



A Reconfigurable High Speed Dedicated BISR Scheme for Repair Intra Cell Faults in Memories

Amgoth Srinivas

PG Scholar, Dept of ECE (VLSI),
CMRIT, Kandlakoya (v), Medchal
Road, Hyderabad, Telangana

Dr. A. Balaji Nehru

Professor, Dept of ECE, CMRIT,
Kandlakoya (v), Medchal road,
Hyderabad, Telangana, India

Ms V. Sumathi

Assistant professor, Dept of ECE,
CMRIT, Kandlakoya (v), Medchal
road, Hyderabad, Telangana, India

ABSTRACT

Shrinking process technology has the advantage of lower area of Integrated Circuits (ICs). This has allowed adding more hardware (features) to existing circuits and enhancing the existing features, Example: - adding more cores to a micro-processor or increasing the resolution of the Video processing hardware of a mobile phone, etc. Execution of complex algorithms need more local memories (SRAMs) embedded in the hardware. As memories are densely packed structures compared to logic (gates and flip flops) the probability of fault occurrence in memories is higher. Thus, adding more complex logic has increased the probability of fault occurrence in ICs and thus decreasing the yield. In this paper we present a Reconfigurable Built in Self Repair (Re-BISR) technique to repair the faults in embedded memories. We also employ Error Correction Codes (ECC) to repair single bit faults in memories. Both the above techniques combined allow us to repair the faults and thereby increasing the yield and reliability of an IC. Re-BISR can repair the faults of several memories in an IC and thus has lesser hardware overhead compared to a dedicated BISR scheme where each RAM has a dedicated BISR module. However, Re-BISR is considerably slower compared to dedicated BISR as RE-BISR operates serially on each memory.

Future version of Re-BISR contains a programmable MBIST scheme to accommodate several March algorithms and also include virtual blocks for redundant memories to increase the repair rate.

We implement the project using Xilinx ISE tool for simulation and synthesis and the code is written in Verilog HDL.

Keywords: *Built-in redundancy analysis (BIRA), built-in self-repair (BISR), built-in self-test (BIST), yield, Re-BISR*

I. INTRODUCTION

The VLSI manufacturing technology advances has made possible to put millions of transistors on a single chip. 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 will be helpful for 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. Their result in Reduction in cost, size and lower the power consumption, and increases the performance, reliability, reuse capability and 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.

With the trend of SOC technology, for any system, for Successful implementation high density and high capacity embedded memories are required. Although it benefits the end user, but these manufacturing process advances are not without limitations. In particular, high density memories in combination with these kinds of process limitations can result in poor over all yields. That is, RAMs have to face more serious problems of yield and reliability than any other embedded cores in an SOC. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. For such

purpose, designers usually propose the redundancy repair logics using spare rows and/or columns to improve the yield and better in functionality.

However, the negative impact of the on yield, due to redundancy increases the silicon area. To maximize the yield, redundancy analysis is necessary. Conventionally, redundancy analysis (RA) is performed on the host computer of the ATE if it is an online process or on a separate computer if it is an offline process. In other way it is time consuming since RA algorithms are complex and the memories that implement the redundancies are usually large. Moreover, the embedded memories are harder to implement by using Automatic Test Equipment (ATE). The technique, BISR (Built in Self Repair) is a promising and most popular solution for enhancing the yield of memories with the redundancy logic. Furthermore, redundancies of memories are not only for defects, but also redundancies can be used to recover yield due to process variation in addition to yield recovery for defects.

Importance of Built-In Self-Test and Built-In Self-Repair

The operation to memory repair the memory can be performed by using external equipment to test the memory, identify the faults and drive a laser beam that performs the repair. To avoid the laser beam, Electrical fuses or anti-fuses can also be used. Taking the complexity point of view, it is very difficult and costly to test the embedded memories from an external memory tester as the chip density continues to grow. In addition to, the accessibility of the embedded memories is low for external testers. To overcome these, the developments replace external equipment by BIST (Built in Self Test) and BISR schemes in order to maintain at reasonable levels the test and repair cost of embedded memories. An additional to this, the advantage is that the BIST and BISR can test and repair embedded memories at any time during the product life. This can reduce the maintenance cost, and increase reliability and product lifetime.

Objective of the project:

RAMs in an SOC will have the various sizes, different number of redundancies, and even different types of redundancy organizations. If each repairable RAM, uses the self contained BISR circuit, then it results in the area cost of BISR circuits in an SOC becomes high. This results in converse effect in the yield and performance of RAMs. To reduce the area cost, several

processor based BISR schemes are also proposed. In these BISR schemes, a BISR circuit can be able to repair multiple RAMs. However, a processor based BISR scheme uses specific instructions to construct the redundancy analysis algorithm. This results in long redundancy analysis time, since a statement of the redundancy analysis algorithm may need multiple instructions to realize it. Therefore, the time efficient and area efficient BISR scheme is needed to improve the yield and performance of RAMs in SOC's economically.

The solution to the above problem is reconfigurable BISR (Re-BISR) scheme, which is implemented in this thesis. The Re-BISR can be shared by multiple RAMs with different sizes and redundancy organizations. This can reduce the area cost of the BISR circuits in an SOC. Also, an efficient reconfigurable BIRA (Re-BIRA) scheme is used to allocate 2D redundancies of multiple RAMs.

Performance of BISR:

In today's advanced submicron technologies that can be allow the implementation of multiple memories on a single chip. This is of the high density; memories are more prone to faults. These faults impact the total chip yield. One way to solve this problem is to enhance the memory by redundant memory locations. The address mapping of the fault free working memory is programmable within certain limits. In order to do so, a memory test is needed to identify the faulty regions.

The memory is tested by external test hardware or by on chip dedicated hardware (memory BIST). The second testing strategy is the preferred method for embedded memories. After memory testing the memory address map is programmed by means of volatile or non-volatile storage on or off chip. To provide the test pattern from a memory BIST a multiplexer in front of the memory is widely used. The redundant spare rows and spare columns are provided into the memory, in order to replace them with faulty locations in the memory. This impacts the performance and area conditions of the memory.

The memory is repaired during testing by storing faulty addresses in registers. These particular addresses can be streamed out after completion of the test. Furthermore, the application can be started immediately after the memory BIST passes. The redundancy logic calculation will not increase the test time of the memory BIST.

The memory Built-in self repair (MBISR) concept is

the one which provides an interface between memory Built-in self repair (MBIST) logic and redundancy wrapper which can be used for storing faulty addresses. This allows using already existing MBIST solutions.

