CHAPTER 3

PREVIOUS WORK ON MEMORY BUILT IN SELF TEST & SELF REPAIR METHODS AND MOTIVATION

The chapter 2 introduced the basic fault models, test algorithms for the semiconductor memories and BISR design for the yield improvement of embedded memories. The question is, if these memories are embedded into an SOC, then how are the test vectors applied and how are the test responses observed. As discussed earlier, there are two main approaches for testing embedded memories (i) external test by direct access using ATE and (ii) Internal test by using BIST. On one hand, direct access to the embedded memory core from the limited number of Input-Output pins needs a high performance ATE, along with a very long testing time. Thus external test becomes infeasible, in particular for the large SOC devices, where the transistor to pin ratio is very high. On the other hand, BIST provides an at-speed and high access to the embedded memory core; it only needs a low cost ATE to initialize the test session and to inspect the final result. Though BIST is state of the art technology for embedded memory testing, it may induce excessive power, in addition to effect on performance and overhead.

Since embedded memories account for more than 60% of the silicon area in today's SOC, relevant approach to test and repair the embedded memories is needed. To improve the yield and reliability of SOC chips, relevant approaches to test the embedded memories and different BISR
approaches to swap out the faulty memory location with the redundant memory through an optimum reconfiguration methods presented in the literature are summarized along with their strengths and limitations. Later the motivation for the present work is presented. A generic BIST & BISR architecture for memory is presented in figure 3.1

**Figure 3.1 Generic BIST & BISR Architecture for Memory**

### 3.1 MBIST Challenges

A typical embedded MBIST approach comprises of an MBIST wrapper and an MBIST controller. The MBIST wrapper includes an address generator, a back ground pattern generator to produce data for testing the memory, a comparator to check the memory output against an expected correct data pattern. The MBIST controller is a finite state machine to generate proper test control signals based on the test
power dissipation will be identical in both the test mode and normal mode. Therefore, if all the memory blocks in a SOC can be activated simultaneously during the functional mode, the power dissipation will not exceed the maximum value. Hence, no test scheduling is required.

4. Design Reuse

Reusing IP cores in an SOC can greatly simplify the design phase and cut down the time to market. A reusable MBIST core with a scalable and portable architecture, associated with a clear methodology for the design flow integration can significantly reduce the cost of test preparation.

5. Built in self Reparability

To improve the yield and reliability of VLSI chips, BIST and error correcting codes alone are not sufficient. Further, BIST and BISR along with the BISD are essential to improve the testability and reparability of RAMs used as embedded memories, or in mission-critical applications where external field testing and repair are prohibitively expensive or infeasible.

The objective of Memory BIST and BISR approach is to meet some or all the above requirements while reducing the cost to test and repair by targeting low area overhead and performance penalty. The existing approaches presented in this chapter have explored in few directions to gain improvements in BIST and BISR architecture like test scheduling, reconfiguration analysis and design implementation.
3.2 Previously proposed Memory BIST and BISR architectures

MBIST architecture is defined as the integration of Controller, Wrapper and interconnects. Several Memory BIST solutions have been proposed to test the memories in [1-4, 9-17]. A successful BIST for embedded memory has to guarantee core accessibility, scalability, in-system programmability, low area overhead and flexibility in the test scheduling [13]. Most of the Memory BIST schemes exploit the parallelism within the memory device to achieve a massive reduction in test time. Due to the regularity and higher fault coverage, the March tests are considered over pseudorandom memory tests. In [3] the limitations of conventional ATPGs enabled the author to develop a BIST. A stimulus circuit to store the test patterns and response circuit to analyze the output are added to the circuit under test. The fault coverage achieved by this BIST depends on the functionality of the circuit under test. Application of test sequences to embedded memories using off chip tests results in a high test time and test cost due to large size of the embedded memories. To overcome this problem, the computed test sequences are generated on-chip using a memory BIST unit in [2].

As BIST introduces extra logic with a performance penalty and area overhead, in [4] a novel approach for testing the embedded memories in complex SOCs is presented. This design uses the existing on chip resources and dedicated hardware for testing. This does not exceed the
functional power constraints with a trade off of testing time against area overhead and performance penalty. In [11] a 9N length BIST routine is written into the existing ROM. However, in this paper only spot defects are considered, where these defects result in breaks or shorts in the circuit. As safety and reliability are the system level requirements, the responsibility of a BIST is to identify the faults both in the normal mode and in test mode of memory under test. The approaches mentioned above lack the on-line testing (concurrent or non-concurrent).

