Anjali Peradath, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 10, Issue 9, (Series-III) September 2020, pp. 06-13

RESEARCH ARTICLE OPEN ACCESS

An Advanced BIRA for Memories using Threshold Technique and Spare Allocation

Anjali Peradath *, Dr. S. Rajkumar **
**(PG Scholar, Department of Electronics and Communication Engineering, Nehru College of Engineering and Research Centre, Pampady, Thrissur, Kerala, India*
*** (Head of Dept., Department of Electronics and Communication Engineering, Nehru College of Engineering and Research Centre, Pampady, Thrissur, Kerala, India*

ABSTRACT
The aim of this paper is to reduce the power consumption and timing with quick fault analysis and spare wire allocation. A common solution which is utilized to guarantee reasonable memory yields is built in redundancy analysis (BIRA). BIRA has two-dimensional spare architecture and is a module that stores and analyzes fault addresses. Numerous different parameters are considered when the performance of BIRA is evaluated such as repair rate, area overhead, and analysis speed. To reduce the area, power consumption and delay a new method of spare allocation using threshold technique is proposed in this paper. It focuses on a 100% repair rate and a minimal area overhead with reduced power consumption and timing. In the fault collection phase, the proposed BIRA identifies the total number and the location of the faults. After the fault collection phase, a redundancy analysis (RA) procedure is performed and the allocation is done with optimized spare wires. By doing so, the proposed BIRA algorithm can repair all repairable faulty memories while maintaining a minimal area overhead and reducing power consumption and delay.
Keywords **-** Analysis speed, Area overhead, BIRA, Delay, Fault addresses, Fault collection phase, Power consumption, Redundancy analysis, Repair rate, Spare allocation, Threshold technique, Two-dimensional spare architecture

---

Date of Submission: 29-08-2020 Date of Acceptance: 14-09-2020

I. INTRODUCTION

The recent rapid growth in the manufacturing of micro-semiconductors has facilitated substantial increases with regards to the memory density and capacity of these devices. As memory density has increased, so too has the chance of faults occurring within these memories. As this can leads to significant yield and reliability issues in semiconductor memories, appropriate solutions need to be found. One promising solution is built-in redundancy analysis (BIRA), which analyzes fault addresses that have been detected and repairs them with spare cells that were incorporated for this purpose. Typically, these redundant cells are incorporated in a two-dimensional spare line architecture comprising row spares and column spares, and the redundancy analysis (RA) for such cells is known to be NP-complete. As such, repair algorithms have to be carefully selected.

The typical performance criteria for BIRAs include repair rate, area overhead, and analysis speed. The repair rate is represented by the percentage of repaired chips in all repairable chips. Many previous BIRA algorithms have looked at achieving a 100% repair rate because this criterion is directly related to the yield of memories. Area overhead is also important because the areas of chips are limited and larger area overhead leads to more expensive chips. Analysis speed, however, has recently been considered to be less important. Previous search-tree-based RA algorithms spent too much time on determining repair solutions because the search mechanisms for determining these solutions were exhaustive. One way of reducing this time was by reducing the analysis speed. However, recent BIRA approaches have used content addressable memories (CAMs) to achieve significant improvements in analysis speeds. As a result, most of the test and repair times of built-in self-repair (BISR) are taken up by the running of built-in self-test (BIST).

In the existing system, it focuses on dividing the faults into 2 categories: spare pivot fault and non-spare pivot fault. Here the fault collection and RA procedure takes a lot of time and hence the area and power consumption is very high. It becomes difficult to implement for large memories. Hence to remove the drawbacks of the exiting BIRA approach a new technique has been introduced.

In order to enhance the area overhead and achieve a 100% repair rate, a BIRA approach with Threshold technique has been proposed. In order to minimize the area overhead, delay and power consumption a threshold based technique has been introduced. In the Threshold technique, a threshold limit is kept. All the rows having faults greater than the threshold will get replaced parallely. The Fault collection phase followed by the RA procedure becomes quick. The faulty memory addresses are quickly replaced with spare wires hence correcting the memory.

