Embedded Static RAM Redundancy Approach using Memory Built-In-Self-Repair by MBIST Algorithm

Rakesh Manukonda **Suresh Nakkala**
LEC, Singarayakonda, MLEC, Singarayakonda,

Abstract
This paper proposes Memory Built in Self Repair (MBISR) approach which consists of a Built-In Self-Test (BIST) module, a Built-In Address-Analysis (BIAA) module and a Multiplexer (MUX) module. On average embedded RAMs occupy 90% area in system-on-chip (SOC), so embedded memory test design has become an essential part of the SOC development. Built-In Self-Repair (BISR) with Redundancy is an effective yield-enhancement blueprint for embedded memories. The entire design consists of a BIST (Built in self-test) which uses MARCH C- algorithm for test pattern generation (TPG), an SRAM of 6 bit address and 4 bit data that operates in 4 modes as circuit under test (CUT), a Built in Address analyzer (BIAA) for storing faulty locations of SRAM and to repair those faulty locations and a MUX module to select test data and external data. The combination of BIST, MUX and BIAA together is called as BISR (Built in Self repair) System in the design. A practical 4K × 32 SRAM IP with BISR circuitry is designed and implemented based on a 55nm CMOS process. Experimental results gives that the BISR occupies 20% area on SoC and it works up to 150MHz range. This paper is implemented using Verilog HDL. Simulation and Synthesis is done using Xilinx ISE 14.2 Tools.

Keywords
Built-In-Self-Repair (BISR); Built-In Address-Analysis; Built-In-Self-Test; Embedded SRAM; Multiplexer MUX;

1. Introduction

The VLSI manufacturing technology advances has made possible to put millions of transistors on a single die. This advancement in IC technology enabled the integration of all the components of a system into a single chip. A complex IC that integrates the major functional elements of a complete end product into a single chip is known as System on Chip (SOC). It enables the designers to move everything from board to chip. SOC incorporates a portable / reusable IP, Embedded CPU, Embedded Memory, Real World Interfaces, Software, Mixed-signal Blocks and Programmable Hardware. Reduction in size, lower power consumption, higher performance, higher reliability, reuse capability and lower cost are the benefits of using SOC. However, before SOC products can be widely seen on the market, many design and manufacturing issues have to be solved first. One of them is testing plural, heterogeneous cores of the SOC and the chip itself.

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.

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. This paper proposes an efficient BISR strategy which can store each fault address only once.

2. 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]:

- **Stuck-at-Fault (SF):** Either a cell or a line is stuck to logical `0' or `1'.

Manuscript received February 5, 2015
Manuscript revised February 20, 2015

- **Transition Fault (TF):** The 0->1 (or 1->0) transition is impossible on a cell or a line.
- **Coupling Fault (CF):** When a cell is written from 0->1 (or 1->0), the content of the other cell is changed. CF is generalized to a k-coupling fault when k-1 cells are changed and is classified into Inversion or Idempotent coupling faults depending upon what content changed.
- **Address Decoder Fault (ADF):** No cell will be accessed with a certain address or multiple cells are accessed simultaneously or a certain cell can be accessed with multiple addresses.
  a) There is no cell for particular address
  b) There is no address for a cell
  c) Multiple cells pointed by single memory address
  d) Multiple Addresses points to single cell

- **Address Decoder Open Faults (ADOF):** CMOS address decoder open faults are caused by open defects in the CMOS logic gates of the memory address decoders, and due to their sequential behavior, cannot be mapped to faults of the memory array itself.
- **Retention Faults (RF):** A cell fails to retain its logic value after some time. This fault is caused by a broken pull-up resistor.
- **Neighborhood Pattern Sensitive Fault (NPSF):** a typical neighborhood pattern sensitive faults preventing the base cell from being transited to a certain value is called `static' NPSF, and an NPSF is called `dynamic' when a transition on the base cell.

MBIST ALGORITHMS:

A. Classical Test Algorithms

Classical test algorithms are either simple, fast but have poor fault coverage, such as Zero-one, Checkerboard; or have good fault coverage but complex and slow, such as Walking, GALPAT, Sliding Diagonal, Butterfly, MOVI, and etc.. Due to these imbalanced conflicting traits, the popularity of these algorithms is decreasing.

B. March-based Test Algorithms

They are the foundations of the memory test. 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. There are many March tests such as March C, March C-, March C+, March 3 and so on. TABLE I compares the test length, complexity and fault coverage of them 'n' stands for the capacity of SRAM.

March tests have the advantage of short testing time with good fault coverage. The proposed MARCH algorithm is MARCH C-. The MARCH C- algorithm has the high fault coverage with less test length. The algorithm steps are as follows.