In [12,15] an on-line and off-line BIST is designed to test the embedded memory. In [12] a unified test design methodology for both on line and off line testing is proposed. The purpose of this is to develop homogeneous test algorithm and test strategy on different abstract levels of computer system from gate-level through RTL to system-level. In [15] the architecture combines the fault latency reduction, code based fault detection, an architecture based fault avoidance to meet the reliability constraints. In [9] an approach with diagnostic testing of embedded memories apart from self testing is presented, which provides complete detection and distinguishing of all faults from a given set of fault models. This approach is based on decomposition of functional memory faults into basic fault effects and output tracing. This approach examines memory tests of linear order, as they require low test application time and realizable BIST circuit with low area overhead.
3.2.1 MBIST controller

Whatever be the test strategy, a Memory BIST consists of a controller to control the flow of test sequencing and other components to generate necessary test control and data. This Memory BIST controller could be designed as FSM based controllers, which is the hardware realization of a selected memory test algorithm. This type of memory BIST has optimum logic overhead, but lacks the flexibility to accommodate any change in the selected memory test algorithm [2]. The FSM for one March test is fixed and any increase in fault model or fault type cannot be detected by the designed March test controller, with that the FSM becomes useless. This result in redesign and re-implementation of the FSM based controller for any minor change in the selected memory test algorithm. This problem can be overcome by a microcode based controller, where a selected memory test algorithm is written in terms of set of supported instructions and loaded into the memory BIST controller. This type of architecture allows changes in the selected test algorithm with no impact on the hardware of the controller. With little complexity in hardware, this BIST can be converted into a general purpose programmable BIST controller. However, this flexibility results in higher logic overhead. In [40] a programmable BIST core for embedded DRAMs is presented. This BIST supports various test modes, thus enabling ATPG to have wide range of March tests to test various memories. It also supports BISD by
feeding error information to the external tests. Moreover, by using a specific test sequence, it can test for critical timing faults, reducing the tests time for AC parametric tests. As this design begins at the RTL level, test element insertion and deletion are relatively easy. An IEEE P1500 based memory wrapper along with a programmable BIST approach is explained in [10]. It is a programmable BIST for diagnosis of embedded cores. In summary with the exception of P1500 memory BIST approaches, most of the stand alone memory BIST architectures focus only on solving the test problems related to a single memory core. They do not account for the specific requirement of integrating the DFT hardware for multiple embedded memory cores. In reference [14] a BIST module implementing a March algorithm is explained in which a wrapper is designed and interfaced the memory to Test Access Port (TAP) controller. Hence, this allows the BIST to test multiple modules in the same chip through single TAP interface. As pure BIST based diagnosis is area expensive and low flexible, a hybrid solution is adopted. In this case, the available ATE software is used to reduce the time to detect and locate the fault. The ATE gathers some information about the embedded memory during the execution of March tests.

If the aforementioned methods target to test a single memory module in an SOC; Reference [13] presents a programmable BIST architecture based on a single micro programmable BIST processor and a set of memory wrappers to test a large number of distributed multi-port
memories of different sizes, access protocols and time. It has provided core accessibility, scalability and in system programmability with flexible test scheduling. The trade off of this method for the above benefit is each port of distributed memory requires a port wrapper and a dispatcher circuit. It also deals with memory modules array. Optimizing the hardware by reusing the wrapper is not allowed, also to collect the diagnostic information, the test must be re-executed on the faulty memory. Yet another approach in [15] combines four strategies to meet strict fault detection, fault tolerance, area over head and system performance constraints. However, BISR capability for the on line BIST is not provided.

In [16] a novel approach to accelerate the fault classification using Image Processing Techniques has been applied. Here an ATE is used to drive the diagnostic scheme for the classification of faults. This approach allows a test time reduction, by limiting the number of BIST runs. The limitation on the efficiency of the method depends on the memory size. It was also observed that the overall efficiency of the algorithm is dropped in the presence of multiple failure types occurring simultaneously in the memory.

From the study of all the above BIST solutions there is a need to design a BIST with programmability, which allows different test algorithm to run and test the embedded memories in an SOC. The programmable memory BIST runs offline to test the embedded memory
for permanent faults like stuck at faults. To meet the system requirements like safety and reliability, fault detection alone is not sufficient; the proposed Memory BIST should have a capacity to diagnosis the faults by setting a bit or a flag indicating the faulty location in the memory. To improve the yield and the fault tolerance of the embedded memory it is proposed to bypass the faulty locations and reconfigure the faulty memory address to one of the available spare memory locations. Thus, it is also proposed to design a BISR for the embedded memory. Hence, in the present thesis the first half of the thesis i.e., chapters 4 & 5 presents the modeling of BIST & BISD through two different approaches (Programmable and FSM based BIST/BISD approaches). The fault information generated by the BISD is considered as the input to the BISR unit.