II. LITERATURE REVIEW

Lee et.al [1] proposed an MBISR generator called BRAINS+, which automatically generates register transfer level MBISR circuits for SoC designers. The MBISR circuit enhances the essential spare pivoting algorithm, with a more flexible spare architecture which is based on a redundancy analysis (RA) algorithm, which can fit failure patterns more efficiently by configuring the same spare to a row, a column, or a rectangle .The MBISR can easily be applied to multiple memories with a distributed RA scheme, with its zero test-time penalty and low area overhead. **Habiby et.al** [2] has implemented BIRA by flipping-analyzers breaks down the analysis process into two phases without any complicated FSM to load different solutions to BIRA. This method achieves low area overhead in memories and a short analysis time with symmetric redundancy configuration. Compared with other parallel BIRAs, proposed method can save 50% of area overhead. Also it is faster than Intelligent Solve First and ESP methods. **Kang et.al** [3] explores the BIRA that minimizes area overhead by utilizing a part of the spare memory as an address mapping table (AMT). The reduced addresses produced by the AMT are used to reduce the extra hardware overhead and are stored in content-addressable memories (CAMs) and used in the RA procedure. By using an exhaustive search RA algorithm, the proposed BIRA can achieve an optimal repair rate. The experimental results show that the previous state-of-the-art BIRA with an optimal repair rate requires a larger area overhead than that of the proposed BIRA. **Oh et.al** [4] discussed that in order to improve area overhead and analysis speed, we propose a BIRA architecture and RA (redundancy analysis) algorithm using a fault-free region in embedded memory instead of deploying extra CAMs. The proposed RA algorithm stores faulty cell addresses in the fault free region and achieves a very high repair rate and brings a significant area reduction. By using its own memory instead of shared CAMs,it allows a parallel memory test. **Kim et.al** [5] investigated that the proposed BIRA uses various types of spares and can achieve a higher yield than a simple row and column spare structure. Herein, a BIRA that can achieve an optimal repair rate using various spare types is proposed. This analyzer can exhaustively search global and local spare types. A fault-storing content-addressable memory (CAM) structure that is small and collects faults efficiently is also proposed in this paper. The experimental results shows a short analysis time and a high repair rate with a small hardware overhead.

III. BACKGROUND WORK

3.1 Background - Spare Pivot Faults

All faults are classified into two groups: spare pivot faults and non-spare pivot faults. During the fault collection phase, if a newly detected fault has a unique row address and column address compared to the collected faults, then the fault becomes a spare pivot fault. Therefore, spare pivot faults do not share row and column addresses with each other. All faults except spare pivot faults are defined as non-spare pivot faults. After a spare pivot set is decided, every non-spare pivot fault should share its row or column address with at least one of the spare pivot fault.

Figure 1 shows an example of a faulty memory with two row spares and two column spares. In this example, there are eight faults into an 8-by-8 memory array. A fault detection order is shown in as a number in the top left corner of each fault. At the end of the fault collection phase, the first, second, fifth, and sixth detected faults make the set of spare pivot faults, {(2, 1), (5, 3), (1, 4), (7, 7)}, which is depicted as red-highlighted faults.

Because each of these four faults has unique row and column addresses, and every non-spare pivot faults shares a row or column address with one or more of the faults in the set, this set of faults meets the conditions of the spare pivot set.

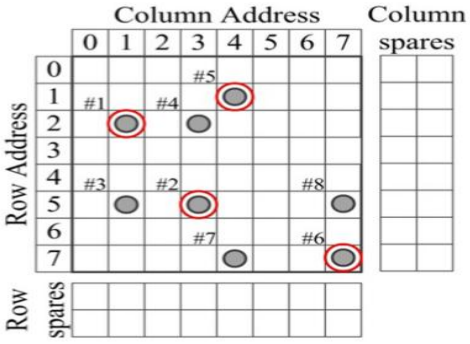

Figure 1: Example of the set of spare pivot faults.

3.2 Motivation

The most critical concern about spare pivot faults is that, if the number of spare pivot faults is

greater than the number of total spares, then the faulty memory cannot be repaired. At least one spare line should be allocated to each spare pivot fault, because of the uniqueness of the row and column addresses. As a result, the maximum number of spare pivot faults is the same as the number of total spares.

Let the numbers of row spares and column spares be represented by R_s and C_s, respectively. The maximum number of spare pivot faults is then R_s+C_s, and the maximum number of non-spare pivot faults is (C_s-1) +$C_s(R_s-1)$. The importance of spare pivot faults is much greater than non-spare pivot faults in RA procedures.

3.3 Fault Collection

The proposed fault collection logic stores all faulty information. Figure 2 depicts the proposed CAM structure, which can be classified into three areas: spare pivot area, a cross-fault area, and a solution area.

The spare pivot area collects raw data from the spare pivot addresses, while the BIST tests the memory arrays.

