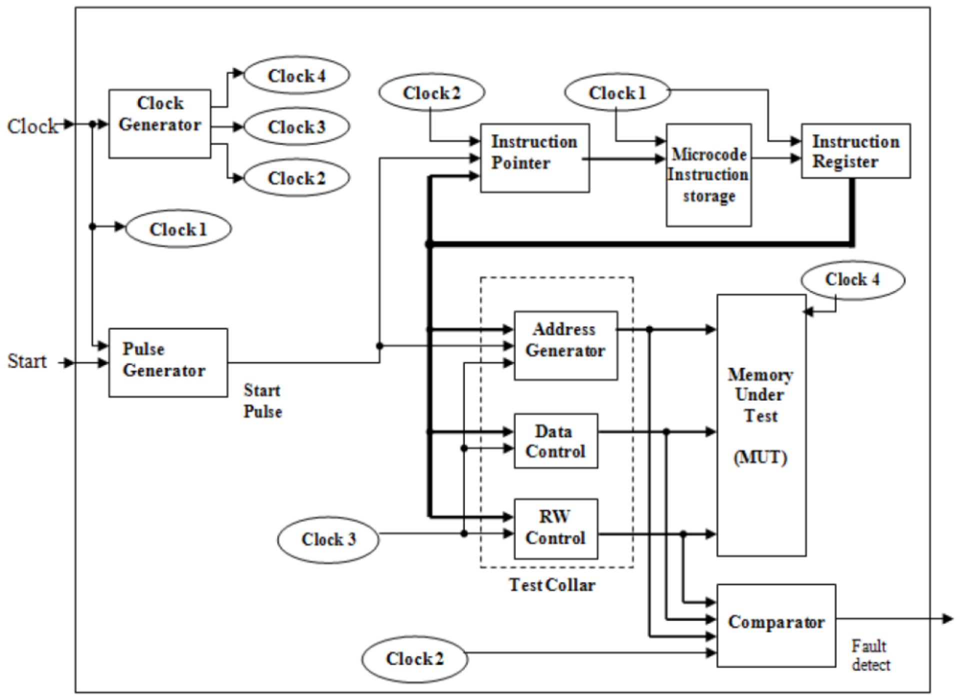

Fig.2 Microcode Architecture

March BLC = {□ (w0); ME0
□ (r0, r0,w0, r0,w1,w1, r1); ME1
□ (r1, r1,w1, r1,w0,w1); ME2
□ (r1, r1,w0,w0, r0); ME3
□ (r0, r0,w0, r0,w1,w1,w0); ME4
□ (r0, r0,w0,w1,w1, r1); ME5
□ (r1, r1,w0,w1); ME6
□ (r1, r1,w0,w0, r0); ME7
(r0, r0,w1,w1,w0)} ME8

This has been illustrated in the present work by implementing March BLC algorithm. However, the same hardware can be used to implement other new March algorithms also. To store predetermined test pattern the instruction storage unit is used.

A. Methodology

The block diagram of the architecture is shown in Fig 2. The BIST Control Circuitry consists of Clock Generator, Test collar circuitry, Microcode Instruction storage unit Pulse Generator, Instruction Pointer, Instruction Register, consists of Address Generator, RW Control, Data Control, Clock Generator generates simulated clock waveforms Clock2, Clock3, Clock4, for the rest of the circuitry based on the input clock (named Clock1) as shown in Fig. 3 Pulse Generator generates a 'Start Pulse' at positive edge of the 'Start' signal which marks start of test cycle. Instruction Pointer points to the next micro word, that is the next march operation to be applied to the memory under test (MUT). Depending on the test algorithm, it is able to i) point at the same address, ii) point to the next address, or iii) jump back to a previous address. The Address Generator, RW Control and Data Control together constitute the Memory Test Collar Comparator gives the fault waveform which consists of positive pulses whenever the value being read out of the memory does not match the expected value as given by Test Collar.

B. Microcode Instruction specification.

The microcode is a binary code that consists of a fixed number of bits, each bit having a particular data or operation value. There is no standard in developing a microcode built in self test instruction. The instruction fields can be structured by the designer depending on the test pattern algorithm to be used. The microcode instruction developed in this work is coded to denote one operation in a single microword. Thus a five operation March element is made up by five micro-code words. Table 1 shows the 7-bit microcode MBIST Instruction word and description of its various fields Bit #1 (=1) indicates a valid microcode instruction, otherwise, it indicates the end of test for BIST Controller. Bits #2, #3 and #4 stand for first operation, in-between operation and last operation of a multi-operation March element. A detailed description of how these three bits are interpreted is given in Table 2.

