ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Memory Debug Technique Using March17N BIST

Ms. Zeenath
Assistant Professor in Electronic & Communication Engineering at Nawab Shah College Of Engineering & Technology(Affiliatedp to JNTUH), Malekpet, Hyderabad-500024, A.P, India
Zeenath_2009982@yahoo.com

Abstract

A Memory Debug Technique plays a key role in System-on-chip (SOC) product development and yield ramp-up. Diagnosis technique plays a key role during the rapid development of the semiconductor memories, for Catching the design and manufacturing failures and improving the overall yield and quality. Conventional failure analysis (FA) based on bitmaps and the experiences of the FA (field application) engineer are time consuming and error prone. The increasing time-to-volume pressure on semiconductor products calls for new development flow that enables and product to reach a profitable yield level as soon as possible.

This investigation on efficient diagnosis algorithms is very important due to the expensive and complex fault/failure analysis process. This MARCH based memory diagnosis algorithm which not only locate fault cells but also identify their types, using the proposed algorithm, stuck-at-faults, state coupling faults, transition faults can be distinguished. The demands in methodologies that allow FA automation thus increase rapidly in recent years. This algorithm proposes a systematic diagnosis approach based on failure patterns and functional fault models of semiconductor memories. By circuit level simulation and analysis, we have also developed a fault pattern generator. Defect diagnosis and FA can be performed automatically by using the fault patterns, reducing the time in yield improvement. The main purpose of this algorithm is for accelerating FA and yield optimization for semiconductor memories.

Keywords: March, Failure Analysis(FA), System On Chip(SOC), Debug, Stuck at Faults

Introduction

Embedded memory test design has become an essential part of the system-on-chip (SOC) development infrastructure. According to the recent ITRS report, the memory cores will occupy more than 90% of the chip area in less than ten years. The yield of on-chip memories thus will dominate the chip yield. Memory diagnostics is quickly becoming a critical issue, so far as manufacturing yield and time-to-volume of SOC products are concerned. Effective memory diagnostics and failure analysis (FA) methodologies will help improve the yield of SOC products, especially with rapid revolution in new product development and advanced process technologies. The yield loss concerned here is mainly due to wafer process defects. Failure analysis can be used to identify the defects causing the yield loss. Based on the analysis results the process can be tuned and/or the design can be modified to enhance the yield. Conventionally, an FA engineer first detects and locates the faulty cell or region, and then performs a series of physical de-processing, e-beam probing, or electron microscope inspection. However, without proper methodologies and tools it is more and more difficult to perform the defect-level testing and diagnostics, as we get into the deep-submicron age. Bitmaps and wafer maps are tools that have been commonly used in FA, because the failure patterns and distributions are helpful for the FA engineers to narrow down the potential cause of the failure, based on their experience.

Many commercial memory testers and associated yield analysis tools can be used for this purpose. The test engineers also can define a particular set of failure patterns based on the past analysis results. Automatic analysis tools will identify and locate these particular patterns in the failed memory or wafer, for subsequent identification of actual defects. Inductive fault analysis also has been used to link the defects to fault models for semiconductor memories, but the work needs continuous improvement, as new memory designs and technologies keep coming out in a fast pace. For memories, fault models are usually defined at the functional level. Based on the faulty circuit behavior, a test algorithm can be developed to detect the faults that are modeled. The quality of the test algorithm usually is determined by its length and fault

ISSN: 2277-9655
Impact Factor: 1.852

coverage. Memory fault diagnostics procedure also has been developed based on functional fault models. Alternatively, faults can be transformed into signatures, with respect to some test algorithm, and be used in the diagnostic procedure. Both the failure pattern and fault type approaches have drawbacks. There are defects causing the memory to fail with different behavior but same failure pattern relying only on that information limits the accuracy of diagnostic results. Also, the current fault models may be good for production test, but they are normally insufficient for diagnostic test that requires much more detailed fault models. So far the fault modeling still relies on the manual analysis.

In this paper, we propose a fault pattern based approach for memory diagnosis. Both bitmaps and functional fault models are used to enhance the results. With fault patterns, a failed memory can be classified and identified both by the failure patterns and the faults associated with these patterns. Similar to the previous memory fault diagnostics approach, a defect dictionary is constructed to map the actual failure response of a memory to possible defects through the fault patterns. This fault pattern based approach can easily be automated. We have developed a systematic diagnostics procedure that targets realistic memory defects, given the actual layout, particle size distribution, and process parameters. Experimental results using industrial SRAM chips justify the effectiveness of the fault patterns.