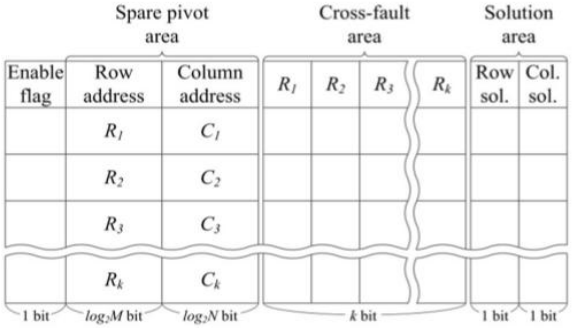

Figure 2: Proposed fault-storing CAM structure.

A certain type of non-spare pivot fault disturbs the repair decision by simultaneously sharing its row address with one spare pivot fault and its column address with another spare pivot fault; this is defined as a cross-fault. For a 100% repair rate, the information of cross-faults should be highlighted, because it affects the repair decisions of two different spare pivot faults.

3.4 Redundancy Analysis

After the fault collection process, the proposed BIRA algorithm determines which addresses can be repair solutions. In order to obtain a 100% repair rate, an exhaustive search algorithm is adopted using the information of the fault-storing CAM. For the sake of convenience, non-spare pivot faults that are not cross-faults are defined as tail-faults in this paper. The goal of the exhaustive search is to cover all of the 1 bit in the cross-fault area. The first step in this process is choosing C_a column addresses in the spare pivot area, and the next step is checking all of the covered parts in the cross-fault area from choosing C_a column addresses and repair decisions of the solution area. If the uncovered parts in the cross-fault area can be covered by R_a row spares, then the faulty memory can be repaired. The final step is to store the repair solution into the solution area.

IV. PROPOSED BIRA APPROACH

4.1 Scheme of the Memory BISR

This BISR consisted of a BIST and a BIRA. The BIST generates test patterns that allow for faulty cells within a memory array to be detected. When a faulty cell is detected, the BIST sends the fault address to the BIRA. The fault address is then stored within the fault collection logic of the BIRA. If the BIRA uses CAMs for the fault collection, the incoming fault address can be compared with the stored fault addresses within one clock cycle. After this test process, the BIRA algorithm performs a RA procedure based on the information collected about the faults. If a repair solution is possible, the BIRA algorithm provides it and updates the RA status, which indicates whether the faulty memory can be repaired. The block diagram of a memory BISR is shown in figure 3.

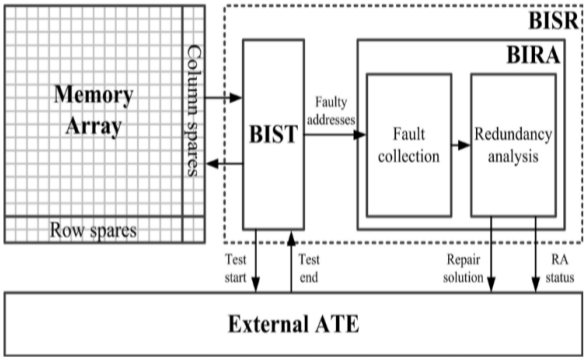

Figure 3: Block diagram of a memory BISR.

4.2 Spare Wire allocation using Threshold Technique

The figure 4 shows the block diagram of the spare wire allocation using Threshold technique. There are three blocks: The LFSR, Memory and the Self-Repair block.

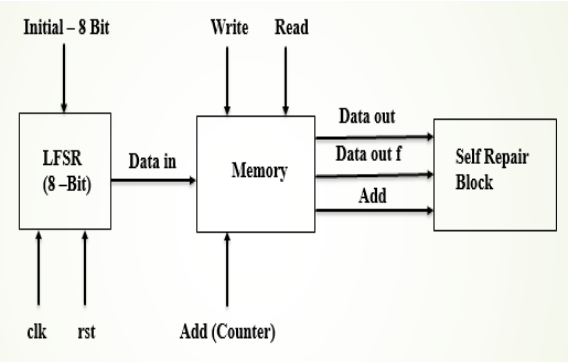

Figure 4: Block diagram of Spare Wire allocation using Threshold Technique.

The test pattern is generated from the 8- bit LFSR. The LFSR output which is "Data in" is the input to the memory which is the circuit under test. Memory can either Read or Write. When "Write" is given, the original data gets stored and when "Read" is given the injected fault is seen in the memory. The original data is represented as "Data out" and faulty data is represented as "Data out f". While testing, the original data will be stored in the memory. While reading the data, the original data, faulty data and corresponding address of both the data are obtained. Comparison is done in the Response Analyzer and the faulty address is obtained. Then using Threshold technique the allocation is done with optimized spare wires.

