Low-Cost Self-Test Techniques for Small RAMs in SOCs

P. Vijaya Prasada Rao¹, P.Jaya Rami Reddy²

¹II year M.Tech Student, Dr. K.V. Subba Reddy Institute of Technology, Kurnool, INDIA
²Assistant Professor, E.C.E. Department, Dr. K.V. Subba Reddy Institute of Technology, Kurnool, INDIA

ABSTRACT

This paper proposes an enhanced IEEE 1500 test wrapper to support the testing and diagnosis of the single-port or multi-port RAM core attached to the enhanced IEEE 1500 test wrapper without incurring large area overhead to small memories. Effective test time reduction techniques for the proposed test scheme are also proposed.

keywords: IEEE 1500 test, multi-port RAM, IC, SOC

I. INTRODUCTION

These days, electronics are used everywhere. We find electronics in dish-washers, electronic stoves, children’s toys etc. All electronics consist of Integrated Circuits (ICs), and ICs are getting more advanced. With more advanced ICs follows increasing development times. Module based design, where blocks of logic (modules) are developed separately, saves time, and therefore it is very attractive. The manufacturing of ICs are not flawless, therefore every IC needs to be tested. If an IC is module based, it can be tested in a modular fashion. For a module based test a test wrapper to enable test access and test isolation are required.

In this thesis we study the IEEE 1500 Standard Testability Method for Embedded Core-based Integrated Circuits (IEEE 1500). This standard makes it possible to use modular testing to test module based ICs. This thesis begins with an introduction, followed by a problem definition. The next chapter gives the complete information on BIST and different fault models. After that follows a chapter which gives a brief explanation of the 1500 standard with its components according to the standard document. The following chapter describes the IEEE 1500 components and the wrapper in detail used in this thesis. An experimental results chapter follows; viewing the simulation results of all the registers of IEEE 1500 and different modules experimented in it. Then conclusions and future work on the low cost self test techniques for small RAM’s in SoC with the enhanced IEEE 1500 test wrapper concludes the thesis.

II. SCOPE AND DESIGN

IEEE Std 1500™ defines a scalable architecture for independent, modular test development and test application for embedded design blocks and also enables test of the external logic surrounding these cores. Modular testing is typically a requirement for embedded non-logic blocks, such as memories, and for embedded pre-designed non mergeable intellectual property (IP) cores. In addition, the IEEE 1500 architecture can also be used to partition large design blocks into smaller blocks of more manageable size and to facilitate test reuse for blocks that are reused from one system-on-chip (SoC) design to the next.

The IEEE 1500 architecture comprises hardware requirements, through the definition of a standardized core wrapper, and uses a test-specific language to communicate information between core providers and core users. This language is the IEEE P1450.6™ core test language (CTL). Although IEEE Std 1500 limited itself to test aspects internal to nonmergeable cores, careful consideration was given to the interoperability of such cores, resulting in plug-and-play (PnP) requirement definitions. SoC-specific issues such as those related to the design of test access mechanisms (TAMs) are excluded from this standard and assumed to be addressed by the SoC designer.

IEEE Std 1500 specifically focuses on defining test requirements for unidirectional non-tristate digital terminals, as these represent a minimum and mandatory set of requirements upon which the more complex bidirectional terminals are based. It is, therefore, implied that support for bidirectional or tristate terminals is
III. HARDWARE DESIGN DESCRIPTION

3.1. IEEE 1500 scalable hardware architecture:

The IEEE 1500 core wrapper comprises the following:

- Serial interface terminals forming the WSP
- A user-defined set of wrapper terminals forming the wrapper parallel port (WPP) and providing parallel access to the wrapper
- A WIR
- A WBY
- A WBR

3.2 Block Diagram:

- WSC, WIR, WBY, WSI, and WSO denote the wrapper serial control port, wrapper instruction register, wrapper bypass register, wrapper serial input, and wrapper serial output.
- The WSC port contains six mandatory control signals, WRCK, WRSTN, Select WIR, Capture WR, Shift WR, UpdateWR, and one optional control signal, TransferDR.
- Control signals and I/Os of a RAM typically consist of a chip enable (CEN), a Read/Write enable (WEN), addresses (ADDR), data inputs (DI), and data outputs (DO) signals.

IV. IMPLEMENTATION

4.1 MARCH Algorithms

March Test Notation A March test consists of a finite sequence of March elements. A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell. For example, + (r1,w0) is a March element and r0 is a March primitive. The address order in a March element can be increasing (*), decreasing (+), or either increasing or decreasing (m). An operation can be either writing a 0 or 1 into a cell (w0 or w1), or reading a 0 or 1 from a cell (r0 or r1).

