**Built-In-Self-Repair for Embedded RAMs with Efficient Fault Coverage Using PMBIST**

L. Dharma Teja, K. Kiruthika and V. Priyanka Brahmaiah  
Department of Electronics and Communication Engineering,  
VNR Vignana Jyothi Institute of Engineering & Technology,  
JNT University, Hyderabad, India

**ABSTRACT**  
Built-In Self-Repair (BISR) with Selectable Redundancy is an effective yield-enhancement strategy for embedded memories. This paper proposes an efficient BISR strategy which consists of a Programmable Memory Built-In Self-Test (PMBIST) module, a Built-In Address-Analysis (BIAA) module and a Multiplexer (MUX) module. Programmable Memory BIST is used to run variety of algorithms without the need of dedicated BIST hardware for each algorithm. It gives flexibility to accommodate new test algorithms developed for newly identified defects. In BIAA module, fault addresses and redundant ones form a one-to-one mapping to achieve a high repair speed. Besides, instead of adding spare words, rows, columns or blocks in the SRAMs, users can select normal words as redundancy. The selectable redundancy brings no penalty of area and complexity and is suitable for compiler design. A Verilog code is written for 4KX32 RAM model with BISR circuit and implemented on FPGA SPARTAN3E.

**KEYWORDS:** Built-In Self-Repair (BISR), Programmable Memory Built-In Self-Test (PMBIST), Built-In Address-Analysis (BIAA).

**I. INTRODUCTION**

As current System-on-Chip (SOC) designs become memory intensive, the manufacturing yield of such devices greatly depend on the yield of embedded memories. Smaller feature sizes and increasing real estate occupied by memories on a chip is resulting in an enormous critical area that is conducive to a large number of potential defects. According to the ITRS 2000 Roadmap the area occupied by embedded memories in System-on-Chip (SoC) is over 90%, and expected to rise up to 94% by 2014 [1]. Thus, the performance and yield of embedded memories will dominate that of SoCs. However, memory fabrication yield is limited largely by random defects, random oxide pinholes, random leakage defects, gross processing and assembly faults, specific processing faults, misalignments, gross photo defects and other faults and defects [2]. Many fault models and test algorithms have been developed to detect defects in a memory[17]. Some of these algorithms are already being used extensively in practice while some others are being adopted rapidly to improve defect coverage. In addition, BIST has become an attractive alternative for implementing such algorithms, offering the benefits of high fault coverage, full speed test application, extensive diagnostics, and on-chip test hardware thereby eliminating the need for sophisticated ATE.  
With the advent of deep-submicron VLSI technology, the memory density and capacity is growing. The clock frequency is never higher. 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. Built-in self-test (BIST) has been proven to be one of the most cost-effective and widely used solutions for memory testing for the following reasons: (1) No external...
test equipment; (2) Reduced development efforts; (3) Tests can run at circuit speed to yield a more realistic test time; (4) On-chip test pattern generation to provide higher controllability and observability; (5) On-chip response analysis; (6) Test can be on-line or off-line; (7) Adaptability to engineering changes; (8) Easier burn-in support.

Traditionally, BIST controllers are capable of running the test algorithms in a pre-specified sequence during manufacturing test. Since these controllers are customized and hardwired for a given memory architecture and a pre-defined set of algorithms, the area overhead is low and tests can be applied at-speed. One of the major disadvantages of hardwired controllers is their limited flexibility. With rapid changes in technology, it is becoming extremely difficult to predict the defect types that could manifest during the manufacturing process. Similarly, often a chip or its later revisions stay long enough to be manufactured with different technology nodes. Under such circumstances, test algorithms that would work well for a certain technology for which the controller was designed may no longer work when manufactured using the next technology node. Consequently, there is an increasing need for a programmable BIST solution that would allow certain degree of flexibility to modify test programs at run time. This would help analyzing the failing parts during production test, identifying of the defects that are escaping, re-designing the existing algorithms or introducing new algorithms, and use it to reduce the test escapes. It also aids in failure analysis thereby speeding up the ramp up period of a new process.

Specifically, the following summarizes some of the advantages of a programmable BIST controller:

- Flexibility to create different data backgrounds, addressing schemes, and test algorithms.
- Flexibility to accommodate new test algorithms developed for newly identified defects. This helps in improving the test quality, which is specially useful for military, medical, and automotive applications.
- Helping failure analysis thereby expediting the yield learning period of a fabrication process.
- Allows better management of test time as different sets of algorithms can be applied at different phases of test, such as wafer sort, burn-in test, parametric, package, etc.

