Memory BIST

Ajay V. Tadiparti

Abstract

With the advent of deep-submicron VLSI technology, core-based system-on-chip (SOC) design is attracting an increasing attention. On an SOC, popular reusable cores include memories (such as ROM, SRAM, DRAM and flash memory), processors (such as CPU, DSP and microcontroller), input/output circuits, etc. Memory cores are obviously among the most universal ones-almost all system chips contain some type of embedded memory [5]. However, to provide a low cost-cost test solution for the on-chip memory cores is not a trivial task. This paper presents a study on memory BIST, algorithms of different test patterns, survey of memory BIST implementations, and discussion of some novel design issues. This paper shall serve as a knowledge base for future design in memory BIST.

1. Introduction

Digital systems are composed of data paths, control paths and memories. Defects in memory arrays are generally due to shorts and opens in memory cells, address decoder and read/write logic. These defects can be modeled as single and multicell memory faults. The dominant use of embedded memory cores along with emerging new architectures and technologies make providing a low cost test solution for these on-chip memories a very challenging task. The addition of extra circuitry to facilitate testing of memory chips, called design for testability (DFT), or to allow the test mechanism to be completely contained within the chip, called built-in self-test, has only recently become of interest. Memory BIST has been proven to be one of the most cost-effective and widely used solutions for memory testing for the following reasons: no external test equipment, Reduced development efforts, tests can run at circuit speed to yield a more realistic test time, on-chip test pattern generation to provide higher controllability and observability, On-chip response analysis, test can be on-line or off-line, adaptability to engineering changes, easier burn-in support. This paper presents algorithms for different test patterns, surveys of current memory BIST architecture, and discussion of various implementation issues. The paper is organized as follows: Section 3 describes the BIST algorithms of various memory tests. Section 4 present different memory BIST implementation schemes and their design trade-offs. Novel design aspects in memory BIST are presented in section 5. The conclusions are offered in Section 6.

I would like to thank Prakash Vippa (peer reviewer) for his corrections and valuable suggestions.

2. Background

The two types of BIST are On-line BIST and Off-line BIST. On-line BIST has tests implemented on-chip. It has shorter test time but an area overhead of one to three percent. Off-line BIST, on the other hand has tests implemented off-chip. It has longer test time but no area overhead. On-line BIST can further be classified into three subgroups: Concurrent BIST, Non-
Concurrent BIST and Transparent BIST. Concurrent BIST is performed during normal use of the chip. This means that faults occurring during normal use can be detected and, depending on the sophistication of the test, even be corrected. A non-concurrent test is not active during normal use of the chip but only in test mode. The advantage of this form of BIST is that the test does not have to preserve the data which is stored on the chip, thus allowing a maximum freedom in the test data to be used. The disadvantage is the circuit cannot detect faults that are not covered by the fault models used. Transparent BIST scheme is very similar to the Non-Concurrent scheme except the memory contents are preserved.

With memory BIST, the entire memory testing algorithm is implemented on-chip, and operates at the speed of the circuit, which are 2 to 3 orders of magnitude faster than a conventional memory test. Most memory BIST schemes exploit the parallelism within the memory device to achieve a massive reduction in test time (and therefore cost). This is done by a test mode where more than one memory cell is accessed with each address, usually by

accessing the entire row of cells on a word line for a single read or write operation. For n cells in the memory, with $\sqrt{n}$ rows $\sqrt{n}$ columns, this reduces test time by a $\sqrt{n}$ factor.

3. Test Algorithms

A test algorithm is a finite sequence of test elements. A test element contains a number of memory operations, data pattern specified for the read operation, address specified for the read and write operations. We divided these test algorithms into two groups: Traditional tests and March-based tests. The following are the well known traditional tests: zero-one, checkerboard, GALPAT and walking 1/0, sliding diagonal and butterfly. These are either simple, fast but have poor fault coverage or have good fault coverage but complex and slow. Due to these imbalanced conflicting traits, the popularity of these algorithms is decreasing. A March-based test algorithm is a finite sequence of March elements. A March element is specified by an address order and a number of reads and writes. Examples of some March-based tests are MATS, MATS+, Marching 1/0, March C-, March Y, March A, March B, and etc.. Since March-based tests are all simple and possess good fault coverage, they are the dominant test algorithms implemented in most modern memory BIST.

4. Memory BIST Implementations

Different memory BIST implementation schemes are described in this section. Memory BIST consists of address generator, test pattern generator, and BIST control logic. The BIST controller can be implemented by either microcode, hardwired logic, or processor-based.