1 $\uparrow$-w0

2 $\uparrow$-r0, w1

3 $\uparrow$-r1, w0

4 $\downarrow$–r0, w1

5 $\downarrow$-r1, w 0

6 $\downarrow$–r0

March C- has better fault coverage than March 3 and shorter test time than March C and March C+. So March C- has been chosen as BIST algorithm in this paper .In above steps, "up" represents executing SRAM addresses in ascending order while "down" in descending order.

Table 1. Comparisons of March Tests

Algorithm Name	Test length	Fault Coverage
March C	11n	AF,SAF,TF,CFin,CFid and CFst
March C-	10n	AF,SAF,TF,CFin,CFid and CFst
March C+	14n	AF,SAF,TF,CFin,and CFid
MATS	4n	AF,SAF
MATS+	5n	AF,SAF
Marching 1/0	14n	AF,SAF,TF
MATS++	6n	AFs, SAFs, TFs, Some CFs
March A	15n	AFs, SAFs, TFs, Some CFs
March Y	8n	AFs, SAFs, TFs, Some CFs
March B	17n	AFs, SAFs, TFs, Some CFs

The BIST module in the paper refers to the MBISR design of Mentor Graphics. It mainly consists of a BIST controller, a test vector generator, an address generator and a comparator. It can indicate when memory test is done and weather there is fault in memory.

3. Proposed MBISR Scheme

A. Basic Architecture

The proposed MBISR Scheme is flexible. 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. Repair is a popular technique for memory yield improvement. The MBISR flow is shown Figure 1.

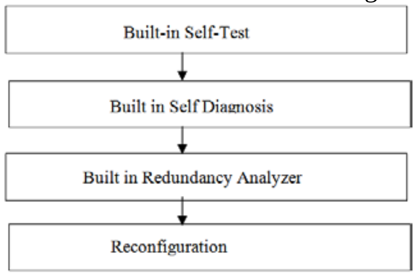

Figure 1. MBISR Flow diagram

In MBISR built in self-test is used to generate the test vectors for the require memory circuit. Built in self-diagnosis is used to detect the faults. After the fault diagnosis the redundancy memory (Spare rows and columns are allocated to the faulty memory locations using redundancy analyzer. In the reconfiguration the defective cells are swapped with the spare cells

A. Proposed BISR Architecture

The proposed architecture mainly consists of three modules: BIST module, BIAA module, MUX module. 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. 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 (test_h = 1), the inputs of SRAM are generated in BISR while they are equal to system inputs in access mode (test_h = 0). The BISR has three modules namely BIST, BIAA and the MUX. Signals test_CEN,test_WEN,test_D, test_A are the test inputs given to SRAM through MUX from BIST when test_h=1. CEN_0, WEN_0, D-0, A_0 are the external inputs given to SRAM by the user when test_h=0. Signal test_done will be high when march test is completed. Over_h is high when number of faults are more than Fault_A_Mem array size. Fail_h is high when there is a fault in the redundancy.

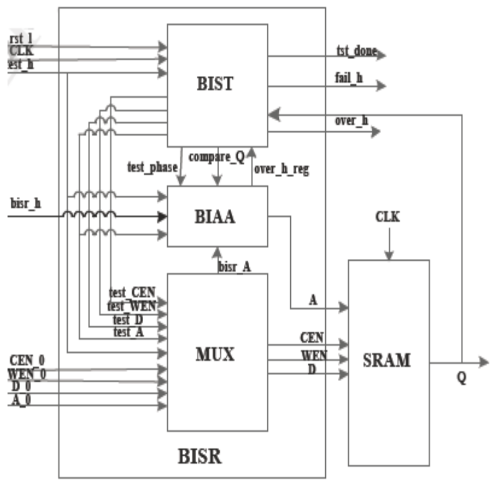

Figure 2. Proposed BISR Architecture

B. BIAA (Built in Address Analyzer) Module

BIST 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).

C. BISR Flow:

Each fault address can be stored only once into Fault-A-Mem. 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 Stuck-at-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. Below figure 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

The fault address is stored in fault-mem-A and it points to a certain redundant address. The redundant address maps one-to-one to the fault-mem-A. The fault-mem-A stores the faulty address in it and the normal data is checked with the fault-mem-A and gets stored in the RAM here we are considering a 64 bits address RAM and we consider 6 redundant Bits. The faulty address are stored in the fault-mem-A and when these fault address are mapped with the test address and when the fault address is found in the fault-mem-A then that faulty address is replaced by the redundant bits. Using this method the BISR can quickly get the corresponding redundant address to replace the faulty one. As shown in Figure 3. 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.

