# AREA, DELAY AND POWER ANALYSIS OF BUILT IN SELF REPAIR USING 2-D REDUNDANCY

Aman Kumar Sabnani[1] and Balwinder Singh[2]

[1]Research Scholar, C-DAC, Mohali (P.B.), India
[2]Coordinator, ACSD, C-DAC, Mohali (P.B.), India

## ABSTRACT

*System on Chip comprises of programmable processor, different controller and memory. As chip size is decreasing memory density is increasing. These high density memories are susceptible to faults. To increase the yield and make device reliable, testing and self-repair are the important issues. To repair embedded memories in SOC, Built in self-repair techniques are used by firstly detecting, then locating and in the end repairing the memory. In this paper six BISR are designed using six different March algorithms as MBIST then compared altogether. Since power dissipation during testing operation is around twice the power dissipation during normal operational mode, thus low power BISR design is necessary considering the power constraints these days. Due to high switching activity chip can get overheated resulting in malfunction and damage. Power consumption is reduced by reducing the switching activity in the address line when writing and reading the memory during test.*

## KEYWORDS

*Memory Built In Self-Test (MBIST), Built In Self-Repair (BISR), Low Power Testing, March Algorithm, Embedded Memories.*

## 1. INTRODUCTION

System-on-Chip being the demand of today‟s world brings hardware and software components together. SOC provides complete solution by incorporating programmable processor, memory, different controllers like audio video and graphics on a single chip. If a single component becomes faulty whole chip needs to be replaced thus affecting the yield. Memory occupies more than 90% of the total chip area. Such high density memories are susceptible to faults which reduces the yield of SOC. This leads to the need of self-repair. Built In Self Repair (BISR) uses Memory Built In Self Test (MBIST) to detect and locate faults in the memory. After locating the faults in the memory the Built In Redundancy Allocation (BIRA) allocates the redundant memory against the faulty memory. Also, according to recent researches it is revealed that the power dissipation is high during testing due to high number of bit transitions. Since in CMOS heat dissipation is proportional to switching activity thus chip can get damaged due to overheating resulting in low yield of SOC. Thus reduction in power consumption during testing and repairing is an important design challenge.

DOI : 10.5121/vlsic.2015.6503 

## 2. MEMORY BUILT IN SELF TEST

Memory Built In Self Test(MBIST) are used to test memories. In MBIST March Algorithms are used to provide test patterns in a specific sequence of read and write operations which are discussed below. Architecture of MBIST is shown in figure 1 below:

