International Journal for Innovative Engineering and Management Research

A Peer Revieved Open Access International Journal

www.ijiemr.org

IJIEMR Transactions, online available on 24th Apr 2020. Link :http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-04

Title: **BUILT IN REDUNDANCY PROCEDURES FOR MEMORY REPAIR**

Volume 09, Issue 04, Pages: 67-78

Paper Authors

V.R.SESHAGIRI RAO, DR.ASHA RANI.M

BUILT IN REDUNDANCY PROCEDURES FOR MEMORY REPAIR

V.R.SESHAGIRI RAO, DR.ASHA RANI.M
[1]ECE Department, Institute of Aernautical Engineering Dundigal Hyderabad, Telangana, India 500043
[2]Professor, ECE Department, JNTUH College of Engineering Hyderabad, Telangana, India, 500 085
[1]vadrevu.vr@gmail.com,[2]ashajntu1@yahoo.com

ABSTRACT

With the latest developments in VLSI technology, the size of memories is rapidly growing. The yield criteria and testing problems have become the most critical areas for memory manufacturing. Traditionally, redundancies are applied so that the faulty cells can be repairable. Using external memory testers redundancy analysis is becoming slow as the chip density continues to grow, especially for the system chip with big embedded memories. This paper presents three redundancy allocation algorithms which can be implemented on-chip. Out of three algorithms, two are based on the local-bitmap concept: the local repair-most approach is good for a general spare configuration, and the local optimization approach has the better repair rate. The essential spare pivoting technique is proposed to simplify control circuitry.

Index Terms— **built-in self-test, DRAM , Built-in self-diagnosis, memory testing ,embedded memory, ,redundancy analysis, SRAM, yield improvement**

ACRONYMS

BISD	built-in self-diagnosis
BIST	built-in self-test
BIRA	built-in redundancy-analysis
BISA	built-in self-analysis
RM	repair-most
FC	fault collection
SA	spare allocation
LRM	local repair-most
LO	local optimization
ESP	essential spare pivoting

NOTATION

M	number of rows of memory array
N	number of columns of memory array
l	fault line of a row or column
F	total number of faulty cells

l

F_l	number of faulty cell in a line
Fo	number of orthogonal faulty cells
m	number of rows of local bitmap
n	number of columns of local bitmap
r_a	available spare rows
r_c	available spare columns
n_r	number of faulty rows not covered
n_c	number of faulty columns not covered

1. INTRODUCTION

MOS memories are microelectronic components they occupy about 30% of the world semiconductor market [1-4]. There is continuous innovation in VLSI technology which keeps increasing the capacity and density of semiconductor memories, integrating more and more cells in a silicon chip; as per Moore's law the capacity of memory chips roughly doubles every 18months[5-7]. As the feature size shrinks, increasing the integration density and chip size, keeping a acceptable yield level is very hard. To improve yield, spare rows and columns of storage cells) are often added so that most faulty cells can be replaced by spare cells [8–13]. Redundancy, however, adds to cost due to area overhead. Therefore, analysis of redundancies to improve yield (after repair) and minimize cost is a vital process during manufacturing. Redundancy analysis using expensive memory testers is not cost-effective as the chip density continues to grow. The use of embedded memories creates yet another problem; embedded memories are even difficult to deal with using external testers [14-19]. There is a prediction that memories will occupy more than 90% of the system chip area within a decade while the chip complexity keeps increasing dramatically It has been widely noted that embedded memories will play an increasingly important role in the semiconductor market in the coming years, because the system-on-chip market is on demand and almost every system chip contains some type of embedded memory [20-23].

Testing embedded memory is more difficult than testing commodity memory in general, unless built-in self-test (BIST) is used [24-27]. Several BIRA/BISR techniques have been proposed in the earlier, e.g., [28-30]. In a DRAM with BIST and BISR circuits has been proposed, using SRAM cells as the redundancy. The preference is on the power-on self-test and self-repairs at the memory board level. In [31-32], a hierarchical BISR structure has been proposed to facilitate memory test, fault location, and self repair. In a microprocessor-based BISD circuit was presented for repairable embedded SRAM with programmable modules. An embedded high-density SRAM with BISR capability has been proposed [33-35], allowing autonomous repair of the SRAM core with negligible timing penalty. The spare architecture was simplified (it has only spare columns), so a simple greedy algorithm can be used for diagnosis and self-repair [36-37]. Some segment-based redundancy architectures have been proposed to improve the repair efficiency with simplicity of the analysis algorithms [38]. Most of the works either assume simplified redundancy architecture, i.e., they assume only spare rows or spare columns are available.

In this paper, we propose three BIRA algorithms. It will be shown that at, by our approaches, high repair rate and low area overhead can be achieved. The BIRA (or Built-In Self-Analysis (BISA) [39]) techniques are flexible and can cooperate both with the external tester and BIST. In addition, when cooperating with BIST, redundancy analysis can be done concurrently when testing, so there is little time penalty for the analysis.