This paper proposes an efficient BISR strategy which can store each fault address only once and PMBIST which gives efficient fault coverage and flexibility to accommodate new test algorithms developed for newly identified defects. The rest of this paper is organized as follows. Section II gives related work. Section III outlines SRAM fault models, test algorithms and BIST design. Section IV introduces the proposed BISR strategy. We present the details of the proposed BISR strategy including the architectures, procedures and the features. In section IV, the experimental results are reported. Section V concludes this paper. Finally, Section VI gives expected future work.

II. RELATED WORK

To increase the reliability and yield of embedded memories, many redundancy mechanisms have been proposed [3-6]. In [3-5] both redundant rows and columns are incorporated into the memory array. In [6] spare words, rows, and columns are added into the word-oriented memory cores as redundancy. All these redundancy mechanisms bring penalty of area and complexity to embedded memories design. Considered that compiler is used to configure SRAM for different needs, the BISR had better bring no change to other modules in SRAM. To solve the problem, a new redundancy scheme is proposed in this paper. Some normal words in embedded memories can be selected as redundancy instead of adding spare words, spare rows, spare columns or spare blocks.

Memory test is necessary before using redundancy to repair. Design for test (DFT) techniques proposed in 1970 improve the testability by including additional circuitry. The DFT circuitry controlled through a BIST circuitry is more time-saving and efficient compared to that controlled by the external tester (ATE) [7]. However, memory BIST does not address the loss of parts due to manufacturing defects but only the screening aspects of the manufactured parts [8]. BISR techniques aim at testing embedded memories, saving the fault addresses and replacing them with redundancy. In [9], the authors proposed a new memory BISR strategy applying two serial redundancy analysis (RA) stages. [10] presents an efficient repair algorithm for embedded memory with multiple redundancies and a BISR circuit using the proposed algorithm. All the previous BISR techniques can repair memories, but they didn’t tell us how to avoid storing fault address more than once.
III. FAULT MODELS, TEST ALGORITHMS AND BIST

A fault model is a systematic and precise representation of physical faults in a form suitable for simulation and test generation [11]. Applying the reduced functional model, SRAM faults can be classified as follows in [12]:

- **AF** ---- Address Fault
- **ADOF** ---- Address Decoder Open Faults
- **CF** ---- Coupling Faults
  - **CFin** ---- Inversion Coupling Faults
  - **CFid** ---- Idempotent Coupling Faults
  - **BF** ---- Bridge Coupling Faults
  - **CFst** ---- State Coupling Faults
  - **CFds** ---- Disturb coupling fault
  - **CFtr** ---- Transition coupling fault.
  - **CFwd** ---- Write Destructive coupling fault
  - **CFrd** ---- Read Destructive coupling fault
  - **CFdrl** ---- Deceptive Read Destructive coupling fault
  - **CFir** ---- Incorrect Read coupling fault
- **DRF** ---- Data Retention Faults
- **SAF** ---- Stuck-at Faults
- **SOF** ---- Stuck Open Faults
- **TF** ---- Transition Faults

The details of the fault models can be referred in [12]. They are the foundations of the memory test. As the memories grow in size and speed, the bit lines, word lines and address decoder pre-select lines will have high parasitic capacitance in addition to a high load. This increases their sensitivity for delay and timing related faults. Also, the significance of the resistive opens is considered to increase in current and future technologies. Since the partial resistive opens behave as delay and time related faults, these faults will become more important in the deep-submicron technologies [14]. Moreover, transistor short channel effect, cross talk effects, impact of process variation have to be necessarily taken into account for developing fault models for embedded memories based on newer technologies. These factors help in the development of new, optimal, high coverage tests and diagnostic algorithms that allow for dealing with the new defects. The greater the fault detection and localization coverage, the higher the repair efficiency; hence higher the obtained yield.

Thus, the new trends in Memory testing will be driven by the following items:

- **Fault modeling**: New fault models should be established in order to deal with the new defects introduced by current and future (deep-submicron) technologies.
- **Test algorithm design**: Optimal test/diagnosis algorithms to guarantee high defect coverage for the new memory technologies and reduce the DPM (Defect-Per Million level).
- **BIST**: The only solution that allows at-speed testing for embedded memories.