TABLE I
7-Bit Microcode Instruction Format

#1	#2	#3	#4	#5	#6	#7
Valid	F0	I0	L0	I/D	R/W	Data

TABLE II
Operation Field Specifications

F0	I0	L0	Description
0	0	0	A single operation element
1	0	0	First operation of a Multi-operation element
0	1	0	In-between operation of Multi operation element
0	0	1	Last operation of a multi operation element

Bit #5 (=1) notifies that the memory under test (MUT) is to be addressed in decreasing order; else it is accessed in increasing order. Bit #6 (=1) indicates that the test pattern data is to be written into the MUT; else, it is retrieved from the memory under test. Bit #7(=1) signifies that a byte of 1s is to be generated (written to MUT or expected to be read out from the MUT); else eight bits of all zeroes are generated. The instruction word is so designed so as to represent any March related algorithm. The Instruction storage unit contents for March BLC algorithm are shown in Table 3. The first march element M0 is a single operation element, which writes zero to all memory cells in any order. Similarly, the second March element M1 is a multi-functional element, and it consists of only five operations: i) R0, ii) R0, iii) W0, iv) R1 and v) W1. MUT is addressed in increasing order because before moving on to the next location each of these five operations is performed on each memory location. The top module shows the interfacing of BIST Controller (including test collar), MUT and Comparator. As the START signal goes high, indicating the start of test, the first March element M0 of March BLC algorithm is executed. As this is a write signal, no values are read out from the memory to be compared with expected or correct values and hence the output FAULT waveform of comparator is high impedance. As read operation starts at the beginning of execution of M1 element, the values from MUT are read out and compared with the expected values. The FAULT waveform shows a 'low' level throughout the test for a fault-free SRAM. The SRAM model is also amended to be in defective state by inserting faults. The simulated waveform is shown in Figure 6.

IV.Conclusion

The above waveforms have shown that the micro-code MBIST architecture above is an effective testing method to test embedded memories as it provides a flexible approach and better fault coverage. Just as March BLC , any new march algorithm can be implemented using the same BIST hardware by replacing the microcode storage unit, and hence we don't need to redesign the entire circuitry.

References

[1] S. Hamdioui, G.N. Gaydadjiev, A.J .van de Goor, "State- of-art and Future Trends in Testing Embedded Memories", International Workshop on Memory Technology, Design and Testing (MTDT'04), 2004.

[2] Sandra Irobi Zaid Al-Ars Said Hamdioui.Memory Test Optimization for Parasitic Bit LineCoupling in SRAMs. IEEE International Test Conference, 2010.

[3] N. Z. Haron, S.A.M. Junos, A.S.A. Aziz, "Modelling and Simulation of Microcode Built-In Self test Architecture for Embedded Memories", In Proc. of IEEE International Symposium on Communications and Information Technologies pp. 136-139, 2007.

[4] S. Hamdioui, Z. Al-Ars, A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories", In Proc. of IEEE VLSI Test Symposium, pp. 395-400, 2002.

[5] S. Hamdioui, et. al, "Importance of Dynamic Faults for New SRAM Technologies", In IEEE Proc. Of European Test Workshop, pp. 29-34, 2003.

[6] International SEMATECH, "International Technology Roadmap for Semiconductors (ITRS): Edition 2001"

[7] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice" ComTex Publishing, Gouda, Netherlands, 1998.

[8] A.J. van de Goor and Z. Al-Ars, "Functional Fault Models: A Formal Notation and Taxonomy", In Proc. of IEEE VLSI Test Symposium, pp. 281-289, 2000.

[9] "Xilinx ISE 6 Software Manuals and help – PDF Collection",http://toolbox.xilinx.com/docsan/xilinx7/ books/manuals .pdf.

[10] Zarrineh, K. and Upadhyaya, S.J., "On Programmable memory built-in self test architectures," Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings , 1999, pp. 708 -713.

[11] Sungju Park et al, "Microcode-Based Memory BIST Implementing Modified March Algorithms", Journal of the Korean Physical Society, Vol. 40, No. 4, April 2002, pp. 749-753 [12] 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.

[13] R. Dekker, F. Beenker and L. Thijssen, "Fault Modeling and Test Algorithm Development for Static Random Access Memories".

[14] R.Dekker, F. Beenker, L. Thijssen. "A realistic fault model and test algorithm for static random access memories". IEEE Transactions on CAD, Vol. 9(6), pp 567-572, June 1990.

[15] B. F. Cockburn: "Tutorial on Semiconductor Memory Testing" Journal of Electronic Testing: Theory and Applications, 5, pp 321-336 1994 Kluwer Academic Publishers,Boston.

[16] I.S. Irobi, Z. Al-Ars, and S. Hamdioui. Detecting memory faults in the presence of bit line coupling in sram devices. IEEE International Test Conference, 2010.

[17] Dr. R.K. Sharma Aditi Sood. "Modeling and Simulation of Microcode-based Built-In Self Test for Multi-Operation Memory Test Algorithms", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No. 2, May 2010 pp.36-40.