In summary, the notation of March test is described as follows:

<table>
<thead>
<tr>
<th>Name of the Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATS {m (w0); m (r0,w1); m (r1)}</td>
<td>Increasing memory addressing order;</td>
</tr>
<tr>
<td>MATS+ {m (w0); * (r0,w1); + (r1,w0)}</td>
<td>Increasing memory addressing order;</td>
</tr>
<tr>
<td>MATS++ {m (w0); * (r0,w1); + (r1,w0, r0)}</td>
<td>Increasing memory addressing order;</td>
</tr>
</tbody>
</table>
MARCH X \{ m (w0); * (r0,w1); + (r1,w0); m (r0) \}
MATCH C \{ m (w0); * (r0,w1); * (r1,w0); + (r0,w1); + (r1,w0); m (r0) \}
MATCH A \{ m (w0); * (r0,w1,w0,w1); * (r1,w0,w1); + (r1,w0,w1,w0); + (r0,w1,w0) \}
MATCH Y \{ m (w0); * (r0,w1, r1); + (r1,w0, r0); m (r0) \}
MATCH B \{ m (w0); * (r0,w1, r1,w0, r0,w1); * (r1,w0,w1); + (r1,w0,w1,w0); + (r0,w1,w0) \}

Table: Irredundant March Test Algorithms

| r0 | Read 0 from a memory location |
| r1 | Read 1 from a memory location |
| w0 | Write 0 to a memory location  |
| w1 | Write 1 to a memory location  |

March Test Algorithms Table lists several relevant March algorithms reported in the literature. Table gives the fault coverage and the operation count of these March algorithms, which are also called irredundant algorithms (by removing any operation from the test, the targeted fault coverage will be reduced). To generate custom March algorithms for improved defect coverage in new process technologies, an effective methodology was proposed. March algorithms are very easy to implement in either software or hardware.

Modified algorithmic test sequence (MATS) \((w0); (r0, w1); (r1)\):
- S1: write 0 to all cells.
- S2: for each cell read 0; write 1.
- S3: read 1 from all cells.

4.2 MBIST Implementations

A memory BIST unit consists of a controller to control the flow of test sequences and other components to generate the necessary test control and data. The various types of memory BIST are categorized according to the schemes of their controllers. Designs of a memory BIST controller could be roughly classified into three different types: (1) a hardwired based, (2) microcode-based, and (3) processor-based. The following three subsections give the specifics of each scheme.

4.2.1 Hardwired-based MBIST

A hardwired-based controller is a hardware realization of a selected memory test algorithm, usually in the form of a 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.

4.2.2 Microcode-based MBIST

A microcode-based memory BIST features a set of predefined instructions, or microcode, which 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. The approach of this microcode-based design is to focus on capturing Address Decoder Open Faults (ADOF) and detecting some NPSFs in addition to the conventional SF, TF, CF, DRF faults. Figure 2.8 illustrates a microcode-based memory BIST which consists of

---

466 Copyright © 2016. Vandana Publications. All Rights Reserved.
storage unit, instruction counter, instruction decoder, LHCA address generator, and comparator. The storage unit is a 6x10 bits ROM that stores the conventional march test algorithms. The instruction counter is a \( \log_2(X)+1 \) bit binary up-down counter which selects the instruction address of the ROM. The instruction decoder generates the up-down address and hold/enable signals by taking the instruction condition bits and LHCA terminal signals. The up down LHCA is a randomly inverted LHCA that is generated to generate 2000 independent random address pattern in hope to provide better NPSF coverage. Finally, the comparator produces an error signal by comparing the test data and RAM output data.

4.2.3 Processor-based MBIST

To provide a low cost test solution for the on-chip memory cores is a challenging task. Conventional hardwired-based MBIST approach is one possible solution and its advantages are short test time and small area overhead. However, sometimes it is not feasible to have one BIST circuit for each memory core. If each memory core on chip requires a BIST circuit, then the area and test pin overhead will be unacceptably high. Therefore, a new type of MBIST scheme which utilizes an on-chip microprocessor to test the memory cores was proposed. The processor-based MBIST 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 BIST core is inserted between the CPU core and the onchip 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. 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 multiple level.