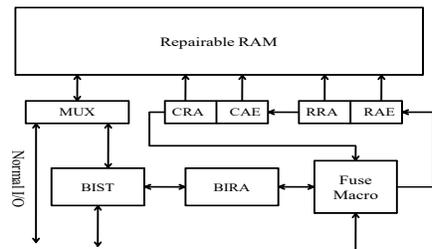


Figure: MBISR scheme for embedded RAMs.

Repairable RAM: A RAM with redundancies and reconfiguration circuit is called as a repairable RAM. Figure 2.2 depicts an example of an 8*8bit-oriented RAM with 1 spare row and 1 spare column. If a spare row is allocated to replace a defective row, then the row address of the faulty row is called row repair address (RRA). Then a decoder decodes the RRA into control signals for switching row multiplexers to skip the defective row if the row address enable (RAE) signal is asserted. Similarly the reconfiguration of the faulty column and the spare column is performed, i.e., give a column repair addresses (CRA) and assert the column address enable signal to repair the defective column using the spare column.

BIST Circuit: It can generate test patterns for RAMs under test. While a fault in a defective RAM is detected by the BIST circuit, the faulty information is sent to the BIRA circuit.

BIRA Circuit: The function of this circuit is to collect the faulty data from the BIST circuit and allocates redundancies according to the collected faulty information using the implemented redundancy analysis algorithm.

This is a hardware command implementation of the selected algorithm to test memory, usually in the form of an FSM state machine. The MBIST figure is shown in Figure 3.6 below.

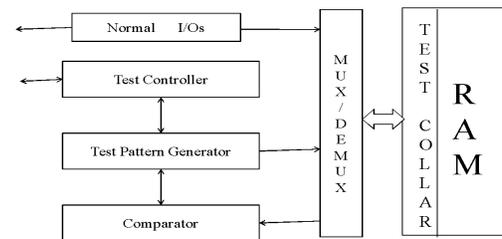


Figure: Memory BIST architecture

March-based test algorithms:

A March-based test algorithm is a finite sequence of March elements. A March element is specified by an address order and a number of reads and writes. Examples of some March based tests are MATS, MATS+, Marching 1/0, March C-, March Y, March A, March B, and etc. Since March-based tests are all simple and possess good fault coverage, they are the dominant test algorithms implemented in most modern memory BIST.

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.

Reconfiguration techniques:

Redundancy analysis and repair procedures are interrelated. During testing, when failures are identified and located, a redundancy analysis procedure determines which failures can be repaired using the given redundant rows and columns. On the basis of this redundancy analysis, either a hard or a soft repair procedure is executed.

In general, hard repair uses fuses, anti fuses, or laser programming to disconnect rows and columns with faulty bits and replace them with redundant rows or columns. This method, long used by commodity memory vendors, has been adopted by ASIC and SOC vendors and is currently the most widely used method for stand-alone as well as embedded memories.

Soft repair uses an address-mapping procedure to bypass the faulty address location. This scheme links BIST and power-on, it will result in every time power is switched on, and memory is tested via BIST. During

this testing, the addresses of all failing locations are stored separately, and an address mapping procedure maps them on to redundant fault-free addresses. Although a dedicated finite-state machine serves to implement an address-mapping procedure, it can also be implemented efficiently through an on-chip microprocessor or other logic cores.

Built in redundancy analysis (BIRA):

As the density of memory has increased, the number of related defects has also increased. To increase device yield, many manufacturers use incorporated redundancy that can be used to replace faulty modules. Therefore, the implementation of effective redundancy algorithms is essential.

Many redundancy analysis algorithms for performing redundancy allocation at ATE have been proposed. However, these algorithms are not adopted to be realized in built-in circuits. Thus, built-in redundancy-analysis (BIRA) algorithms which can cost-effectively be realized with built-in circuits are required for BISR schemes.

Kawagoe et al. presented a comprehensive real-time exhaustive search test and analysis (CRESTA) scheme for bit oriented memories. The CRESTA uses an exhaustive search approach to allocate redundancies of a RAM such that it can achieve the optimal repair rate. However, this scheme is only for bit-oriented memories and the hardware cost for implementing this scheme is drastically increased with the number of redundancies of a RAM. Xiao gangetal. Extended the CRESTA scheme to support the repair of word-oriented memories. A column repair vector stores the information for column repair temporarily, such that the BIRA design can perform redundancy allocation at-speed. Essential spare pivoting (ESP) repairs faulty cells when the number of faulty cells is more than an essential number in the same address, and local repair-most (LRM) repairs faulty cells using repair-most method with a local bitmap. ESP incurs the lowest hardware overhead costs, but ESP and the LRM do not have 100% repair efficiencies. In particular, ESP has the lowest repair efficiency when the number of spares is large. Intelligent solve, which is based on the exhaustive binary search tree, is an algorithm to reduce the number of backtracks required to search the repair solution. However, Intelligent solve requires a long search time in nearly all serious cases.

Recently, a memory test methodology using built-in redundancy analysis (BIRA) algorithms was

introduced. To overcome the disadvantages of CRESTA, BIRA algorithm called Range Checking First Algorithm (RCFA) is implemented for allocating 2D redundancy. If the numbers of column entries are greater than the number of row entries, then spare row is allocated. Else, spare column is allocated. When the number of row entries and column entries are equal then spare column is allocated.

Spare row and spare column (2-D Redundancy):

With this approach, spare rows and spare columns are added in the memory array. Either a spare row or a spare column can be used to replace a faulty cell. This approach is more efficient than the first approach when multiple faulty cells are detected.

However, the complexity for finding the optimal spare allocation is NP-complete. Moreover, owing to the high bandwidth of embedded RAMs, more spare columns and rows are required to achieve sufficient chip yield. This in turn increases the fabrication cost.

Built-in self-test (BIST):

Need for BIST:

Due to the limitations of the fabrication process, many components of the chip are prone to structural errors which may lead to improper functioning of the chip. To detect these errors before the chip goes in to a system is very important to reduce the risk of the system failing field trials.

The different components that can be affected by these fabrication errors are standard cells (gates), wires and high density memories. Based on the nature of the build and functionality of these components, the testing process for these errors can be grouped as 1) logic testing (standard cells and wires) and 2) memory testing (memories).

The test process is generally done by the external device called ATE (Automatic Test Equipment). The ATE applies a pre-generated set of inputs called as test vectors to the chip and compares the outputs to a predefined set of expected values.