The well-known fault models, developed before late 1990’s could not explain the occurrence of many faults that were detected using experimental results based on DPM screening of a large number of tests applied to a large number of memory chips that were performed at that time, suggesting the existence of additional faults. This implied that new memory technologies involving high density of shrinking devices lead to newer faults and stimulated the introduction of new fault models, based on defect injection and SPICE simulation. Write Disturb Fault (WDF), Transition Coupling Fault (Cft), Deceptive Read Disturb Coupling Fault (Cfdrl) etc. are examples of some such newly defined fault models [14]. Another class of faults called Dynamic faults which require more than one operation to be performed sequentially in time in order to be sensitized. Traditional tests, like March C-, are thus 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 SS [15] and March RAW [16]. March SS 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.
An efficient and economical memory test should provide the best fault coverage in the shortest test time [13]. BIST is used to test memories in the paper and its precision is guaranteed by test algorithms. The algorithms in most common use are the March tests. March tests have the advantage of short test time but good fault coverage. In order to verify whether a given memory cell is good, it is necessary to conduct a sequence of write and read operations to the cell. The actual number of read/write operations and the order of the operations depend on the target fault model. Most commonly used memory test algorithms are March tests, in which there are finite sequences of March elements. A March element is a finite sequence of reads (r) or writes (w) operations applied to a cell in memory before processing the next cell. The address of the next cell can be in either ascending or descending address order. The notations are summarized in the Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>A read operation</td>
</tr>
<tr>
<td>w</td>
<td>A write operation</td>
</tr>
<tr>
<td>↑</td>
<td>Up addressing order</td>
</tr>
<tr>
<td>↓</td>
<td>Down addressing order</td>
</tr>
<tr>
<td>▼</td>
<td>Any addressing order</td>
</tr>
</tbody>
</table>

Example of some March-based tests are MATS, MATS+, March-C, March-Y, March-A and March-B. Since March-based tests are all simple and causes good fault coverage, there the dominant test algorithm implemented in most modern memory BIST. Comparison of different March algorithms and its fault coverage are summarized in Table II.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Descriptions</th>
<th>Fault coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>March c</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↓[r0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin, CFid, and CFst</td>
</tr>
<tr>
<td>March c-</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin, CFid, and CFst</td>
</tr>
<tr>
<td>March x</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin</td>
</tr>
<tr>
<td>March y</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin</td>
</tr>
<tr>
<td>March a</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin, CFid</td>
</tr>
<tr>
<td>March b</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin, CFid</td>
</tr>
<tr>
<td>MATS</td>
<td>↑[w0];↑[r0,w1];↑[r1];</td>
<td>Some AF, SAF</td>
</tr>
<tr>
<td>MATS+</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF</td>
</tr>
<tr>
<td>MATS++</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF</td>
</tr>
<tr>
<td>March SS</td>
<td>↑[w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0,w1];↑[r1,w0];↑[r0];</td>
<td>AF, SAF, TF, CFin, CFid, CFin, CFst, CFtr, CFrd, CFin, CFin, CFin, CFin</td>
</tr>
</tbody>
</table>

IV. PROPOSED BISR STRATEGY AND PMBIST

4.1. Redundancy Architecture

The proposed SRAM BISR strategy is flexible. The SRAM users can decide whether to use it by setting a signal. So the redundancy of the SRAM is designed to be selectable. In another word, some normal words in SRAM can be selected as redundancy if the SRAM needs to repair itself. We call these words Normal-Redundant words to distinguish them from the real normal ones. We take a 64 × 4 SRAM for example, as shown in Figure 1. There are 60 normal words and 4 Normal-Redundant words. When the BISR is used, the Normal-Redundant words are accessed as normal ones. Otherwise, the Normal-Redundant words can only be accessed when there are faults in normal words. In this
case, the SRAM can only offer capacity of 60 words to users. This should be referred in SRAM manual in details. This kind of selectable redundancy architecture can save area and increase efficiency. After BISR is applied, other modules in SRAM can remain unchanged. Thus the selectable redundancy won’t bring any problem to SRAM compiler.