Details of Methodology

March algorithm consists of several March elements, separated by semicolon. The up-arrow stand for ascending order of address sequence, down-arrow stands for descending order. Inside the parenthesis is the specification of read writes operation and the corresponding data background. These read write operations are to be applied to each address, one by one, following the address order in front All operations inside the parenthesis have to be performed before we proceed to next address. We use the March signature to represent the results from all operations in the test algorithm, which are either correct (represented by 0) or incorrect (represented by1).We assume here that only the read operation can detect the failure. For some special memories, however write-through and write-verify operations can also detect the error

⇑ (w0); ⇑ (r0,w1,r1); ⇑ (r1,w0,r0); ⇑ (r0,w1); ⇓ (r1,w0,r0); ⇑ (r0); ⇓ (r0,w1,r1); ⇑ (r1);

Notation of March17N Algorithms

⇑	: address 0 to address n-1
⇓	: address n-1 to address 0
w0	: write 0
w1	: write 1
r0	: read a cell whose value should be 0
r1	: read a cell whose value should be 1

Architecture

The March 17N memory bist Architecture consists of 3 different blocks, i.e. BIST Register block, BIST state machine block and Memory block. Further BIST register consists of three registers BIST kickoff register, BIST status register, BIST broadcast status register. With this architecture four memories are used for better understanding however a provision for further increment of memories is provided. Each of the blocks are designed separately using a HDL code and then verified individually. After the satisfactory implementation of all the blocks, these blocks are integrated to perform the memory diagnosis operation. Each individual block is discussed in depth below.

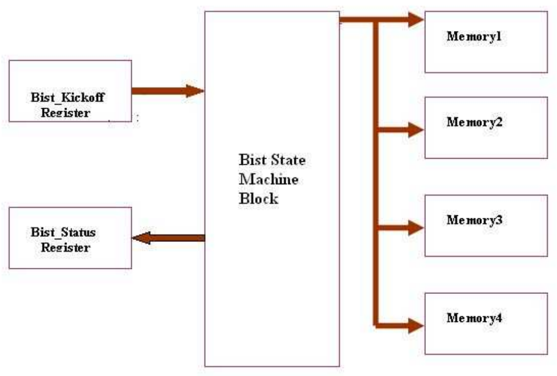

Figure 1: Memory Bist Architecture

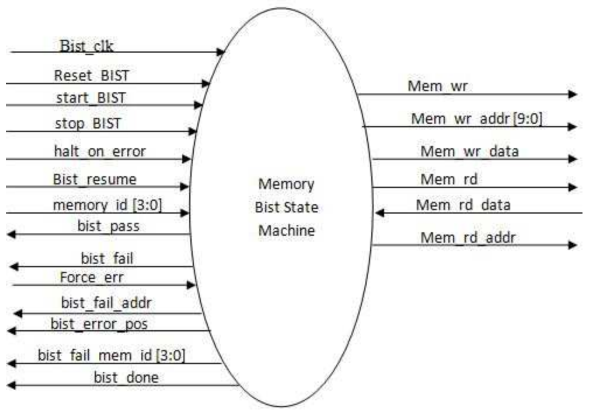

Figure 2: Memory Bist Signal Diagram and Signal Description

March 17n Memory BIST Hardware Engine

Signal Description:

Signal	Direction	Width (bits)	Description
bist_clk	Input	1	This is the Memory Bist clock input. All the internal state machines and hardware will get clocked with this clock.
bist_reset	input	1	Asynchronous reset; will reset the memory Bist state machine, and internal hardware.
bist_start	input	1	This input will trigger the Memory BIST state machine. It will be reset by the state machine when bist_done is asserted.
bist_stop	Input	1	This input is required to stop memory BIST state machine synchronously. The state machine should stop gracefully and display the status Pass/Fail. When the state machine gets bist_stop it should assert bist_done on the next clock along with the bist pass/fail status on the bist_pass/bist_fail output pins.
bist_halt_on_erro r	input	1	When this bit is programmed, Memory BIST state machine will halt itself in case of BIST failure and latch the error information (bist error addr and bist_error_pos) onto the appropriate output pins. The state machine can only come out of halt when bist_resume or bist_stop is programmed.When this bit is not programmed and bist_start is programmed, memory BIST state machine will never halt in case of failure and will only stop when bist_done is asserted (Memory BIST operation is complete). The state machine can halt if bist_stop is programmed. Bist_resume does not have any meaning in this case.
bist_resume	input	1	This input is required to resume BIST state machine, in case when halt_on_error is programmed. Bist_resume will kick off the state machine from the point where it got the failure (address) and will resume the bist operation from that point on.Bist_resume does not have any meaning when halt_on_error is not programmed.
bist_memory_id	input	2	The input bit [1:0] selects the memory from a group of 4 memories for performing memory BIST operation.
bist_broadcast	input	1	This input selects all the memories one by one for performing Memory BIST Operation. The status of each individual memory is reported in BIST_BROADCAST_STATUS register.
bist_mem_rd_dat a	input	32 (para meter ized)	Memory Read data bus. The widths can be variable across the memories, this input has to be parameterized depending upon the width of the memories.
bist_force_error	input	1	Insert error during Memory BIST Write Operation.
bist_pass	outputt	1	This output displays the successful operation of Memory BIST Operation. This output along with bist_done signifies the full completion of Memory BIST.
bist_fail	Output	1	This output signifies memory BIST failure. If halt on error is programmed Bist state machine will halt on failure and will stop if bist_stop is programmed and continue if bist_resume is programmed. Every BIST fail will increment a fail counter by one, and bist_fail will be cleared once bist_resume is programmed. While declaring Memory BIST Pass/Fail Bist state machine will monitor bist fail counter, if the counter is a non zero value bist is failed else bist is passed. When halt on error is not programmed, bist fail counter signifies no of failures occurred during memory BIST Operation, and the last value of error will be there on the error_pos and error_addr bus of memory BIST.
bist_fail_addr	Output	12 (para meter ized)	This fail address bus hold valid only when bist_fail is asserted. This fail address bus signifies the address of the memory where a comparison failure has occurred. This bus will be cleared when bist_resume is programmed. If halt on error is programmed this bus will be updated with the fail address value every time a