The disadvantages of using ATE for testing are

1. Cannot perform At-Speed testing.
2. Very expensive i.e. cost increases in proportion to the number of test vectors.
3. One ATE machine can only test one chip at a time

there by causing delays for systems that are to be manufactured on a large scale.

These disadvantages can be overcome by BIST. BIST requires a small amount of extra hardware to be inserted into the chip. The basic concept of BIST involves the design of test circuitry around a system that automatically tests the system by applying certain test stimulus and observing the corresponding system response. BIST can find the logic to verify a faulty-free status automatically, without the need for externally applied test stimuli and without the need for the logic to be part of a running system.

Memory BIST (MBIST):

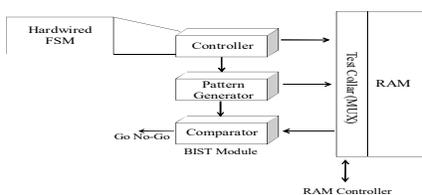


Figure: Memory BIST architecture

Memory cores are obviously among the most universal ones-almost all system chips that contains some type of embedded memories. However, to provide a low cost-cost test solution for the on-chip memory cores is not a big task.

Digital systems are composed of data paths, control paths and memories. Defects in memory arrays are generally due to shorts and opens in memory cells, address decoder and read/write logic. These defect scan be modeled as single and multi cell memory faults. 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.

The addition of extra circuitry to facilitate testing of memory chips, called DFT, or to allow the test mechanism to be completely contained within the chip, called BIST, has become of interest. MBIST, as its name implies, is used specifically for testing memories. It typically consists of test circuits that apply, read, and compare test patterns designed to expose defects in the memory device. MBIST has been proven to be one of the most cost-effective and widely used solutions for memory testing for the following reasons: no external test equipment, reduced development efforts, tests can run at circuit speed to yield a more realistic test time,

on-chip test pattern generation to provide higher controllability and Observability.

There now exists a variety of industry-standard MBIST algorithms, such as the walking 1/0 algorithm, march algorithm, the checkerboard algorithm, and the varied pattern background algorithm. With MBIST, the entire memory testing algorithm is implemented on-chip, and operates at the speed of the circuit, which are 2 to 3 orders of magnitude faster than a conventional memory test. Most MBIST schemes exploit the parallelism within the memory device to achieve a massive reduction in test time. This is done by a test mode where more than one memory cell is accessed with each address, usually by accessing the entire row of cells on a word line for a single read or write operation. For n cells in the memory, with \sqrt{n} rows \sqrt{n} columns, this reduces test time by a \sqrt{n} factor. 17

The memory BIST is similar to the basic BIST circuit, where CUT is replaced with the RAM. Additional block called controller is needed. A Controller is a hardware realization of a selected memory test algorithm, usually in the form of a Finite State Machine (FSM).

Proposed Re-BISR Scheme:

The block RAM details table is the one which can be deals with the information which includes the memory data width, memory depth, number of spare rows and number of spare columns used for storing the configurations of RAMs. RAM details table is of size 4×16 . FSM is the main block that acts as a controller for generating the control signals during testing and repairing processes.

The overall RAM Re-BISR flow can be expressed as follows. The RAM configurations (details) should be known by these two circuits Before the BIST circuit and the Re-BIRA circuit start testing and repairing the RAMs. This is can be done by the FSM, where it generates the necessary control signals that are required for sending the RAM configurations from RAM details table to BIST and Re-BIRA circuits.

Table (I): Configurations of RAMs

| RAM No | Data Width * Memory Depth | No. of Spare Rows | No. of Spare Columns |
|--------|---------------------------|-------------------|----------------------|
| RAM 0 | 8 * 16 | 2 | 2 |
| RAM 1 | 16 * 32 | 2 | 3 |
| RAM 2 | 16 * 128 | 3 | 2 |
| RAM 3 | 16 * 256 | 0 | 0 |

In Figure shown above is the modified block diagram of the proposed Re-BISR scheme for repairing multiple RAMs in

An SOC.

The Four repairable RAMs with deferent sizes are having different number of redundancies are considered in our proposed Re-BISR scheme as shown in the below figure. All these RAMs are word oriented memories and there configurations are as listed in the above table.

The table (II) shows the stuck-at-faults in the four repairable RAMs at their respective fault row and fault column locations as shown below.

| RAM Number | Fault Row | Fault Column | Stuck-at-fault |
|------------|-----------|--------------|----------------|
| RAM 0 | 6,7 | 3 | 0 |
| | 14 | 5, 4 | 1,1 |
| RAM 1 | 15, 16 | 13 | 0 |
| | 28 | 14,13 | 1,1 |
| RAM 2 | - | - | - |
| RAM 3 | 243 | 6 | 0 |

Figure: Location of Stuck-at Faults

Reconfigurable implicit excess examination (Re-BIRA):

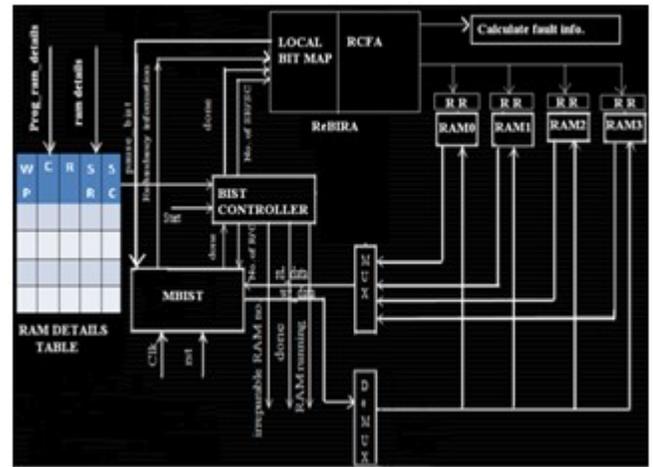


Figure: Block diagram of the proposed Re-BIRA architecture

The proposed piece chart of Re-BIRA is appeared in the above figure .If the BIST recognizes blame, at that point the blame data is sent out to the Re-BIRA hardware, and after that the Re-BIRA performs repetition portion on the fly utilizing the tenets of the actualized excess calculation. The excess calculation utilized here is Range Checking First Algorithm. Blame data incorporates blame line and blame segment addresses and the nearness of blame.

The proposed conspire utilizes a neighborhood bitmap (fault_table) of size 4*64 to store blame data of the flaws distinguished by the BIST circuit. Once the neighborhood bitmap is full, the MBIST is stopped and the Re-BIRA designates redundancies as per the blame data. After, the Re-BIRA designates a repetition to repair a relating flawed line /segment, the nearby bitmap is refreshed and the MBIST is continued. This procedure is iterated until the point when the test and repair process is finished. When one extra component is allotted, the repair marks are sent to the repairable RAM. Repair marks incorporate repair enlist compose, repair enlist information repair enlist address.