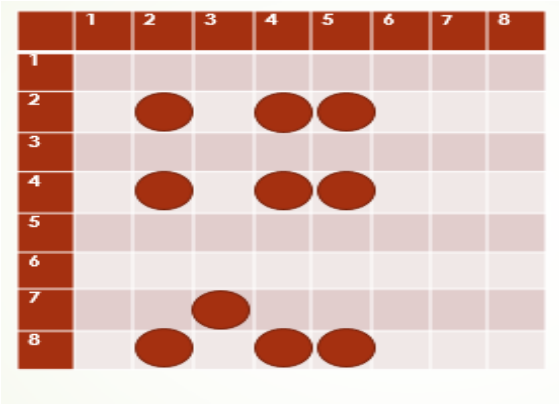

Figure 5: Example of spare wire allocation using threshold technique.

Figure 5 shows an example of a faulty memory. In this example, there are ten faults into an 8-by-8 memory array. In the second, fourth and eighth row, the 2^{nd},4^{th} and 5^{th} bits are faulty and in the seventh row the 3^{rd} bit is faulty. Once the number of faults and the faulty locations are found they have to be replaced with spare wires. In the Threshold technique, a threshold limit is kept. In this example the threshold is kept as 2. All the rows having faults greater than 2 will get replaced parallelly. That is the second, fourth and eighth row will get replaced. The seventh row will still be left out as it has only 1 fault which is less than the threshold. Now the spare column is allocated for column 3. Hence all the faulty addresses are corrected using the threshold based approach resulting in reduced area, delay and power consumption.

V. RESULT AND DISCUSSION

5.1 PROPOSED SYSTEM

5.1.1 Area

The Figure 6 represents the Area (166 total equivalent gate count for design) obtained from Xilinx Software for proposed system.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	8	13,824	1%	
Number of 4 input LUTs	17	13,824	1%	
Logic Distribution				
Number of occupied Slices	16	6,912	1%	
Number of Slices containing only related logic	16	16	100%	
Number of Slices containing unrelated logic	0	16	0%	
Total Number of 4 input LUTs	17	13,824	1%	
Number of bonded IOBs	249	510	48%	
Number of GCLKs	1	4	25%	
Number of GCLKIOBs	1	4	25%	
Total equivalent gate count for design	166			
Additional JTAG gate count for IOBs	12,000			

Figure 6: Area of proposed system.

5.1.2 Delay
The Figure 7 represents the Delay (3.140 ns) obtained from Xilinx Software for proposed system.

```
Timing Summary:
---------------
Speed Grade: -7

   Minimum period: 3.140ns (Maximum Frequency: 318.471MHz)
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: 6.500ns
   Maximum combinational path delay: No path found
```

Figure 7: Delay of proposed system.

5.1.3 Power
The Figure 8 represents the Power (122 mW) obtained from Xilinx Software for proposed system.

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		122
Vccint 1.80V:	64	115
Vcco33 3.30V:	2	7
Clocks:	41	73
Inputs:	8	15
Logic:	0	0
Outputs:		
Vcco33	0	0
Signals:	0	0
Quiescent Vccint 1.80V:	15	27
Quiescent Vcco33 3.30V:	2	7

Figure 8: Power of proposed system.