4.1. Microcode-based BIST

Predefined instructions, or microcode is used to write the selected test algorithms. The written tests are loaded in the memory BIST controller. This microcode-based type of memory BIST allows changes in the selected test algorithm with no impact on the hardware of the controller. This flexibility, however, may come with the cost of higher logic overhead for the controller. A recent microcode-based memory BIST implementing modified march algorithm was proposed in [3]. This design requires only one third of microcode storages of others. The proposed memory BIST can be widely used for the embedded memory testing, especially under SOC design environment because of its superior flexibility and extendibility in applying different combination of memory test algorithms.

4.2. Processor-based BIST

An on-chip microprocessor is utilized to test the memory cores [6]. The processor-based memory BIST is done by executing an assembly-language program in the on-chip microprocessor to generate test patterns including the address sequence, data patterns, and control signals. The memory outputs are then compared with the expected correct data. The advantage of such a scheme is that it is highly flexible because various test algorithms can be realized by simply modifying the assembly programs run on the microprocessor and it is also easy to support testing of multiple memory cores. However, the drawback is a much longer test time.

The BIST core is inserted between the CPU core and the on-chip bus, which also connects the memory cores. In normal operation mode, the CPU transparently accesses the system bus with slight time overhead introduced by the multiplexers. The overhead can be minimized by careful design of the multiplexers which can be integrated with the bus drivers. In memory BIST mode, the BIST circuitry takes over the control of the on-chip bus. It executes certain test algorithm programmed by the CPU and generates the addresses, input data, and control signals of the memory core. It also compares the memory output response with the expected correct data. In order to allow these two different modes, several multiplexers are used to multiplex the address bus, data input bus, data output bus, and control bus between the CPU core and the BIST circuitry. Other blocks in the BIST circuit include: (1) an address counter which generates the test address sequence; (2) a comparator which compares the memory output response with the expected correct data; and (3) a BIST controller which controls the BIST circuit.

4.3. Hardwired-based BIST

A hardwired-based controller is a hardware realization of a selected memory test algorithm, usually in the form of a Finite State Machine (FSM). This type of memory BIST architecture has optimum logic overhead, however, lacks the flexibility to accommodate any changes in the selected memory test algorithm. This results in re-design and re-

implementation of the hardwired-based memory BIST for any minor changes in the selected memory test algorithm. Although it is the oldest memory BIST scheme amongst the three, hardwired-based BIST is still much in use and techniques have been kept developing.

Scheme	Test Time	Area OH	Routing OH	Flexibility
Hardwired	short	low	high	zero
Microcode	average	high	low	low
Processor	long	zero	zero	high

Table 1: Trade-offs between Different Memory BIST Schemes [4]

Table 1 gives the summary of this comparison of the three implementations. The four evaluation metrics used are: test time, area overhead, routing overhead, and flexibility. The routing overhead is directly translated into design efforts and time to market. The flexibility is expressed in terms of programmability of algorithms and adaptability to engineering changes. As shown in the table, the Hardwired-based BIST is fast, compact, incur the most design efforts and possesses the least flexibility. On the opposite end of spectrum, the Processor-based BIST is the most flexible, zero area or routing overhead, but incur long test time. The Microcode-based designs is somewhere in between these two extreme prototypes.

5. Novel Design Aspects for Memory BIST

This section presents the novel design aspects for modern memory BIST. These various innovations took different points of view to improve the design of memory BIST including change in algorithm, add-on technique, and renovated architecture. The summary of four papers ([7],[8],[9],[10]) are presented in details in the following subsections.

5.1. Defect Oriented Analysis

In this design, the defect coverage is used as the main idea rather than fault coverage, as a new metric to measure the quality of the memory test algorithms [7]. As there is no linear correspondence between fault coverage and defect coverage, and defect coverage provides a better estimate for overall test quality in terms of defects-per-million of shipped parts; therefore, test engineers should measure the quality of memory tests by their defect coverage rather than fault coverage. This paper ([5]) shows defect-oriented metric as a better measure for the quality of memory tests. While fault coverage is useful to the test engineers in the process of design and development of tests, because there is no linear correspondence between fault coverage and defect coverage, thus fault coverage is not a good indicator of test quality. Defect coverage, on the other end, provides a better estimate for overall test quality in terms of defects-per-million of shipped parts.

5.2. Integrated Diagnostics of Embedded Memory

The main idea of this design methodology is to integrate full diagnostic capabilities into existing memory BIST designs to allow both effective and efficient manufacturing test by isolating faults incurred during the manufacturing or design process. The technique improves the overall satisfiability of the cost, quality, and time to market requirements of embedded memory and is used for a wide range of PowerPC™ microprocessors [8]. The author uses 6-transistor SRAM designs to illustrate this technique and he claims that it can also be used in other types of volatile and nonvolatile memories. Integration of diagnostics into a memory BIST is a simple extension to the existing logic. Addition of few simple controls and observation points can enhance a memory BIST, to achieve all manufacturing, design and diagnostics needs, without compromising any of the other requirements.