			failure occurred. If halt_on_error is not programmed, bist state machine will update address bus with the fail address, and will immediately jump back to the state from where it arrived and continue BIST operation irrespective of whether bist_resume is programmed or not. It will only be cleared when bist_reset or bist_resume is programmed. Once bist fail occurs, and if bist_resume is programmed, it will jump back to the state from where it arrived and start BIST operation from the next address pointed by bist fail address.
bist_error_pos	Output	5	The value present on this four bit error_pos bus is valid only when bist_fail is asserted by the bist state machine. This 4 bit signifies the error bit position within the byte read. This bus will be cleared when bist_resume is programmed. If halt on error is programmed this bus will be updated with the fail position value every time a failure occurred. If halt_on_error is not programmed, bist state machine will update error position bus with the fail position within the byte, and will immediately jump back to the state from where it arrived and continue BIST operation irrespective of whether bist_resume is programmed or not. It will only be cleared when bist_reset or bist_resume is programmed. Once bist fail occurs, and if bist_resume is programmed, it will jump back to the state from where it arrived and start BIST operation from the next address pointed by bist fail address.
bist_fail_mem_id	Output	2	The value present on this four bit mem_id bus is valid only when bist_fail is asserted by the bist state machine. This 2 bit signifies the failed memory among a group of 4 memories. This bus will be cleared when bist_resume is programmed. If halt on error is programmed this bus will be updated with the fail memory id value every time a failure occurred. If halt_on_error is not programmed, bist state machine will update fail memory id bus with the fail id, and will immediately jump back to the state from where it arrived and continue BIST operation irrespective of whether bist_resume is programmed or not. It will only be cleared when bist_reset or bist_resume is programmed. Once bist fail occurs, and if bist_resume is programmed, it will jump back to the state from where it arrived and start BIST operation from the next address pointed by bist fail address.
bist_done	Output	1	This bit signifies the completion of Memory BIST Operation. Memory BIST pass/fail will be declared only when bist_done is high. Bist_done will be asserted when Memory BIST state machine completes the final state of bist operation. It will also be asserted a clock cycle later than bist_stop is programmed.
bist_mem_wr	Output	1	This is memory write enable signal. All the writes to the memory are only valid when this signal is high.
bist_mem_wr_addr	Output	12 (parameterized)	This is memory write address.
bist_mem_wr	Output	1	This is memory write enable signal. All the writes to the memory are only valid when this signal is high.
bist_mem_rd	Output	1	This is memory read enable signal. Data on the read data bus is valid one clock cycle later after bist_mem_rd is asserted.
bist_mem_rd_addr	Output	12 (parameterized)	