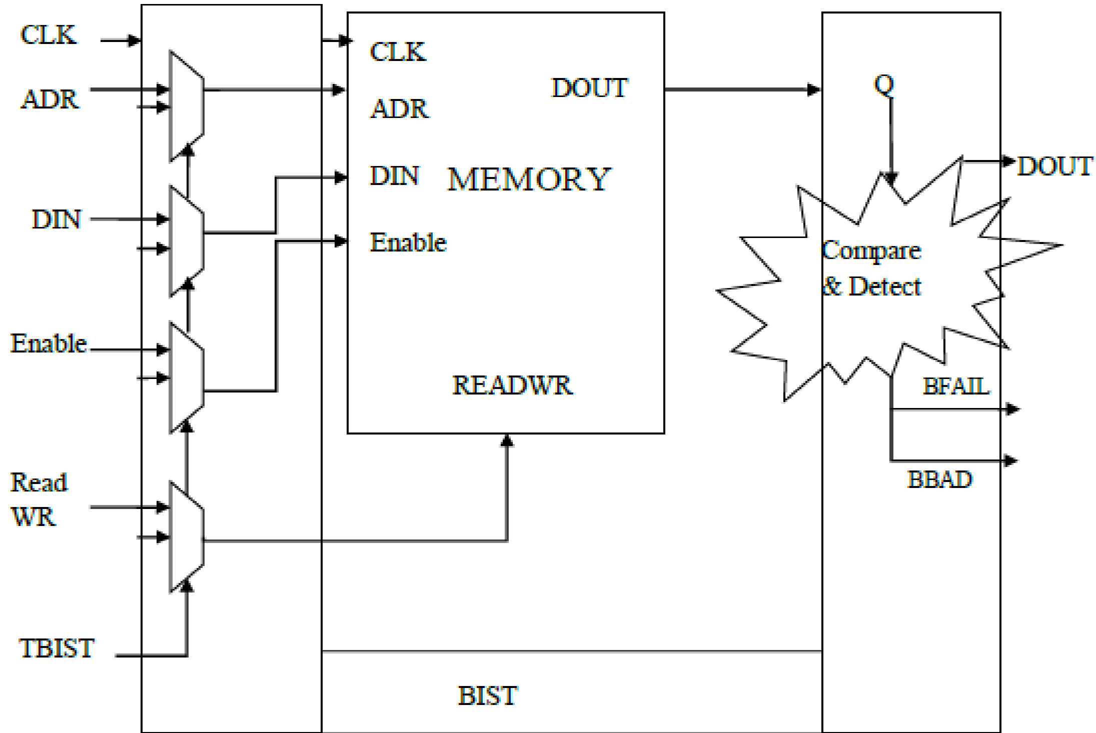


Figure 1. Block Diagram of BIST

In the figure 1, when Test signal is low normal read and write operations will be performed. Clock (CLK), address (ADR), data (Din), enable, memory read writes (ReadWR) are the corresponding inputs given by the user. When test signal is high BIST will take over the control of memory. BIST will provide the input to be written on the specific memory address and will read the desired memory address. Thus it will compare and detect the fault. When fault is detected fault signal will be high and fail signal will give the faulty location. To provide patterns for testing March algorithms are used in BIST. A March test consists of a finite sequence of March elements. An operation can consist of writing a 0 into a cell (w0), writing a 1 into a cell (w1), reading an expected 0 from a cell (r0), and reading an expected 1 from a cell (r1). Some of the most popular notations for MARCH tests are described as follows:

↑: address ordering transition in ascending order

↓: address ordering transition in descending order

↕: address ordering can alter in either way

**r0**: read action (reading a 0 from a cell)

**r1**: read action (reading a 0 from a cell)

**w0**: write action (writing a 0 to a cell)

**w1**: write action (writing a 0 to a cell).

Six different march algorithms namely MATS, MATS+, March X, March Y, March LR, March C- are used in designed BIST. Their corresponding fault coverage is given in the Table 1. March LR algorithm has the maximum fault coverage and is most efficient of all. MATS, MATS+ are simple and fast operating algorithm but with low fault coverage. Faults namely SAF (stuck at fault), TF (transition fault), CF (coupling fault), ADF (address decoder fault), RF (retention fault), LF (linked fault), can be easily detected using March LR algorithm.

Table 1. Fault Coverage of March Algorithms

| **Test** | **March Elements** | **Fault Coverage** |
|---|---|---|
| MATS | {↕(w0); ↕(r0,w1); ↕(r1)} | Stuck at fault and Address decoder fault |
| MATS+ | {↕(w0); ↑(r0,w1); ↓(r1,w0)} | Stuck at fault and address decoder fault and transition fault |
| MARCH X | {↕(w0); ↑(r0,w1); ↓(r1,w0); ↕(r0)} | Stuck at fault, transition fault, address decoder fault, some coupling fault |
| MARCH Y | {↕(w0); ↑(r0,w1,r1); ↓(r1,w0,r0);↕(r0)} | Stuck at fault, transition fault, address decoder fault, some coupling and linked fault |
| MARCH C- | {↕(w0); ↑(r0,w1); ↑(r1,w0);↓(r0,w1); ↓(r1,w0); ↕(r0)} | Stuck at fault, transition fault, address decoder fault, coupling and retention fault |
| MARCH LR | {↕(w0); ↓(r0,w1); ↑(r1,w0,r0,w1); ↑(r1,w0);↑ (r0,w1,r1,w0); ↑(r0)} | Stuck at fault, transition fault, address decoder fault, all coupling and linked fault |

## 3. REDUNDANCY ALLOCATION ALGORITHM

Redundant memory is the spare memory only used to repair faults. Normal memory operation on specific redundant memory address occurs only when it has replaced the faulty main memory. Redundancy used in the designed BISR is two dimensional (spare rows and spare columns).

Repair rules:

- The first rule is that if there are multiple faults in a row; repair the faulty row by spare row. Save the location of the faulty row corresponding to the redundant row used.

- The second rule is that, if there is one bit fault in a memory address then it will be repaired using the redundant column within that row. Store the faulty bit location and Spare row will not be used.