Algorithm 1 RCFA: A Range-Checking First Algorithm for allocating 2D redundancy

1. Run BIST; Paused and goes to Step 2 when it detects an abnormal problem.
2. Check to see if the error was corrected. If so, please go to Step 1. Otherwise, go to Step 3.
3. Check if the image is full. If so, go to the next step. If not, go to Step 1.

4. If $NFRE > NFCEL$ (NFCER) provides an alternative backup that can replace CMF.

(If there is a backup column, reserve an existing backup line to replace the RMF). If $NFRE < NFCEL$ (NFCER) allocates the alternate range available for the RMF replacement (if the backup backup row is allocated as a backup alternative to CMF); Otherwise, $NFRE = NFCEL$ (NFCER) replaces the sick element with the largest number of bits that have problems with corresponding backup entries. If spare parts are gone, the memory card cannot be recovered.

Check if BIST is finished. If so, go to the next step. If not, go to step 1 when the local picture is not finished. Go to Step 4 when the image is finished.

Check if the image is empty. If so, please export the repair address and then stop it. If not, go to Step 4.

An example of an explanation of RCFA is provided. It is assumed to be possible to repair RAM with two additional ranges and two columns. Also, there are two corresponding 4x4 thumbnails, as shown in Figure 5.5, where RAR and CAR are not shown briefly. Basic database data 0 or 1 shows the location of the address, address, and address of the column in RAR and CAR corresponding to no damage or inaccuracies. In this example, two different cases were shown.

Case 1: The misuse of RAM is tested by the BIST chain, and the corresponding base Raster positioning status is shown in Figure 5.5 (a). As the basic Raster plot is completed, RCFA previews $NFRE$, $NFCEL$ and $NFCER$. In this case, $NFRE = 4$, $NFCEL = 2$ and $NFCER = 1$. Since the $NFRE$ is larger than $NFCEL$ and $NFCER$, the alternate row instead of CMF is used. In this case, CMF is in the right thumbnail. Therefore, the base backup column in the right indent is used to replace the corresponding column. Assume no errors were found and BIST was performed after the local backup distribution. Then $NFRE = NFCEL = 2$ and the number of infected bits in RMF and CMF is 1. However, the number of backup rows and base columns are 2 and 1, respectively. Therefore, the available rows are used to replace the corresponding broken lines. Finally, the remaining bits can be fixed by the available range or the base backup column. Figure 5.5 (b) shows the repair status when a surplus analysis is performed.

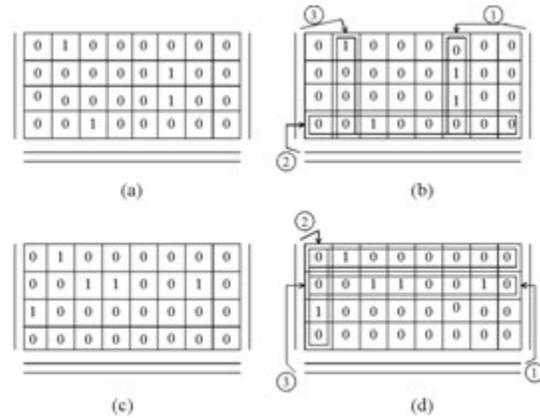
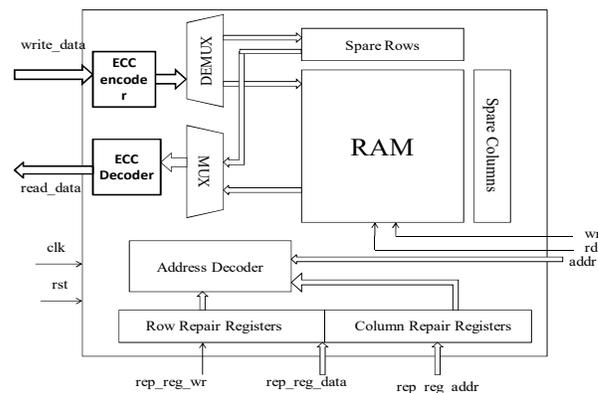


Figure: Example for two different defective RAMs

Case 2: If a local image is finished and the situation is shown in Figure 5.5 (c). In this case, $NFRE = 3$, $NFCEL = 4$ and $NFCER = 1$. Since $NFCEL$ is larger than $NFRE$, RCFA provides a range available for this RMF replacement. Once again, the error is found, and the Beastie is taken after the distribution of the initial reserve. Then $NFCEL = 2$ and the browser that $NFRE$ numbers in RMF and CMF is 1 therefore, or class, are used to replace the locally available disease rows or columns on left help left. Assume that the alternate column is used to replace the problem column. Finally, the remaining bits are repaired from the available backup order. Figure 5.5 (d) shows repair status when abbreviation analysis is performed.

Repairable RAM:

A RAM with redundancies and reconfiguration circuit is called as a repairable RAM. The block diagram of repairable RAM is shown below



A RAM with redundancies and reconfiguration circuit is called as a repairable RAM. The above Figure depicts an example of an 8*8bit-oriented RAM with 1 spare row and 1 spare column. If a spare row is allocated to replace a defective row, then the row address of the faulty row is called row repair address

(RRA). Then a decoder decodes the RRA into control signals for switching row multiplexers to skip the defective row if the row address enable (RAE) signal is asserted. Similarly the reconfiguration of the faulty column and the spare column is performed, i.e., give a column repair addresses (CRA) and assert the column address enable signal to repair the defective column using the spare column.

After the Re-BIRA circuit allotting the redundancies, the repair marks are sent to the repair registers introduce in the repairable RAM. The repair marks incorporates rep_reg_wr (motion for composing the faulty locations in the repair registers), rep_reg_data (damaged locations), rep_reg_addr(line and section repair address registers areas). At that point, the RAM is repaired utilizing this data as takes after. At the point when, either clk flag or rst flag is high, the line repair registers and the segment repair registers are filled to zeroes.

At the point when rep_reg_wr flag is high, if rep_reg_addr is not as much as the quantity of extra lines, at that point the rep_reg_data (inadequate line addresses) is built into the row_repair_address_reg, else if rep_reg_addr is not as much as the quantity of aggregate of extra lines and extra sections, the rep_reg_data is built into the column_repair_address_reg.