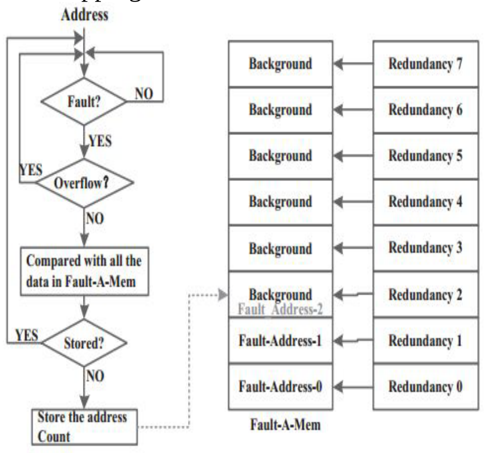

Figure 3. Flow chart showing address storage

D. BISR Features:

TABLE II. SRAM OPERATION MODES

Modes	*Repair selection*	*Operation*
Test mode (*test_h*=1)	Default: repair (*bisr_h*=1)	Access normal words. Repair faults and test.
	Don't repair (*bisr_h*=0)	Access normal words. Test only.
Access mode (*test_h*=0)	Repair (*bisr_h*=1)	Access normal words. Repair faults and write/read SRAM.
	Don't repair (*bisr_h*=0)	Access Normal- Redundant and normal Words. Write/read SRAM only.

Firstly, the BISR strategy is flexible. TABLE II lists the operation modes of SRAM. 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 Stuck at-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.

4. Experimental Results

The proposed BISR was designed at RT level and it was synthesized to gate-level using Synopsys DC compiler.

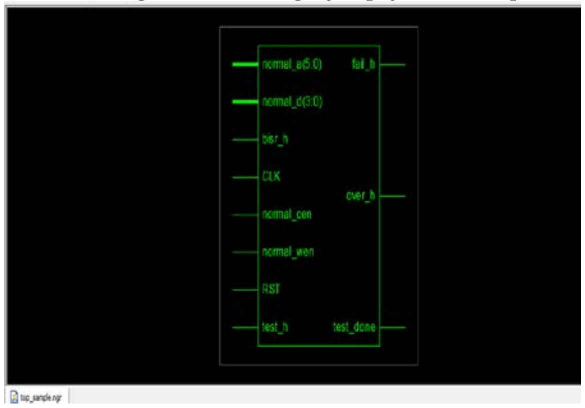

Figure 4. RTL schematic of top module

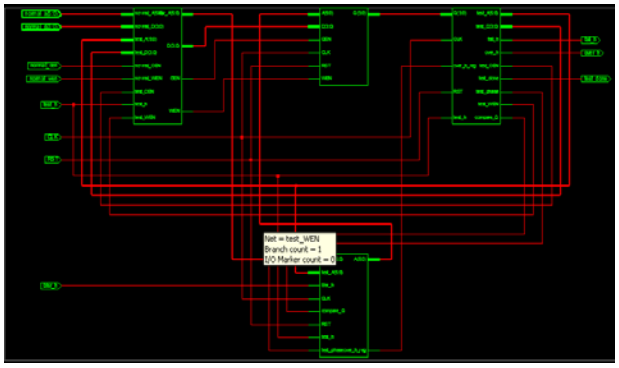
Figure 5. RTL schematic of top module with internal modules

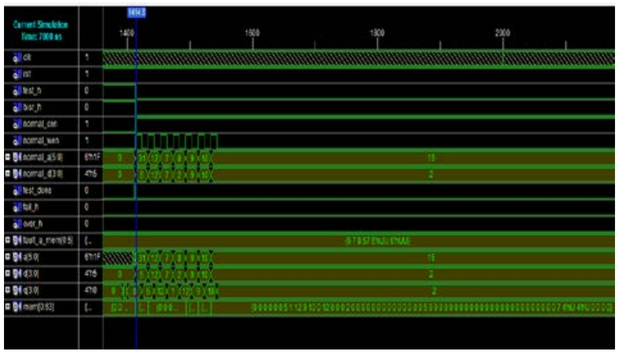
Figure 6. Simulation of top module 1

normal_a[5:0] and normal_d[3:0]are the external address and data given to the SRAM. SRAM has 4 faulty locations 6,7,8,57. q[3:0] is the output of SRAM.mem[0:63] is the SRAM of 64 locations. When test_h=1 and bisr_h=0 BIST tests 58 locations and BIAA stores the faulty addresses in fault_a_mem[0:5]. If fault locations are more than 6 BISR system indicates overflow by making over_h=1.When test_h=1 and bisr_h=1 BIST tests the redundant locations. In Figure 5.1a 4 redundant locations are tested since SRAM has 4 faulty locations.