## 4. BUILT IN SELF REPAIR

Built In Self Repair (BISR) comprises Memory Built In Self Test (MBIST) and Built In Redundancy Allocation (BIRA).Architecture of BISR is discussed in figure 2 below:

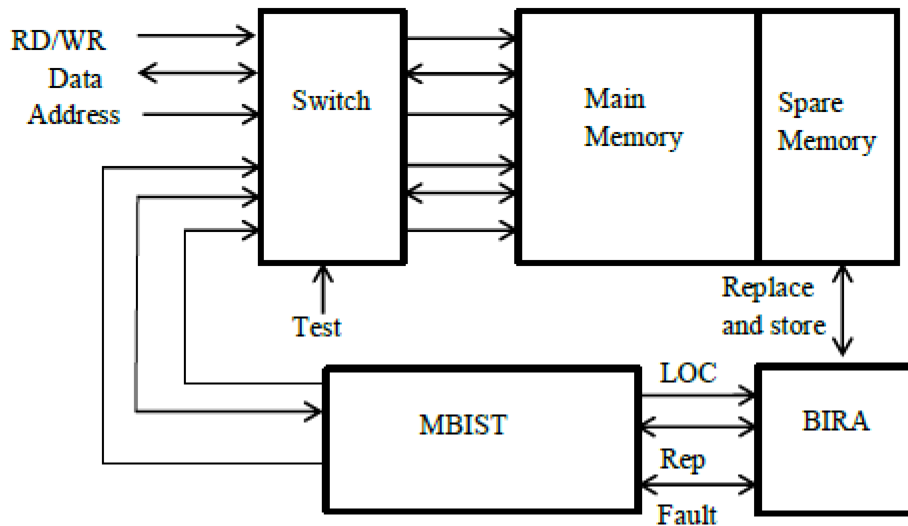


Figure 2. Architecture of BISR

In this BISR memory of 1100x8 is designed. Main memory is of 1KB and rest is redundant memory. In normal mode simple operations like reading and writing memory are carried out. Inputs like data, address and RD/WR are given. RD/WR input decides whether data is to be written or read. Data is the 8 bit data which is either to be written or read. Address is the 10 bit address of memory location where data is to be retrieved or to be stored. When test input is high MBIST takes over control of the memory. MBIST provides all inputs that are Data, Address and RD/WR to the memory. MBIST writes the memory either „0‟ or „1‟ in a specific order and reads back the memory to check the faulty locations. If fault is detected then fault signal goes high and location of the fault is given to the BIRA. It assigns the redundant memory corresponding to the faulty memory according to the repair rules. After allocation fault signal is turned low. Six different BISR are designed using Xilinx tool. Comparative analysis of all the designed algorithms is also performed in terms of number of slices, delay and power dissipation as shown in Table 2.

Table 2. Comparative Analysis of all BISR Designs

| **Algorithm** | **No. of slices used** | **Delay (ns)** | **Static Power (W)** | **Dynamic Power (W)** |
|---|---|---|---|---|
| **MATS** | 14775 | 4.603 | 4.400 | 0.860 |
| **MATS+** | 18200 | 5.299 | 4.400 | 0.861 |
| **MARCH X** | 18214 | 5.230 | 4.407 | 1.048 |
| **MARCH Y** | 21739 | 5.289 | 4.409 | 1.111 |
| **MARCH C-** | 21341 | 5.468 | 4.410 | 1.140 |
| **MARCH LR** | 19948 | 5.638 | 4.409 | 1.125 |

## 5. DYNAMIC POWER REDUCTION

The methodology to reduce power uses the principle that is: reorder the original test such that signal transitions on address line are minimized without affecting the fault coverage. Switching activity on each address line depend on the address counting method (order in which addresses are enumerated while reading and writing the memory). As shown in the Table 3, to reduce the dynamic power the first two march elements of the all algorithms are modified and merged such that transition on address line is reduced. In original test counter is used one more time than in low power test. Thus, this leads to the reduction in dynamic power. Reduction in dynamic power is shown in Table 4 for all the algorithms.

Table 3. Algorithm for Test

| **March Test** | **Original Test** | **Low Power Test** |
|---|---|---|
| MATS | {↕(w0);↕(r0,w1);↕(r1)} | {↕(w0,r0,w1); ↕(r1)} |
| MATS + | {↕(w0);↑(r0,w1);↓(r1,w0)} | {↕(w0);↑(r0,w1);↓(r1,w0)} |
| March X | {↕(w0);↑(r0,w1);↓(r1,w0); ↕ (r0);} | {↑(w0,r0,w1);↓(r1,w0); ↕(r0);} |
| March Y | {↕(w0);↑(r0,w1,r1);↓(r1,w0,r0); ↕ (r0);} | {↑(w0,r0,w1,r1);↓(r1,w0,r0);↕(r0);} |
| March C- | {↕(w0);↑(r0,w1);↑(r1,w0);↓(r0,w1);↓(r1,w0); ↕(r0);} | {↑(w0,r0,w1);↑(r1,w0);↓(r0,w1);↓(r1,w0);↕(r0);} |
| March LR | {↕(w0);↓(r0,w1); ↑(r1,w0,r0,w1); ↑(r1,w0);↑(r0,w1,r1,w0); ↕(r0);} | {↕(w0,r0,w1); ↑(r1,w0,r0,w1); ↑(r1,w0);↑(r0,w1,r1,w0);↕(r0);} |