Prior to the design of BISR there is a need to analyze different approaches proposed in the literature to swap in the spare memory with the faulty memory location in an optimized way to use the spare elements.

It is a known fact that a small performance degradation or defect in core components can result in unacceptably low manufacturing yield and reliability of SOC. Unlike legacy printed circuit board or multi chip modules based systems; embedded core cannot physically be replaced once they are fabricated into a SOC [21]. To realize both enhanced yield and reliability, BIST and BISR are utilized to test and to allocate
redundancy for the embedded cores. Proper utilization of spare is significantly desirable to achieve yield and reliability.

The memory repair problem dates back to the evolution of 256K bit DRAMs in the early 80s; where two to three spare rows or columns were added to each quadrant of 64K bit sub array to improve the chip yield. Simple, Greedy and Exhaustive search algorithms [1] were developed to repair the memory array. Since the search space was very small for such problem sizes, all these algorithms provided high through put in memory diagnosis and repair. But with the memory size increasing to several mega bits for the last few decades, the defect patterns became sufficiently complex and the search space grew extensively [22]. With this, the problem of memory repair has become NP complete. A number of heuristic algorithms such as Branch and Bound, Approximation, Best-first search etc., have been proposed to solve this problem. The key limitation of these algorithms is that their worst case complexity is nearly exponential.

Off late a word oriented BISR methodology is described in [23] with the usage of fuse boxer. Whereas in [24] a BISR machine is proposed, which can test at-speed and repair embedded SRAMs. Unlike the common approach to blow laser fuses, the repair is completely accomplished by BISR by storing the repair operation in an on-chip FLASH memory in [23], the BIST and redundancy logic are placed in parallel to the memory unit without any additional delay, and the
redundancy will be activated within one clock cycle after the fault
detection. Also this BISR uses spare words instead of spare rows or
columns.

Apart from word repair technique, several papers discussed the self
repair techniques by using column replacement [18], column or data bit
replacement [19] or both column or row replacement [27]. In [18, 28] self
repair technique for embedded memories is used which reuses the
surrounding logic like embedded processor normally found in any SOC to
detect the fault and to analyze the redundancy technique. The limitation
with this type of conventional processor based BIST is timing efficiency
as the limited processor instruction set and its machine cycle, as the test
signals are generated by the embedded processor in SOC, the area
overhead may be minimized but the test and the repair time is not in
control. Row/Word repair is the simplest BISR approach. Hence majority
of the previous works considered this scheme. However, work on
column/ data bit BISR is more recent due to the difficulty in elaborating
the reconfiguration functions for this repair. To minimize this complexity,
a single fault per test sessions may be considered. That is, the memory is
tested till a first fault is found and repaired. Then, the memory is tested
again until the second fault is found and repaired, and so on. This
process may simplify the job of BISR, but the test and repair time will be
very high as the number of faults increase. In [19, 29] a dynamic repair
scheme that increases the repair efficiency of data-bit repair is
presented. This is done by using a single spare unit for repairing the fault affected in several regular units. But the drawback of this scheme is the size of each spare unit which is equal to the ratio of number of memory cells to the number of words. It results in a significant area overhead for memory with small to medium word width.

Finally, whatever be the redundancy logic used in BISR, the process of repair of RAMs can be divided into three steps. In the first step a test algorithm is executed by the BIST on the memory under test. If a fault is detected, it is necessary to locate it by using the BISD. Finally the BISR unit has to allocate redundant memory spare to replace the faulty cells [6]. This allocation has to be done with the help of an efficient RA.

A simple repair algorithm which does the fail address analysis is presented in [27], where the faulty addresses are stored in a Content Addressable Memory (CAM) and have a count of number of faults in rows or columns, and if the number of Row fail count is less than the available spare rows or number of column fail count is less than available spare columns, the self repair unit repairs the faulty row/column with the redundancy. The fail count is decremented by one after the repair. There are many other heuristic algorithms for the spare allocation. Out of many approaches the three simple algorithms are row first algorithm, column first algorithm and small bit map algorithm. The Row first and column first algorithms are easy to implement, whereas the small bit map algorithm needs to store some fault location information and
perform redundancy allocation. A hybrid solution to these approaches is presented in [7, 28, and 32]. In [7] each redundant row is partitioned into redundant words and repair with a little complex address remapping circuit. However, this approach increases the utilization of the spare elements. Whereas in [28 and 32] a new technique called block repair architecture is presented instead of dividing the row into words, it is virtually divided into blocks. The reconfiguration is performed at the block level. As the memory structure is not modified, the hardware overhead for the virtual row-block divide-circuit is negligible. It was observed that it has higher repair rate than the other 1-Dimentional redundancy techniques. The limitation of this approach is, if there are many faults detected in a memory array, one can use more block select lines to enhance the repair rate, but more the blocks in a memory row, higher is the hardware and word line delay.