5.3. Automatic Generation

The objective of is to minimize test effort for embedded memories in SOC by enabling automatic test integration of multiple and heterogeneous memory cores in a SOC environment. The paper [9] presents an automatic generator for memory BIST circuits called BRAINS (BIST for RAM in Seconds). It has a graphic user interface and is integrated with a memory compiler to form an IP generator for various memory configurations. It also features an automatic test grouping and scheduling which can optimize the overhead in test time, performance, and power consumption. With a configurable and extensible architecture, the proposed framework facilitates easy memory test integration for core providers as well as system integrators. The automatic generation of memory BIST cores extends the ability of system-level test integration, multi-port and multi-memory support,

and test grouping and scheduling for parallel testing. The BIST access can be parallel for easy test control, or serial for simple IO interface, depending on the test requirement. The proposed BRAINS generates configurable and extensible BIST circuits, and performs test grouping and scheduling, as well as overall test planning of embedded memory cores for SOC designs in a cost-effective way.

5.4. Built-in Self-Repair using Redundant Words

The design [10] proposes a word oriented memory Built-in Self-Repair methodology (BISR) that targets on embedded SRAM and does not rely on spare rows and columns. It uses BIST logic to identify faulty words and redundancy logic to store faulty addresses and data immediately after its detection during test; wrapper logic to replace defect words. This paper presents a new memory built-in self-repair concept that uses spare words instead of spare columns and rows. It uses redundancy logic to perform a software repair as long as spare words are available when a MBISR test finds failures. Permanent storage of those failing addresses in the field is possible with electrical or field
programmable fuses.

5. Conclusions

Memory testing is very important but challenging. Memory BIST is considered the best solution due to various engineering and economic reasons. March tests are the most popular algorithms currently implemented in BIST hardware. Various implementation schemes for memory BISTs are presented and their trade-offs are discussed: A Hardwired-based BIST is fast and compact, whereas a Processor-based BIST cost near zero hardware overhead and very flexible. The future Memory BIST designs should be fast, small, efficient, robust, and flexible. Using memory BIST has various advantages much as no external test equipment, reduced development efforts, at-speed tests. However, there are many challenges associated with it such as silicon area overhead, extra pins and routing. In addition, the testability of the test hardware itself is another difficult task. A future memory BIST tester should have very short test time since this ultimately determines the final testing cost. We need high-quality test algorithms, those with high coverage which has high correlation to yield. The hardware needs to be simple since this determine the speed and area overhead of the tester.

6. References

[1] Essentials of Electronic Testing for Digital, memory & Mixed-Signal VLSI Circuits, Michael L Bushnell and Vishwani D. Agrawal, Boston: Kluwer Academic Publishers, 2000.

[2] Testing semiconductor memories Theory and Practice, A.J. Van de Goor, Wiley Publications, 1991.

[3] Dongkyu Youn, Taehyung Kim, Sungju Park, “A microcode-based memory BIST implementing modified march algorithm,” Asian Test Symposium, 2001. Proceedings. 10th, 2001, pp. 391-395

[4] Allen C. Cheng, “Comprehensive Study on designing Memory BIST: Algorithms, implementations and Trade-offs”, EECS 579, University of Michigan Fall 2002

[5] Ching-Hong Tsai, and Cheng-Wen Wu, “Processor-programmable memory BIST for bus-connected embedded memories,” Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific, 2001, pp. 325 -330

[6] 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

[7] Jee, A., “Defect-oriented analysis of memory BIST tests,” Memory Technology, Design and Testing, 2002. (MTDT 2002)

[8] Hunter, C., “Integrated diagnostics for embedded memory built-in self test on PowerPC™ devices,” IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1997. (ICCD 1997). Proceedings, 1997. pp. 549-554

[9] Kuo-Liang Cheng, Chia-Ming Hsueh, Jing-Reng Huang, Jen-Chieh Yeh, Chih-Tsun Huang, Cheng-Wen Wu, “Automatic generation of memory built-in self-test cores for system-on-chip,” Asian Test Symposium, 2001. Proceedings. 10th, 2001, pp. 91-96

[10] Schöber, V.; Paul, S.; Picot, O., “Memory built-in self-repair using redundant words,” International Test Conference, 2001. (ITC 2001). Proceedings, 2001. pp. 995 -1001