Table 4 Power Reduction

| Algorithm | Dynamic Power | | Reduction (W) | Reduction (%) |
|---|---|---|---|---|
| | Original Test | Low Power Test | | |
| MATS | 0.860 | 0.764 | 0.096 | 11.16 |
| MATS+ | 0.861 | 0.774 | 0.087 | 10.10 |
| March X | 1.048 | 0.933 | 0.115 | 11 |
| March Y | 1.111 | 0.981 | 0.31 | 11.7 |
| March C- | 1.140 | 0.988 | 0.152 | 13.3 |
| March LR | 1.125 | 0.980 | 0.145 | 12.9 |

## 6. SIMULATION RESULTS

Size of the memory designed is 1KB (1024x8 bits). Size of the redundant memory is 76x8 bit. Synthesis is performed using Xilinx project navigator 14.6 tools. Technology mapping chosen was Virtex 6. Simulation is performed on Isim simulator. Faults are injected in the memory deliberately at positions „2", „4" and „6". At position „2" single bit fault is injected and at „4" and „6" multiple bit faults are injected. After giving input test high faults are detected using MBIST and fault location are stored. For repair the „4" and „6" locations are repaired using redundant location and „2" location is repaired using redundant column. Location of the faulty memory is stored corresponding to the redundant replacement. Simulations of fault injection, detection and repairing using rows and column are shown in the figure 3 and 4.