The most critical piece (MSB) in the line repair address registers and section repair address registers demonstrates the substantial piece. At the point when the wr flag is high and if MSB is one and the line repair address enroll information in the line repair deliver enlist is equivalent to the address that is given by the MBIST circuit, at that point the information is built into the extra lines through write_data else if MSB in the segment repair address enlist is high then the information is built into the extra segments through write_data. At the point when rd flag is high, the parameter rd_mem is set to high and when the MSB in the column repair address enlist is one then the information display in the extra lines is perused the read_data stick. At the point when the rd flag is high and MSB in the segment repair address enlist is one then the information display in the extra sections is perused read_data stick. At the point when rd flag is low, the grouping of information containing each of the zeroes is perused read_data stick.

The repairable RAM has an ECC based memory with excess bits in each word to repair all single piece issues per word. We utilize hamming code for ascertaining the

check bits. The ECC encoder ascertains the check bits for every approaching information and the information bits and check bits are put away in the memory. The ECC decoder checks for single piece blames in the information read from the memory and adjust one single piece blame. In the event that the shortcomings are more than 1 bit ECC isn't fit for recognizing and remedying the deficiencies and the REBISR conspire is in charge of blunder adjustment through extra recollections.

Error Correcting Codes:

The benefit of changing to advanced innovation from simple innovation is the capacity to identify and adjust mistakes prompted amid transmission or capacity. Blunder recognition and rectification is a whole science today with more current systems proposed ordinary which have focal points over their past ages as far as energy of region or speed. There are numerous blunder discovery and rectification systems which are planned or streamlined for a specific application.

Keeping in mind the end goal to distinguish or revise mistakes, we need excess i.e. information which does not convey any novel data and are available for blunder identification and adjustment process.

Case:-

In the event that a 3 bit esteem is to be transmitted over a medium which is inclined to impedence, the likelihood of recognizing mistake is zero without excess.

A 3 bit information has 8 conceivable mixes 000, 001, 010, 011, 100, 101, 110 and 111. Because of mistake, any of the information can be changed into any of the other 7 esteems. In any case, all the 8 blends being legitimate we can't decide the information that has touched base at the beneficiary is mistake information or right information.

To recognize or remedy any blunder, we should have the capacity to separate the legitimate blends from the invalid ones. For instance, on the off chance that we utilize a 4 bit an incentive to speak to just 8 substantial blends, we can without much of a stretch distinguish a blunder where a legitimate code word changes into an invalid code word as there are 8 invalid code words. This plan is prevalently known as an equality conspire where we add an additional excess piece to a legitimate information which changes it into a code word. All single piece mistakes can be distinguished by this equality plot.

To recognize all "X" bit blunders the separation of the code ought to be "X+1" and

To remedy all "X" bit blunders the separation of the code ought to be "2X+1".

Separation of the code is the base number of bit contrasts between any two code words.

There are numerous mistake amendment strategies. The multifaceted nature of the code word i.e. the excess data estimation approach and the blunder discovery and remedy systems may differ in view of the code word and the quantity of mistakes to be identified or redressed. Hamming Code is an exceptionally mainstream single image mistake remedy conspires and is likely the most proficient as well.

Example of Hamming code generation:-

If we have 4 bits of data d3, d2, d1 and d0. The process of generating the hamming code word is

We have to insert check bits in all bit positions with index which is a power of two like 2⁰ = 1, 2¹ = 2 and 2² = 4. Data bits will occupy the other positions such as 3rd, 5th, 6th, 7th, 9th, etc. Check bit 1 also called as C1 will have a value equal to the xor of all bits whose index contains "1" in the LSB. C2 will have a value equal to the xor of all data bits whose index contains "1" in the bit next to the LSB, etc. Example:-

Data bits = 1011 -> d3 = 1 , d2 = 0, d1 = 1, d0 = 1

Code word d3d2d1C4d0C2C1

Index 7 6 5 4 3 2 1

| Index | BIT | binary |
|-------|-----|--------|
| 1 | C1 | 001 |
| 2 | C2 | 010 |
| 3 | D0 | 011 |
| 4 | C4 | 100 |
| 5 | D1 | 101 |
| 6 | D2 | 110 |
| 7 | D3 | 111 |

Index of C1 contains "1" in the bit position 0. So, C1 is calculated as xor of all the data bits whose index contain "1" in bit position 0. D0, D1 and D3 have indices 3, 5 and 7 which

$$C1 = D0 \text{ xor } D1 \text{ xor } D3 = 1 \text{ xor } 1 \text{ xor } 1 = 1 \Rightarrow C1 = 1$$

$$C2 = D0 \text{ xor } D2 \text{ xor } D3 = 1 \text{ xor } 0 \text{ xor } 1 = 0 \Rightarrow C2 = 0$$

$$C4 = D1 \text{ xor } D2 \text{ xor } D3 = 1 \text{ xor } 0 \text{ xor } 1 = 0 \Rightarrow C4 = 0$$

Therefore. The code word is

Code word 1 0 1 0 1 0 1 = 1010101

Code word d3d2d1C4d0C2C1

Index 7 6 5 4 3 2 1

Error correction also works on the same principle as code word generation. Based on d3, d2, d1 and d0 in the received code word, the check bits are calculated. The calculated check bits are compared with the received check bits in the received code word. The sum of the indices of the erroneous check bits gives the index of the error bit. Error is corrected by inverting the bit with the specified index.

SIMULATION AND SYNTHESIS RESULTS:

Repairable RAM0:

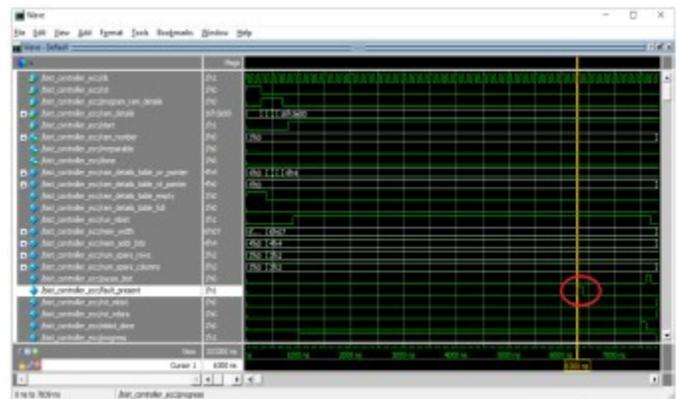


Figure: Repairable RAM0 fault present waveform

In the above figure, MBIST detects that there is fault present in repairable RAM0 before repair.

The above faulty location will be detected or corrected by ECC and corresponding figure will be as shown in the below figure.