HDL Synthesis Report

```
Macro Statistics
# Adders/Subtractors             : 3
 6-bit adder                     : 2
 6-bit subtractor                : 1
# Counters                       : 1
 3-bit up counter                : 1
# Registers                      : 75
 1-bit register                  : 5
 4-bit register                  : 61
 6-bit register                  : 9
# Latches                        : 1
 4-bit latch                     : 1
# Comparators                    : 19
 3-bit comparator greatequal     : 1
 3-bit comparator greater        : 1
 3-bit comparator less           : 1
 6-bit comparator equal          : 12
 6-bit comparator greater        : 1
 6-bit comparator less           : 1
 7-bit comparator greatequal     : 1
 7-bit comparator lessequal      : 1
# Multiplexers                   : 1
 4-bit 64-to-1 multiplexer       : 1
# Tristates                      : 1
 6-bit tristate buffer           : 1
```

Final Results

```
RTL Top Level Output File Name:
top_sample.ngr
Top Level Output File Name : top
sample
Output Format                : NGC
Optimization Goal            : Speed
Keep Hierarchy               : NO
Design Statistics
# IOs                        : 19
Cell Usage:
# BELS                       : 540
#      INV                   : 2
#      LUT2                  : 20
#      LUT3                  : 155
#      LUT3_D                : 4
#      LUT3_L                : 4
#      LUT4                  : 198
#      LUT4_D                : 13
#      LUT4_L                : 15
#      MUXF5                 : 84
#      MUXF6                 : 28
#      MUXF7                 : 12
#      MUXF8                 : 4
#      VCC                   : 1
# FlipFlops/Latches          : 319
#      FDCE                  : 4
#      FDE                   : 311
#      LDE                   : 4
# Clock Buffers              : 1
#      BUFGP                 : 1
# IO Buffers                 : 18
#      IBUF                  : 15
#      OBUF                  : 3
```

5. Conclusion

An efficient BISR strategy for SRAM with selectable redundancy has been presented in this paper. It is designed flexible that users can select operation modes of SRAM.

The BIAA module can avoid storing fault addresses more than once and can repair fault address quickly. We are performing Built in self-test and repair to detect the faults in offline mode. The future scope of this study can be to make use of any other algorithm which can detect more number of faults. We can also try to detect the faults in the online mode.

References

[1] Semiconductor Industry Association, "International technology roadmap for semiconductors (ITRS), 2003 edition," Hsinchu, Taiwan, Dec.2003.

[2] C. Stapper, A. Mclaren, and M. Dreckman, "Yield model forProductivity Optimization of VLSI Memory Chips with redundancy and Partially good Product," IBM Journal of Research and Development,Vol. 24, No. 3, pp. 398-409, May 1980.

[3] W. K. Huang, Y. H. shen, and F. lombrardi, "New approaches for repairs of memories with redundancy by row/column deletion for yield enhancement," IEEE Transactions on Computer-Aided Design, vol. 9, No. 3, pp. 323-328, Mar. 1990.

[4] P. Mazumder and Y. S. Jih, "A new built-in self-repair approach to VLSI memory yield enhancement by using neuraltype circuits," IEEE transactions on Computer Aided Design, vol. 12, No. 1, Jan, 1993.

[5] H. C. Kim, D. S. Yi, J. Y. Park, and C. H. Cho, "A BISR (built-in selfrepair) circuit for embedded memory with multiple redundancies," VLSI and CAD 6th International Conference, pp. 602-605, Oct. 1999.

[6] Shyue-Kung Lu, Chun-Lin Yang, and Han-Wen Lin, "Efficient BISR Techniques for Word-Oriented Embedded Memories with Hierarchical Redundancy," IEEE ICIS-COMSAR, pp. 355-360, 2006.

[7] C. Stroud, A Designers Guide to Built-In Self-Test, Kluwer Academic Publishers, 2002.

[8] Karunaratne. M and Oomann. B, "Yield gain with memory BISR-a case study," IEEE MWSCAS, pp. 699-702, 2009.

[9] I. Kang, W. Jeong, and S. Kang " High-efficiency memory BISR with two serial RA stages using spare memories," IET Electron. Lett., vol. 44, no. 8, pp. 515-517, Apr. 2008.

[10] Heon-cheol Kim, Dong-soon Yi, Jin-young Park, and Chang-hyun Cho, "A BISR (Built-In Self-Repair) circuit for embedded memory with multiple redundancies," in Proc. Int. Conf. VLSI CAD, Oct. 1999, pp.602-605.

[11] M. Sachdev, V. Zieren, and P. Janssen," Defect detection with transient current testing and its potential for deep submicron CMOS ICs," IEEE International Test Conference, pp. 204-213, Oct. 1998.