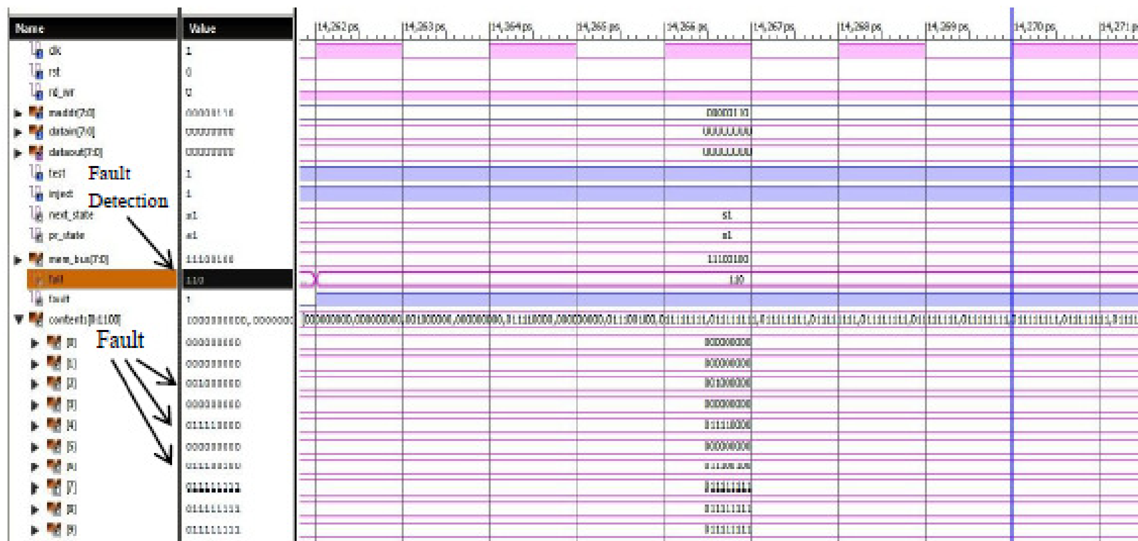


Figure 3. Fault Injection and Detection

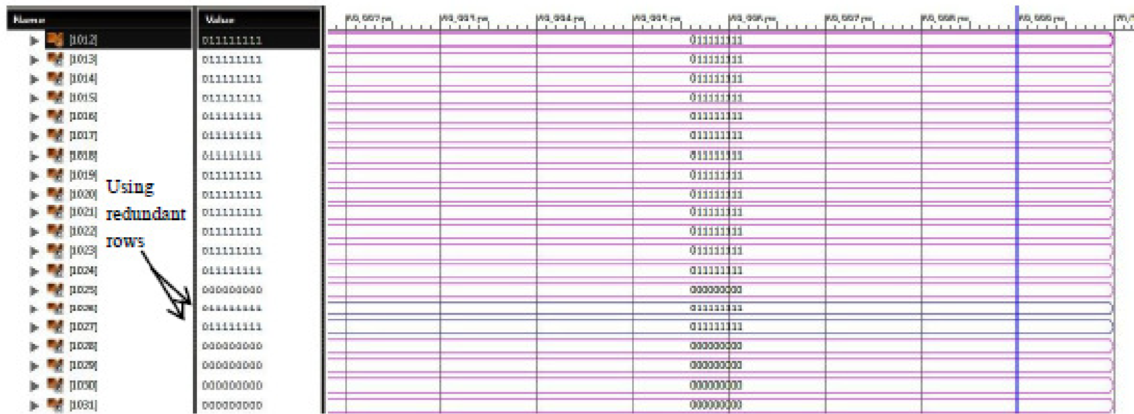


Figure 4. Using Redundant Rows After Repair

In figure 3 faults injection and detection are shown in the simulation. In figure 4 fault is being repaired using redundant rows.

## 7. CONCLUSION

In this paper, BISR architecture is designed to test and repair 1KB memory. BIST module detects the fault using six March algorithms (MATS, MATS+, March X, March Y, March C- and March LR). Simulation results show that faults are detected and repaired using redundant memory (spare column and rows). Single bit fault is repaired with spare column within that row and multiple bit faults are repaired using redundant row. Analysis of device utilization and delay has been done using Xilinx synthesis tool for all MBIST. BISR design using all MBIST is compared altogether. BISR using March LR algorithm has maximum fault coverage. BISR using March C- algorithm uses more device slices among other BISR designs. BISR using MATS algorithm is easy to design and implement which also has low dynamic power dissipation. Power is calculated using Xpower analyser and dynamic power is reduced by reducing the memory address transition in the algorithm. Dynamic power is reduced around 10%-13% for different BISR designs.

## REFERENCES

[1] Jin-Fu Li, Jen-ChiehYeh, Rei-Fu Huang, Cheng-Wen Wu, “A built-in self-repair design for RAMs with 2-D redundancy,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.13, no.6, pp.742-745, June 2005.

[2] Yu-Ying Hsiao, Chao-Hsun Chen, Cheng-Wen Wu, "A built-in self-repair scheme for NOR-type flash memory,” In: Proceedings of 24th VLSI Test Symposium, April 30 2006-May 4 2006.

[3] Banupriya, C., Chandrakala, S., "A low power Built in Self Repair technique for word oriented memories," In: Proceedings of International Conference on Electronics and Communication Systems (ICECS), 13-14 February 2014.

[4] Van de Goor, A.J., Gaydadjiev, G.N., Mikitjuk, V.G., Yarmolik, V.N., "March LR: a test for realistic linked and unlinked faults," In: Proceedings of 14th VLSI Test Symposium, 28 April-1 May 1996.

[5] Shyue-Kung Lu, Yu-Chen Tsai, Hsu, C.H., Kuo-HuaWang, Cheng-Wen Wu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.14, no.1, pp.34-42, January 2006.

[6] Seongmoon Wang, "A BIST TPG for Low Power Dissipation and High Fault Coverage," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.15, no.7, pp.777-789, July 2007.

[7] Lee, M., Li-Ming Denq, Cheng-Wen Wu, "A Memory Built-In Self-Repair Scheme Based on Configurable Spares," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.30, no.6, pp.919-929, June 2011.

[8] Tsu-Wei Tseng, Jin-Fu Li, Hsu, Chih-Chiang, "ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.18, no.6, pp.921-932, June 2010.

[9] Lala, Parag K. "Transient and permanent fault injection in VHDL description of digital circuits." IEEE Transactions on Circuits and Systems, vol.3, no.12, pp. 192. April 2012.

[10] Singh, Balwinder, Sukhleen Bindra Narang, and Arun Khosla. "Modeling and Simulation of Efficient March Algorithm for memory Testing," In: Proceedings of conference on Contemporary Computing Springer Berlin Heidelberg, pp. 96-107, June 2010.

[11] Jyotika; Singh, B., "Memory yield and repair rate improvement scheme using built in self repair techniques," In: proceedings of IEEE conference on Innovative Applications of Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH), pp.350-354, 28-29 Nov. 2014.

[12] A. A. Ivaniuk, "Random Access Memory Faults Descriptions and Simulations using VHDL," In: Proceedings of 14th International Conference on Microelectronics & Computer Science, pp 29-33, June 2007.