Figure: ECC correcting single bit error in repairable RAM0

In the above figure, ECC is correcting a single bit fault in repairable RAM0.

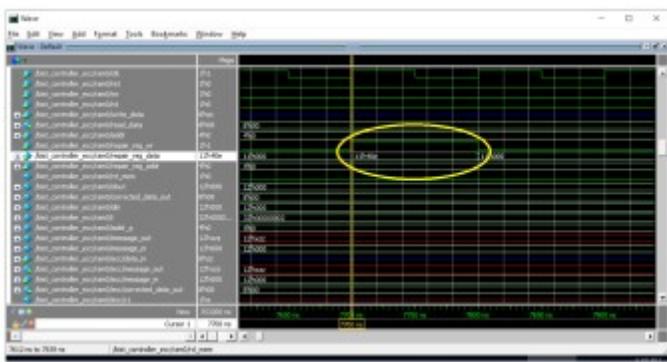


Figure: REBISR writing repair registers of repairable RAM0

- (i) There are 3 fault locations in repairable RAM0, one is a 2 bit fault and the other 2 are single bit faults.
- (ii) Single bit faults are corrected by ECC where-as REBISR is allocating spare location for the 2 bit fault.

Repairable RAM1:

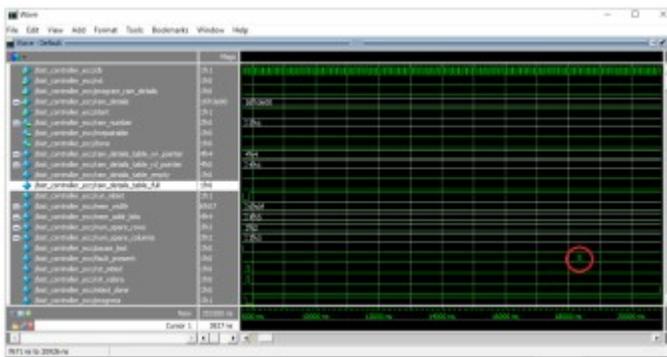


Figure: Repairable RAM1 fault present waveform

In the above figure BIST will notice that there is fault present in repairable RAM1 before it undergoes the repair process.

These errors can be corrected by the error correcting codes, which can be as shown below.

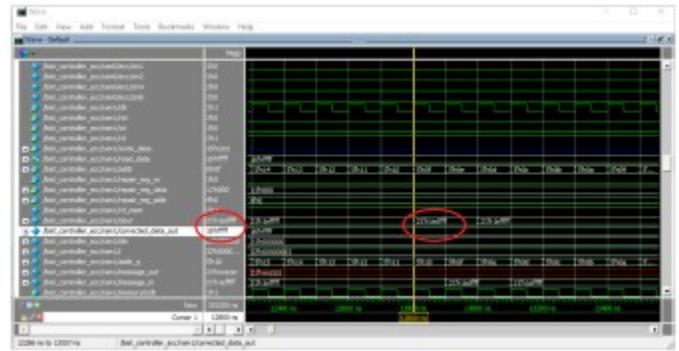


Figure: ECC correcting single bit error in repairable RAM1

In the above figure, ECC is correcting a single bit fault in repairable RAM1

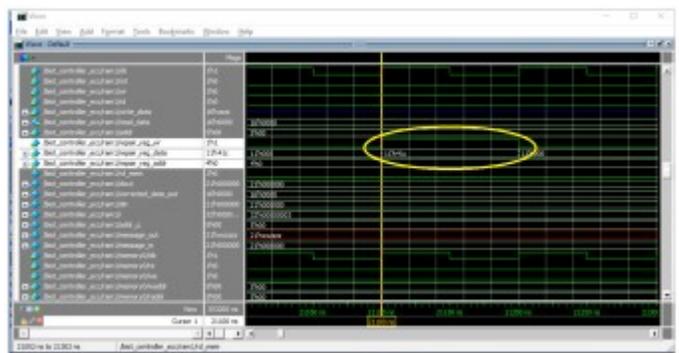


Figure: REBISR writing repair registers of repairable RAM1

There are 3 fault locations in repairable RAM1, one is a 2 bit fault and the other 2 are single bit faults. Single bit faults are corrected by ECC where-as REBISR is allocating spare location for the 2 bit fault.

Repairable RAM2:

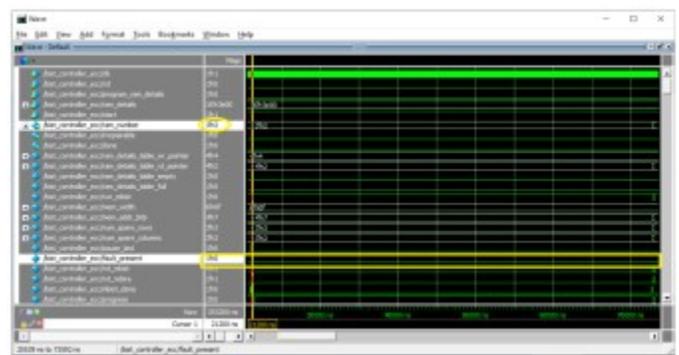


Figure: Re-BISR detecting NO faults in repairable RAM2

From the above figure we can conclude that no faulty

locations in the RAM2

Repairable RAM3:

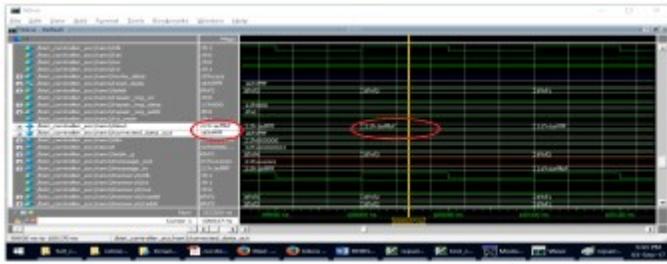


Figure: ECC correcting single bit error in repairable RAM3

In the above figure BIST finds that single bit error and the corresponding Error can be corrected by Error Correcting Codes (ECC). Since these errors are single bit errors that can be corrected by the ECC.