In [33] a near similar approach to [27, 5] is presented where a Built In Redundancy Analyzer (BIRA) is used, which executes the proposed RA algorithm with 2-D redundancy structure. In this paper the spare column is partitioned into several groups and each spare group is further logically partitioned into segments for better utilization as shown in figure 3.2. Here the faulty cell is replaced by the spare column of that segment to which the faulty cell belongs. However, the area overhead is relatively high when compared to other 1-D redundancy techniques. In [5], a word oriented BISR technique is presented to repair SRAM. Here
the repairing is done during the reset period and the process of repair or referring the faulty lines is not required during the normal operation. Hence, the access time penalty and additional power consumption is negligible. There is flexibility in this design as it uses soft fuse circuit as a redirection device. In this design, each row is connected with K number of soft switches. As far as no defective row is found, the row decoder address and topological address are identical for a row.

Figure 3.2 Spare Memory Segmenting

If the first defect is detected say at a word line WL_{n+1}, the soft fuse switches the switch S1, by which the damaged row is skipped and redirects the address to the next word that is WL_{n+2}, thereafter all the
remaining word lines i.e., WL_{n+2}, WL_{n+3}... should switch to S1, until the next defective row is detected. With each defective row detected, the WL topological address switch by 1 until all the switches are over. The limitation of this method is only non-consecutive damages up to K-1 can be handled. Also though there is no defect at word line WL_{n+2}, still the switching to next location is a must; which is a time taking and unnecessary process.

In [34] a Comprehensive Real time Exhaustive Search Test and Analysis (CRESTA) procedure to repair a memory is presented. It is a bit oriented fault analyzer. In this procedure final repair strategy is based on the result of all the sub analyzers. If there are m spare rows and n spare columns in a memory, the CRESTA contains m\times{}nCm sub analyzers. Each sub-analyzer analyzes the incoming Row/column address of faulty memory cells in parallel in a different repair strategy using spare row/column. For example, if there are 2 spare rows and 2 spare columns then 4C_2=6 sub analyzers are needed. The six repair strategies are RRCC, RCRC, RCCR, CRRC, CRCR, and CCRR. In this order each sub analyzer allocates the spare row/column to the faulty cells that are appearing in the order. Reference [17] presents an exhaustive search procedure for word oriented memory repair. The parallelism in this proposed CRESTA algorithm drastically improves the speed of the analysis. However, the hardware overhead will be very high as the number of spare elements increase.
In [20] a simulator for evaluating the RA algorithms is presented. This simulator can calculate the repair rate of the given RA algorithms with the associated memory configuration and redundancy structures. In [22] a procedure to solve the memory array repair by using neural network is demonstrated. With respect to the current sequential repair algorithms, the neural nets connectivity and computational property provides an attractive solution.

From the study of previous approaches to design a BISR unit for an embedded memory in a SOC, it is observed that there is a need to design a Built In Self Repair Analyzer. This BISRA has to analyze the given failure information which is generated by the BIST & BISD circuits and it has to come out with an efficient Redundancy analysis algorithm, which uses the available spare elements optimally. As the fault tolerance and reliability of the system are the goals of a BISR, the analyzer has to decide upon the repairing faulty memory with the utilization of the available spare memory without stopping the normal operation of the system. It is also aimed to maintain low area overhead, power consumption and negligible time delay to test and repair. Though CRESTA provides a solution for an optimum utilization of spare elements, the limitation is area overhead and designed for bit oriented memory repairing. It is not suitable for word oriented memories.
Two different BISR schemes for the embedded SRAMs in a SOC are presented. If the BISR procedure is divided into three steps, the first step is BIST. The second is BISD and the third step is the BISR. Two different hardware controller BIST architectures are presented in chapters 4 and 5. The BISD is included in the BiST architecture presented in chapter 5. The two BISR designs are presented in chapters 6 and 7. The first approach is aimed at providing fault tolerance through 1-D redundancy and the second approach is an exhaustive search approach to allocate a spare element in the place of faulty memory element, such that the repair rate increases with the same number of spare elements than by a conventional redundancy analyzer. It is designed for word oriented memories through 2-D redundancy. These approaches also aimed at achieving 100% fault repair and minimal over head penalties due to the BISR logic.