**Figure 1. Architecture of redundancy in SRAM**

### 4.2. Overall BISR Architecture

The architecture of the proposed BISR strategy is shown in Figure 2. It consists of three parts: BIST module, BIAA module and MUX module. We call the SRAM with BISR a system. The BIST module uses March C- to test the addresses of the normal words in SRAM. It detects SRAM failures with a comparator that compares actual memory data with expected data. If there is a failure (compare\_Q = 1), the current address is considered as a faulty address. The BIAA module can store faulty addresses in a memory named Fault\_A\_Mem. There is a counter in BIAA that counts the number of faulty addresses. When BISR is used (bisr\_h = 1), the faulty addresses can be replaced with redundant addresses to repair the SRAM. The inputs of SRAM in different operation modes are controlled by the MUX module. In test mode (bist\_h = 1), the inputs of SRAM are generated in BISR while they are equal to system inputs in access mode (bist\_h = 0).
4.3. BISR procedure

Figure 3 shows the proposed BISR block diagram. The BISR starts by resetting the system (rst_1 = 0). After that if the system work in test mode, it goes into TEST phase. During this phase, the BIST module and BIAA module work in parallel. The BIST use March C- to test the normal addresses of SRAM. As long as any fault is detected by the BIST module, the faulty address will be sent to the BIAA module. Then the BIAA module checks whether the faulty address has been already stored in Fault-A-Mem. If the faulty address has not been stored, the BIAA stores it and the faulty address counter adds 1. Otherwise, the faulty address can be ignored. When the test is completed, there will be two conditions. If there is no fault or there are too many faults that overflow the redundancy capacity, BISR goes into COMPLETE phase. If there are faults in SRAM but without overflows, the system goes into REPAIR&TEST phase. The same as during TEST phase, the BIST module and BIAA module work at the same time in REPAIR&TEST phase. The BIAA module replaces the faulty addresses stored in Fault-A-Mem with redundant ones and the BIST module tests the SRAM again. There will be two results: repair fail or repair pass. By using the BISR, the users can pick out the SRAMs that can be repaired with redundancy or the ones with no fault.

4.4. Features of the BISR

Firstly, the BISR strategy is flexible. In access mode, SRAM users can decide whether the BISR is used base on their needs. If the BISR is needed, the Normal-Redundant words will be taken as redundancy to repair fault. If not, they can be accessed as normal words. Secondly, the BISR strategy is efficient. On one hand, the efficiency reflects on the selectable redundancy which is described as
flexible above. No matter the BISR is applied or not, the Normal-Redundant words are used in the SRAM. It saves area and has high utilization. On the other hand, each fault address can be stored only once into Fault-A-Mem. As said before, March C- has 6 steps. In another word, the addresses will be read 5 times in one test. Some faulty addresses can be detected in more than one step. Take Stuckat-0 fault for example, it can be detected in both 3rd and 5th steps. But the fault address shouldn’t be stored twice. So we propose an efficient method to solve the problem in BIAA module. Figure 4 shows the flows of storing fault addresses. BIST detects whether the current address is faulty. If it is, BIAA checks whether the Fault-A-Mem overflows. If not, the current fault address should be compared with those already stored in Fault-A-Mem. Only if the faulty address isn’t equal to any address in Fault-A-Mem, it can be stored. To simplify the comparison, write a redundant address into Fault-A-Mem as background. In this case, the fault address can be compared with all the data stored in Fault-A-Mem no matter how many fault addresses have been stored. At last, the BISR strategy is high-speed. As shown in Figure 4, once a fault address is stored in Fault-A-Mem, it points to a certain redundant address. The fault addresses and redundant ones form a one-to-one mapping. Using this method, the BISR can quickly get the corresponding redundant address to replace the faulty one.

![Figure 4. Flows of Storing Fault Addresses](image)

4.5 Programmable MBIST Architecture

The architecture for programming march test algorithms proposed in the fig.5. This architecture uses an instruction register specifying the current march test sequence by means of several fields indicating. 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. This flexibility, however, may come with the cost of higher logic overhead for the controller. Maximum number of march elements for all the algorithms mentioned above is six. Operations O1 to O6 specifies either read or write operation. One data polarity bit for each operation field (P1, P2, … P6), specifying if the corresponding operation uses the data word in its direct or complemented form. Programmable MBIST contains four important blocks. BIST controller coordinates the operations of different blocks of the BIST. A counter that can count in either direction, based on an up/down selector input, is known as an up/down counter. A Comparator that can compare the data which is written into the memory with the actual data which is read from memory.
V. EXPERIMENTAL RESULTS

The verilog code for BISR is simulated in modelsim simulator and synthesized in Xilinx 10.2, and the BISR output is checked in Spartan 3E FPGA kit. To verify the function of BISR, a Stuck-at-0 and Stuck-at-1 fault was set in the SRAM. In PMBIST march c- algorithm is implemented to test the memory. Similarly March SS algorithm also implemented. As it is Programmable BIST we can change any test algorithm to test the memory according to our requirement. After completion of test it is identified that 100th, 1000th locations stuck at 0 and stuck at 1 fault has occurred respectively. After repair the faulty address is mapped to redundant address and the fault location is repaired .and also the output is checked in Spartan 3E FPGA kit using chipscope pro IP Core Generator by adding ICON and VIO into the design.