5.1.4 Output Waveform
The Figure 9 represents the output waveform obtained from ModelSim Software for proposed system.

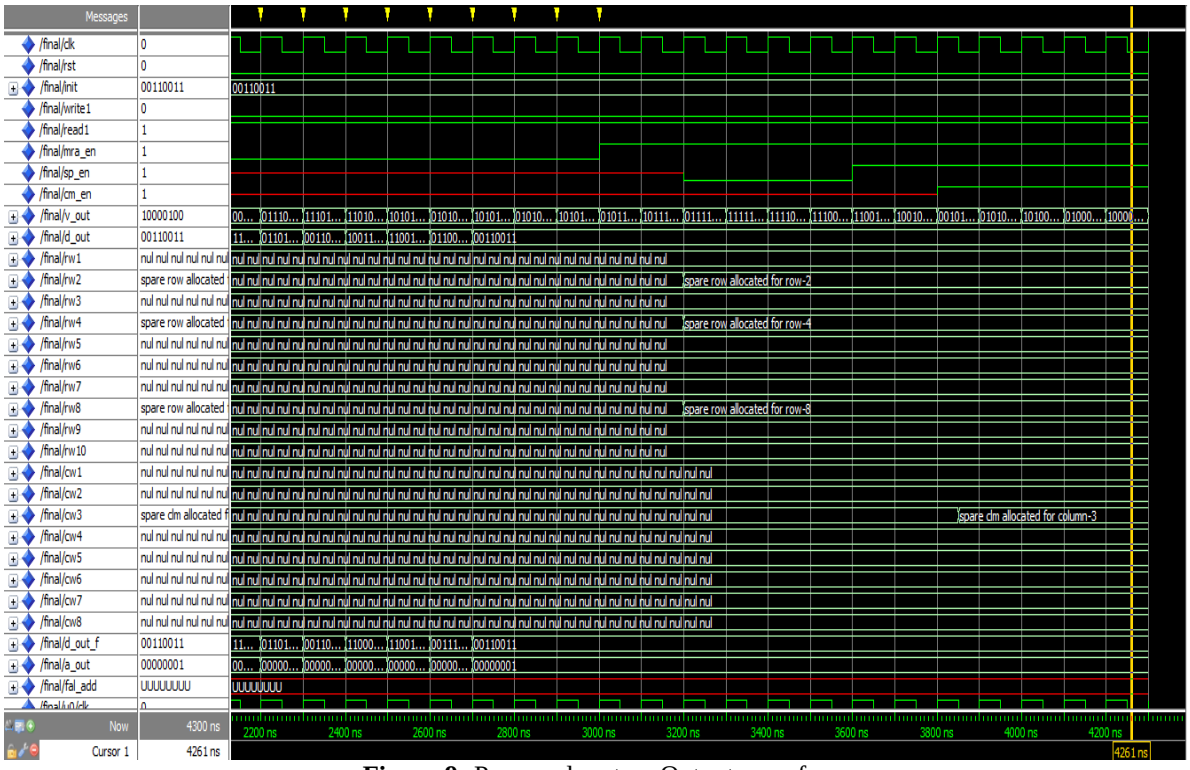

Figure 9: Proposed system Output waveform

5.2 Performance Analysis

Table 1 shows the comparison between the Area, Delay and Power of the existing and proposed models.

Table 1: Comparison table of the area, delay and power of the existing and proposed model.

BIRA Models	AREA (Total Equivalent gate count for design)	DELAY (in ns)	POWER (mW)
Existing System	1,549	8.389	147
Proposed System	166	3.140	122

Comparison of the area for the existing and proposed system are shown in figure 10. The area of the existing system is 1,549 and proposed system is 166 total equivalent gate count respectively.

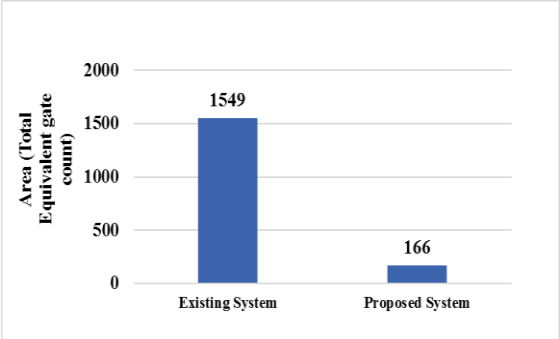

Figure 10: Area results.

Comparison of the delay for the existing and proposed system are shown in figure 11. The delay of the existing system is 8.389 ns and proposed system is 3.140 ns respectively.

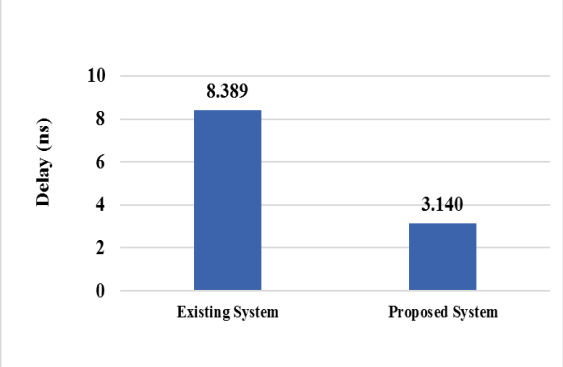

Figure 11: Delay results.

Comparison of the power consumption for the existing and proposed system are shown in figure 12. The power consumption of the existing system is 147 mW and proposed system is 122 mW respectively.