4.2.4 Programmable MBIST:

Most of memory bist approaches concerns the programmability of the memory test algorithm. To enable programmability of all components of memory test, test algorithm, test data, address sequence.

The architecture for programming march test algorithms proposed in the fig. This architecture uses an instruction register specifying the current match test sequence by means of several fields indicating. The block diagram of the programmable MBIST is as shown in the following figure 4.4

Figure 4.4 Block diagram of programmable MBIST

In the above block diagram, programmable MBIST contains 4 important parts.
1. BIST controller
2. Counter
3. Comparator
4. Address memory

4.3 BIST Controller

Built-in self-test (BIST) is a design technique that allows a circuit to test itself. It is a set of structured-test techniques for combinational and sequential logic, memories, multipliers and other embedded logic blocks. The principle is to generate test vectors, apply them to the circuit under test or device under test, and then verify the response. Being an automated testing, BIST enables testing at high speed and high fault coverage. BIST controller coordinates the operations of different blocks of the BIST. Based on the test mode(TM) input to the controller, the system either operates in the normal mode or in the test mode. In this paper we explain an implementation of a restart able logic BIST controller for a combinational logic circuit using VHDL. It allows us to suspend the signature generation at any desired point in the test sequence. In this case, the BIST circuit is considered to comprise hold logic and a signature generation element. The hold logic will be implemented such that an external signal (HOED) can temporarily suspend signature generation in the signature generation element at specified times during the BIST session.

4.4 Counter

A counter that can change state in either direction, under the control of an up/down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value and when the selector is in the down state, the counter decrements the count. A ring counter is a circular shift register which is initiated such that only one of its -flops is the state one while others are in their zero states. A ring counter is a Shift Register (a cascade connection of flip-flops) with the output of the last one connected to the input of the first, that is, in a ring. Typically, a pattern consisting of a single bit is circulated so the state repeats every n clock cycles if n flip-flops are used. It can be used as a cycle counter of n states. A Johnson counter (or switch tail ring counter, twisted-ring counter, walking-ring counter, or Amoebas counter) is a modified ring counter, where the output from the last stage is inverted and fed back as input to the first stage. The register cycles through a sequence of bit-patterns, whose length is equal to twice the length of the shift register, continuing indefinitely. These counters find specialist applications, including those similar to the decade counter, digital-to-analog conversion, etc. They can be implemented easily using D- or JK-type flip-flops.
4.5 Comparator

A method, apparatus and system of a self-test output for high density BIST are disclosed. In one embodiment, an integrated circuit includes one or more memories, a BIST controller coupled to the one or more memories to perform write operation and to receive a PASS/FAIL signal from each embedded memory and one or more comparators coupled to the one or more memories latch mutually identical outputted data coming from the memories upon a rising edge of an ORDY signal. In addition, the comparators may compare the latched mutually identical outputted data and output associated PASS/FAIL signal to the BIST controller. The BIST controller registers the received PASS/FAIL result upon receiving the PASS/FAIL signal from the comparators.

4.6 Address memory

In computing, memory address is a data concept used at various levels by software and hardware to access the computer's primary storage memory. Memory addresses are fixed-length sequences of bits conventionally displayed and manipulated as unsigned integers. Such numerical semantic bases itself upon features of CPU, as well as upon use of the memory like an array endorsed by various languages. In other words, a computer and even one program in it may have several different memory address spaces. A digital computer's memory, more specifically main memory, consists of many memory locations, each having a physical address, a code, which the CPU (or other device) can use to access it. Generally only system software, i.e. the BIOS, operating systems, and some specialized utility programs (e.g., memory testers), address physical memory using machine code operands or processor registers instruct the CPU to direct a hardware device, called the memory controller, to use the memory bus or system bus, or separate control, address and data busses, to execute the program's commands. The memory controllers' bus consists of a number of parallel lines, each represented by a binary digit (bit). The width of the bus, and thus the number of addressable storage units, and the number of bits in each unit, varies among computers and so on.

V. SIMULATION RESULTS

5.1 Figure: boundary register
5.2 Figure: comparator

5.3 Figure: top

5.4 Figure: top_with_faults
5.5 Figure: top_with_go_no_go

5.6 Figure: top_with_go_no_go_faults
VI. CONCLUSION

An enhanced IEEE 1500 test wrapper has been presented to test distributed embedded memories in SOC. It extends the IEEE 1500 test wrapper to support march tests for small and different types of memories. Also, effective test time reduction techniques for RAMs with the proposed test wrapper have been proposed.

REFERENCES