Memory BlockDiagram

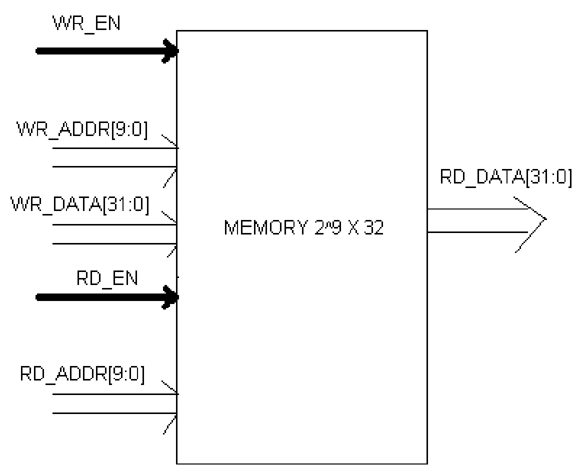

Figure 3: Memory Block Diagram

Memory is designed by writing a HDL code in verilog, the above schematic is the expected block of memory that the tool should generate. wr_en is a write enable signal to the memory which when activated starts the writing process in the memory , wr_addr is a signal vector of 10 bits which can locate upto 2^10 memory locations, Wr_data is a signal vector which consists of the data to be written on the specified location, Rd_en is a enable signal which when activated starts the read process from the specified location pointed by Rd_addr signal vector, Rd_data is the output from the memory which is the input for the state machine providing the data from the specified locations. The verilog codesigned it is simulated using simulation tool [MODELSIM] to generate the waveform which is shown below. After thorough simulation the memory module is synthesized using synthesis tool [Xilinx ISE] which generates the synthesis report and the hardware to be targeted on the board de for memory and its test bench are shown below. Once the memory module is designed it is simulated using simulation tool [MODELSIM] to generate the waveform which is shown below. After thorough simulation the memory module is synthesized using synthesis tool [Xilinx ISE] which generates the synthesis report and the hardware to be targeted on the board.

March Related Algorithm

1.A Microcode-based Memory BIST Implementing Modified March Algorithm:

The memory BIST implements march algorithms, which are slightly modified by Adopting DOF(Degree of Freedom) concept to detect ADOFs(Address Decoder Open Faults) on top of conventional stuck faults. Among the different testing algorithms ranging from O(*n*) to O(*nlog(n)*), BIST(Built-In Self Test) techniques with O(*n*) to O(*n*) complexity algorithms are widely adopted for embedded memories. Memory test patterns can be generated randomly through either test equipment or a BIST circuitry

2. A March-Based Fault Location Algorithm for SRAM:

A March-based fault location algorithm is proposed for repair of word-oriented Static RAMs. A March CL algorithm of complexity 12N, N is the number of memory Words , is defined for fault detection and partial diagnosis. A 3N or 4N March-like algorithm is used for location of the aggressor words of inter-word state, idem potent, inversion, write-disturb coupling faults (CF). Then another March-like algorithm of complexity 9(1+logB), B is the number of bits in the word, is applied to locate the aggressor bit in the aggressor is applied to locate aggressor bit in the aggressor word.

Conclusion and Future Work

I have proposed a fault-pattern oriented methodology for semiconductor memory defect diagnostics, which greatly reduces the effort in memory testing and provides profitable yield. The proposed notion of fault pattern combines the strengths of the conventional failure-pattern approach and our previous fault-type approach for easier isolation of real defects.

Since the fault patterns have to be customized for different products and process technologies, automation is very important. I have also developed a systematic procedure to explore the fault patterns, which includes a layout-based defect injection tool that provides very accurate results from realistic defect models. With the proposed approach, high-quality defect diagnostics can be automated. An industrial case has been shown to justify the work. The main contribution of the paper is thus a methodology and procedure for accelerating FA and yield optimization for semiconductor memories.

This algorithm (March 17N) is much faster compared to other March based algorithms. It also used to ramp up the yield of company. As compared to other March based algorithm this is dominant and covers the faults of semiconductor memories accurately. We can detect the faults of memories with optimized design and less effort made by the designers.

This paper not only increases the speed of detecting the faults but also reduces the cost of the paper with optimized design. It covers all kinds of faults and guarantees of fault free memories in SoCs

ISSN: 2277-9655
Impact Factor: 1.852

At this stage I have simulated the bist architecture for only four memories but it can be upgraded to any number of memories depending upon the requirement and target board availability, the future up gradation is possible because we have kept few bits reserved for future up gradation in the bist_kickoff register and bist_status register. Also this project is the very first step for the new technology called BISR (built in self repair).

References

[1] Chih-Wea Wang, Kuo-Liang Cheng, Jih-Nung LeeFault. Pattern Oriented Defect Diagnosis for Memories,2003.

[2] Hans-Joachim Wunderlich, University of Stuttgart, BIST FOR SYSTEMS- ON-A-CHIP,1999.

[3] Dongkyu Youn, Taehyung Kim and Sungju Park Dept. of Computer Science & Engineering Hanyang University, A Microcode-based Memory BIST Implementing Modified March Algorithm, ,SaDong, Ansan, Kyunggi-Do, 425-791 Korea .2002.

[4] Y. E. Hong, L. S. Leong,W. Y. Choong, L. C. Hou, and M. Adnan, "An overview of advanced failure analysis techniques for Pentium and Pentium Pro microprocessors", Intel Technology Journal, , no. 2, 1998.

[5] J. T. Chen, J. Rajski, J. Khare, O. Kebichi, and W. Maly, "Enabling embedded memory diagnosis via test response compression", in Proc. IEEE VLSI Test Symp. (VTS), Marina Del Rey, California, Apr. 2001, pp. 292–298.