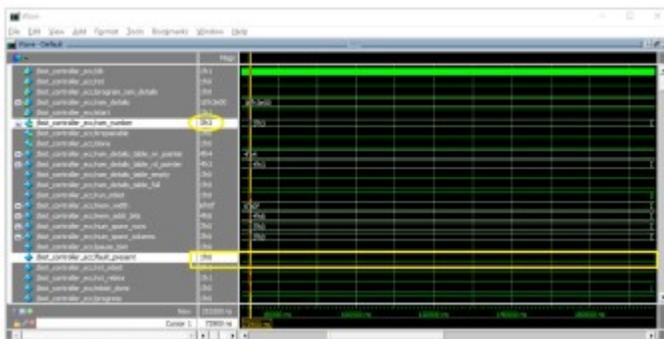


Figure: Re-BISR detecting NO faults in repairable RAM3

The above figure shows the result after correcting the single bit error by using ECC, and it denotes that error-free locations in RAM3. Repairable RAM3 has one single bit fault and the ECC corrects this fault and hence, the MBIST does not detect any faults.

MBIST results after repair:

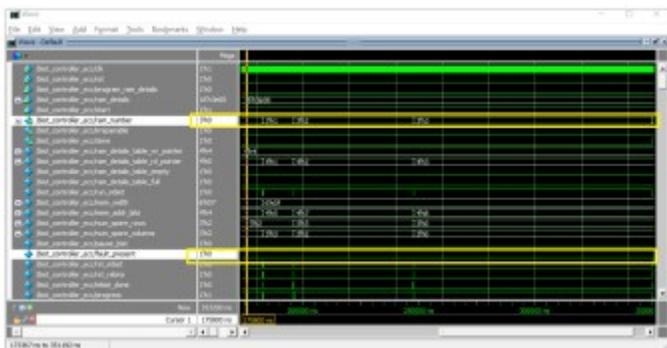


Figure: Re-BISR detecting NO faults in any repairable RAM after repair

We run MBIST for the second time after repairing the faults in the RAMS and we can observe from the above figure that there are no faults detected by MBIST as all the single bit faults are corrected by ECC and the multi bit faults are repaired by REBISR by moving the data to spare location when available.

Design summary of MBIST:

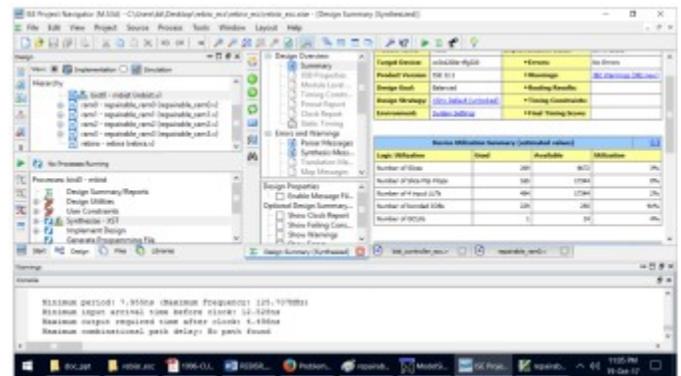


Figure: Design summary of MBIST

The above figure shows that design summary of the memory BIST that can be used in this particular project.

Hamming code Design:

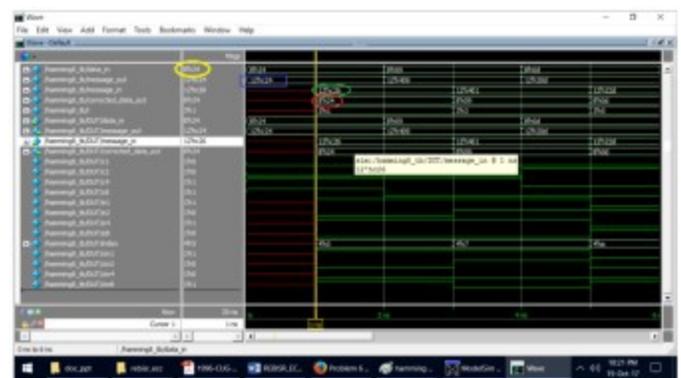
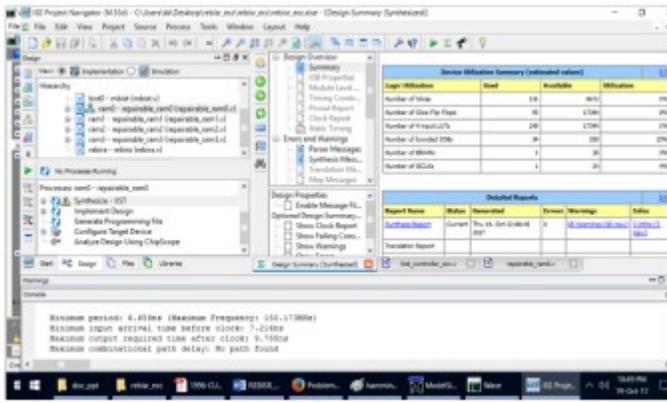


Figure: Hamming Code for 8 bit data

In the above figure, we can see the hamming code recreation for 8 bit information. We have infused the information 8'h24 (yellow circle) and the code word is 12'hc24 (blue square). We have infused mistake into the code word to make it 12'hc26 (green hover) and after revision, we got back 8'h24 (red hover) in the wake of remedying the 1 bit blunder.

Synthesis results of Repairable RAM:

The above figure indicates the, Synthesis results of the repairable RAM.

SUMMARY**Conclusions:**

A reconfigurable BISR plot for repairing multiple repairable RAMs with various sizes and diverse quantities of redundancies has been displayed. An efficient BIRA calculation for 2-D excess assignment has likewise been introduced. The BIRA calculation has been acknowledged in reconfigurable BIRA equipment, with the end goal that it can bolster different RAMs. Test comes about demonstrate that the Re-BISR plot brings about low region cost when contrasted and the devoted BISR. Likewise, the reconfigurable BISR plot has more prominent adaptability than the committed BISR conspire as the previous one backing the repair of numerous recollections. Subsequently, our Re-BISR plot has low range cost compared with the other BISR conspire for general applications.

Applications:-

1. RE-BISR plan can be utilized in any complex SOCs which have different installed recollections executing complex calculations.
2. It is additionally extremely supportive when an innovation hub isn't yet develops and can be utilized to repair the few memory shortcomings amid beginning times of an innovation hub.

Future Scope:-

Future variant of Re-BISR contain a programmable MBIST plan to oblige a few March calculations and furthermore incorporate virtual squares for repetitive recollections to build the repair rate.