Fig. 6 shows the Stuck- at-0 Fault at 100th memory location before repair. The expected data is all 1's but actual data is all 0's.Similarly Fig. 7 shows the Stuck- at-1 Fault at 1000th memory location before repair. The expected data is all 0's but actual data is all 1's. Fig. 8, Fig. 9 shows 100th, 1000th memory location after repair process based on March C-, March SS algorithms respectively. The data which is written and read to that location are same. So fault is repaired successfully. It is synthesized and implemented in SPARTAN 3E FPGA Kit. Table III shows design summary of BISR.
Figure 6. Stuck-at-0 Fault at 100

Figure 7. Stuck-at-1 Fault at 1000

Figure 8. Fault Locations after Repair (March c-Algorithm)
VI. CONCLUSIONS

An efficient BISR strategy for 4KX32 RAM with selectable redundancy has been presented and efficient fault coverage is obtained using PMBIST in this paper. The function of BISR has been verified by Modelsim simulation results. The BIAA module can avoid storing fault addresses more than once and can repair fault address quickly. The PMBIST architecture described here is an effective testing method to test embedded memories as it provides a flexible approach and better fault coverage. Just as March SS, any new march algorithm can be implemented using the same BIST hardware by changing the instructions without the need to redesign the entire circuitry. When compared with existing architectures in the literature, the proposed architecture is not only more flexible but also can be extended easily to incorporate complex data and addressing schemes that may be required to support complex algorithms in the future.

VII. FUTURE WORK

The proposed PMBIST architecture mainly focused on SRAM faults. Although these architectures are applicable to heterogeneous memories, the existing restriction is that test algorithms must be March-based. For some types of memories, such as large embedded DRAMs, March based test algorithms are not sufficient to detect certain physical defects. Therefore, a future work can improve the proposed architectures to support both March-based and non March-based test algorithms for embedded DRAM-specific memory faults, such as SNPSFs (Static Neighborhood Pattern Sensitive Faults). For
certain process technologies, circuit techniques or memory types, such as high-density DRAMs, testing NPSFs may be a requirement.

REFERENCES


AUTHORS

L. Dharma Teja obtained her B.Tech. degree from JNT University, Hyderabad in 2006, and M.Tech in VLSI System design from JNT University, Hyderabad in 2010. Pursuing PhD in the field of VLSI, in JNT University, Hyderabad, registered in 2011. She has started her career as Assistant professor and served for 2 years in DVR & Dr. HS MIC College of Technology, Kanchikacherla, Vijayawada from June 2006 to September 2008. Assistant Professor in the department of ECE in CMR Institute of Technology from May 2010 to May 2011. Assistant Professor in the department of ECE in VNR Vignana Jyothi Institute of Engineering Technology from June 2011 to till date. She is a life member of ISTE and IETE. Her areas of research interest are Fault Tolerance and Digital System Design in the field of VLSI design.
K. Kiruthika obtained her B.E Degree from ANNA University, Chennai in 2005. She has started her career as Embedded Engineer and served for 2 years in Indus Electronics (India) Pvt Ltd, Coimbatore from June 2005 to August 2007 and as a Hardware Engineer in Cornet Technology (India) Pvt Ltd, Chennai from September-2007 to November-2009. She is pursuing her M.Tech in VLSI System design from JNT University, Hyderabad in 2011-13. Her areas of research interest are Fault Tolerance and Digital System Design in the field of VLSI design.

V. Priyanka Brahmaiah obtained her B.Tech. Degree from JNT University, Hyderabad in 2007, and M.Tech in VLSI System design from JNT University, Hyderabad in 2010. She has started her career as Assistant professor and served for 2 years in MLR Institute of Technology & Management Dundigal, Hyderabad from June 2010 to June 2012. Assistant Professor in the department of ECE in Gokaraju Rangaraju Institute of Engineering & Technology from July 2012 to November 2012. Assistant Professor in the department of ECE in VNR Vignana Jyothi Institute of Engineering Technology from December 2012 to till date. She is a Life member of ISTE and IETE. She presented one research paper in International Journal. Her areas of research interest are Bio-Medical, Signal and Image Processing, Human Computer Interface.