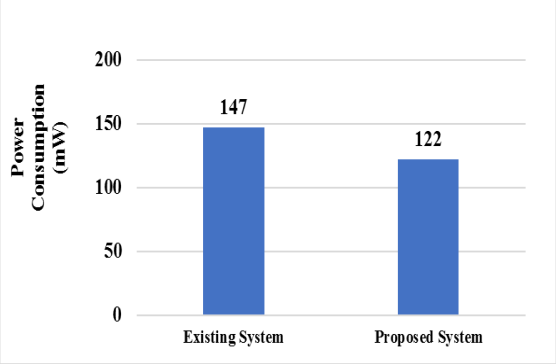

Figure 12: Power consumption results.

VI. CONCLUSION

This paper carries out a detailed investigation of the faults, and it proposes an efficient BIRA approach using the characteristics of these faults. The proposed BIRA algorithm using threshold technique can acquire the smallest area overhead and can have a 100% repair rate. The experimental results from both the circuit and application levels demonstrate that the proposed system delivers greater improvements in energy saving, design area and reducing delay. From this paper it can be concluded that the proposed BIRA approach using threshold technique and spare wire allocation has the smallest area (166 Total Equivalent gate count for design), delay (3.140 ns) and power consumption (122 mW) compared with the existing method. In conclusion, it is expected that the proposed BIRA approach will prove to be a practical solution to the yield and reliability problems commonly encountered in commodity memories.

REFERENCES

[1]. M. Lee, L.-M. Denq, and C.-W. Wu, "A memory built-in self-repair scheme based on configurable spares," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 30, no. 6,* pp. 919-929, Jun. 2011.

[2]. P. Habiby and R. Niarakiasli, "An improved BIRA for memories with optimal repair rate using a flipping analyzer," *in Proc. ICEE, May 2012*, pp. 188-193.

[3]. 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 Integr. Syst., vol. 22, no. 11,* pp. 2336-2349, Nov. 2014.

[4]. C.-H. Oh, S.-E. Kim, and J.-S. Yang, "A BIRA using fault-free memory region for area reduction," *in Proc. APCCAS, Oct. 2016,* pp. 480-482.

[5]. J.Kim, W. Lee, K. Cho, and S. Kang, "Hardware-efficient built-in redundancy analysis for memory with various spares," *IEEE Trans. Very Large Scale Integr. Syst., vol. 25, no. 3,* pp. 844-856, Mar. 2017.

[6]. 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.

[7]. J.-F. Li, J.-C. Yeh, R.-F. Huang, and C. –W. Wu, "A built-in self-repair design for RAMs with 2-D redundancy*," IEEE Trans. Very Large Scale Integr. Syst., vol. 13, no. 6,* pp. 742-745, Jun. 2005.

[8]. S.-K. Lu, Y.-C. Tsai, C.-H. Hsu, K.-H. Wang, and C.-W. Wu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," *IEEE Trans. Very Large Scale Integr. Syst., vol. 14, no. 1,* pp. 34-42, Jan. 2006.

[9]. K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architecture: A tutorial and survey*," IEEE J. Solid-State Circuits, vol. 41, no. 3,* pp. 712-727, Mar. 2006.

[10]. 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. Syst., vol. 17, no. 12,* pp. 1665-1678, Dec. 2009.

[11]. T.-W. Tseng, J.-F. Li, and C.-C. Hsu, "ReBISR: A reconfigurable built-in self-repair scheme for random access memories in SOCs," *IEEE Trans. Very Large Scale Integr. Syst., vol. 18, no. 6,* pp. 921–932, Jun. 2010.

[12]. 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 using a branch analyzer*," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 29, no. 12,* pp. 2014-2026, Dec. 2010.

[13]. T.-J. Chen, J.-F. Li, and T.-W. Tseng, "Cost-efficient built-in redundancy analysis with optimal repair rate for RAMs," *IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 31, no. 6,* pp. 930–940, Jun. 2012.

[14]. B.-Y. Lin, M. Lee, and C.-W. Wu, "Exploration methodology for 3D memory redundancy architectures under redundancy

constraints," *in Proc. ATS, Nov. 2013,* pp. 1-6.

[15]. K. Cho, W. Kang, H. Cho, C. Lee, and S. Kang, "A survey of repair analysis algorithms for memories," *ACM Comput. Surveys (CSUR), vol. 49, no. 3,* pp. 1-41, Oct. 2016.

Anjali Peradath, et. al. "An Advanced BIRA for Memories using Threshold Technique and Spare Allocation." *International Journal of Engineering Research and Applications (IJERA)*, vol.10 (09), 2020, pp 06-13.