REFERENCES:

1. Semico Res. Corp., Phoenix, AZ, USA, ASIC IP Rep., 2007. [Online]. Available: http://www.semico.com/content/se_mico_systemschip-%E2%80%93braver-new-world
2. T. Kawagoe, J. Ohtani, M. Niuro, T. Ooishi, M. Hamada, and H. Hidaka, "A built-in self-repair analyzer (CRESTA) for embedded DRAMs," in Proc. Int. Test Conf., Oct. 2000, pp. 567–574.
3. C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Trans. Rel., vol. 52, no. 4, pp. 386–399, Dec. 2003.
4. [4]P. Öhler, S. Hellebrand, and H.-J. Wunderlich, "An integrated built-in test and repair approach for memories with 2D redundancy," in Proc. 12th IEEE Eur. Test Symp., May 2007, pp. 91–96.
5. W. Jeong, I. Kang, K. Jin, and S. Kang, "A fast built-in redundancy analysis for memories with optimal repair rate using a line-based search tree," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 12, pp. 1665–1678, Dec. 2009.
6. W. Jeong, J. Lee, T. Han, K. Lee, and S. Kang, "An advanced BIRA for memories with an optimal repair rate and fast analysis speed by using a branch analyzer," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 12, pp. 2014–2026, Dec. 2010.
7. A. Tanabe et al., "A 30-ns 64 Mb DRAM with built-in self-test and self-repair function," IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1525–1533, Nov. 1992.
8. [8]Y. Zorian, "Embedded memory test and repair: Infrastructure IP for SOC yield," in Proc. Int. Test Conf., Oct. 2002, pp. 340–349.
9. Y. Zorian and S. Shoukourian, "Embedded-memory test and repair: Infrastructure IP for SoC yield," IEEE Des. Test Comput., vol. 20, no. 3, pp. 58–66, May/June 2003.
10. C.-D. Huang, T.-W. Tseng, and J.-F. Li, "An infrastructure IP for repairing multiple RAMs in SOCs," in Proc. Int.Symp.VLSI Design, Autom.Test, Apr. 2006, pp.1 4.
11. C.-D. Huang, J.-F. Li, and T.-W. Tseng, "ProTaR: An infrastructure IP for repairing RAMs in system-

- on-chips,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 10, pp. 1135–1143, Oct. 2007.
12. C.-H. Su, R.-F. Huang, and C.-W. Wu, “A processor-based built-in selfrepair design for embedded memories,” in Proc. 12th Asian Test Symp., Nov. 2003, pp. 366–371.
 13. T.-W. Tseng, J.-F. Li, C.-C. Hsu, A. Pao, K. Chiu, and
 14. E. Chen, “A reconfigurable built-in self-repair scheme for multiple repairable RAMs in SOCs,” in Proc. IEEE Int. Test Conf., Oct. 2006, pp. 1–9.
 15. T.-W. Tseng, J.-F. Li, and C.-C. Hsu, “Re-BISR: A reconfigurable built-in self-repair scheme for random access memories in SOCs,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 6, pp. 921–932, Jun. 2010.
 16. T.-W. Tseng and J.-F. Li, “A shared parallel built-in self-repair scheme for random access memories in SOCs,” in Proc. IEEE Int. Test Conf., Oct. 2008, pp. 1–9.
 17. T.-W. Tseng, J.-F. Li, and C.-S. Hou, “A built-in method to repair SoC RAMs in parallel,” IEEE Des. Test Comput., vol. 27, no. 6, pp. 46–57, Nov./Dec. 2010.
 18. S.-K. Lu, Z.-Y. Wang, Y.-M. Tsai, and J.-L. Chen, “Efficient built-in selfrepair techniques for multiple repairable embedded RAMs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 31, no. 4, pp. 620–629, Apr. 2012.
 19. M. Miyazaki, T. Yoneda, and H. Fujiwara, “A memory grouping method for sharing memory BIST logic,” in Proc. Asia South Pacific Design Autom. Conf., Jan. 2006, pp. 671–676.
 20. S.-Y. Kuo and W. K. Fuchs, “Efficient spare allocation for reconfigurable arrays,” IEEE Des. Test Comput., vol. 4, no. 1, pp. 24–31, Feb. 1987.
 21. K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,” IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.
 22. H.-Y. Lin, F.-M. Yeh, and S.-Y. Kuo, “An efficient algorithm for spare allocation problems,” IEEE Trans. Rel., vol. 55, no. 2, pp. 369–378, Jun. 2006.
 23. C.-L. Wey and F. Lombardi, “On the repair of redundant RAM’s,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 6, no. 2, pp. 222–231, Mar. 1987.
 24. W. K. Huang, Y.-N. Shen, and F. Lombardi, “New approaches for the repairs of memories with redundancy by row/column deletion for yield enhancement,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 9, no. 3, pp. 323–328, Mar. 1990.
 25. C. Kothandaraman, S. K. Iyer, and S. S. Iyer, “Electrically programmable fuse (eFUSE) using electromigration in silicides,” IEEE Electron Device Lett., vol. 23, no. 9, pp. 523–525, Sep. 2002.
 26. H. Ito and T. Namekawa, “Pure CMOS one-time programmable memory using gate-ox anti-fuse,” in Proc. IEEE Custom Integr. Circuits Conf., Oct. 2004, pp. 469–472.
 27. R.-F. Huang, J.-F. Li, J.-C. Yeh, and C.-W. Wu, “Raisin: Redundancy analysis algorithm simulation,” IEEE Des. Test Comput., vol. 24, no. 4, pp. 386–396, Jul./Aug. 2007.
 28. M. Tarr, D. Boundreau, and R. Murphy, “Defect analysis system speeds test and repair of redundant memories,” Electronics, vol. 57, pp. 175–179, Jan. 1984.
 29. W. Kang, H. Cho, J. Lee, and S. Kang, “A BIRA for memories with an optimal repair rate using spare memories for area reduction,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 22, no. 11, pp. 2336–2349, Nov. 2014.

Authors profile:



AMGOTH SRINIVAS Received his bachelor’s degree in 2015 in electronics and communication engineering from CMR College of engineering and technology, India which is affiliated with JNTU Hyderabad, India. His areas of interest include VLSI design. He is pursuing his M-Tech in VLSI SYSTEM DESIGN from CMR Institute of Technology.



Dr. Balaji Nehru working as professor in ECE department of CMRIT & Co-Dean of , IRAC . He has an experience of 25 years In Teaching And administrative Fields in various Reputed

Engineering colleges as Assistant professor, Associate professor HOD, Dean, Principal, Of various Reputed engineering colleges. He has published around 20 Research publications in various international journals. He is a life member of ISTE and member of IEEE.



Ms. V Sumathi Received her B.Tech, M. Tech degree in VLSI System Design and Working as Assistant Professor in CMR institute of

technology. And has 3 Years Experience in teaching field and conducting many workshops in college