2. CONVENTIONAL REDUNDANCY ANALYSIS ALGORITHMS

Before the discussion of more efficient algorithms, the terminology and notation are defined first. A memory *block* consists of rows and columns of storage cells, i.e. The origin of the array is the upper left corner. There are spare rows and spare columns. The following definitions are followed.

Definition 1: A faulty line is said to be covered if all faulty cells on the line have been planned to be repaired by specific spare rows and/or spare columns.

Definition 2: A faulty line is either a row or a column on which one or more faulty cells exist. The number of faulty cells in l is F_l . A faulty line can be a either a faulty row or a faulty column.

Definition 3: A faulty cell which does not any common row or column with any other faulty cell is called as an orthogonal faulty cell.

Let the number of available spare rows and available spare columns during the analysis process be denoted as r_a and c_a respectively. During BIRA, any faulty line which consists of *k* faulty cells requires either 1 spare line in the same direction or *k* perpendicular spare lines.

Definition 4: Must repair faulty line is defined as any faulty row (column) with *k* faulty cells (i.e.,$F_{l\,=}$ k), where, k > c_a (k > r_a) .

Aavide 1: F_l can be limited to >=c+1(r+ 1) for the must-repair faulty row (column) l.

In case there are c+1 or more faulty cells in the faulty row, then it is a must-repair row (see Definition 4). And Aavide 1 indicates that a fault counter which counts up to c+1 is enough for the must-repair analysis.

Actually, redundancy analysis is done on a memory tester using software. The tester stores the bitmap (a map of the faulty cells) after a diagnostic test, and performs redundancy analysis based on the bitmap. The algorithms consist of two phases: the *must-repair* phase followed by the *final-repair* phase. In the must-repair phase, all the

must-repair faulty lines are identified first by counting the number of faulty cells F_l for each faulty line, limiting the number of remaining faulty cells. With Aavide 1, the storage to record F_l for each faulty line can be restricted effectively. In the final-repair
phase, simple algorithms, such as the *repair-most* algorithm or *fault-driven* algorithm, are used. The repair-most approach calculates F_l of the remaining faulty lines and selects the faulty line with the largest F_l for the repair one by one;

X	X	X	X								
				X	X	X	X				
								X			
								X			
									X		
									X		
										X	
										X	
											X
											X

Fig 1 :Defective Memory of Worst case Here $r_a = 2$ and $c_a = 4$

Let F denote the total number of faulty cells in the memory block, and Fo represents the number of orthogonal faulty cells.

Two important early termination conditions for redundancy analysis are as follows.

Condition 1: After the must-repair phase, $F > 2\, r_a c_a$

Condition 2: $Fo > r_a + c_a$

If any one of the condition is met, then the analysis process stopped. Early termination conditions help to identify the memory which cannot be repaired by available spares. As per Aavide 1, we have the following two additional early termination conditions.

Condition 3: if $n_c > r_c$ if $r_a = 0$ where n_c is the number of faulty columns not covered so far

Condition 4: if $n_r > r_a$ if $c_a = 0$ where n_r is the number of faulty rows not covered so far.

If any of the above four conditions is true, then the memory block is unrepairable.

A defective memory block of the bad case is shown in Fig. 1, where $F=2r_ac_a$ **after the must-repair phase**, and all the available spare rows and columns have to be used for repairing the faulty lines. Especially, the available spare rows cover r_a faulty rows, where each faulty row has c_a faulty cells and among these faulty rows, no cells share the same column address. Therefore, the faulty cells in these rows are located in

r_a x c_a different columns. Similarly, the c_a faulty lines covered by the available spare columns have r_a x c_a row addresses for the faulty cells. In this special case, the width of the bitmap is $c_a r_a + c_a$ and the height is r

$_a$ c $_a$ + r $_a$. Therefore, in general, the size of the bitmap can be limited to (r $_a$ c $_a$ + r $_a$) X (c $_a$ r $_a$ + c $_a$) instead of M X N after the must-repair phase. Without the must-repair phase, the maximum size of the bitmap can be limited to (r(c+1)+r) X (c(r+1)+c) due to Aavide 1.

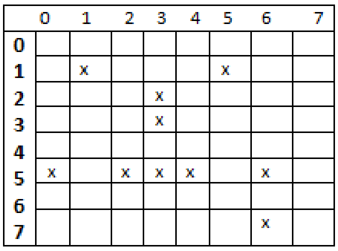
3. NEW BUILT-IN REDUNDANCY ANALYSIS APPROACHES

A. Repair-Most Using Local Bitmap

Having the full bitmap on-chip for the purpose of redundancy analysis obviously is not feasible. Our goal is for the BIRA circuit to properly allocate redundancies in parallel with the BIST operation. Two requirements of BIRA circuit are i) optimum repair rate ii) low area overhead.

The *repair rate* of the redundancy analysis is defined as the ratio of good memories after the redundancy analysis and all the faulty memories. *repair efficiency* is defined as the repair rate with respect to unit area overhead. The area overhead should be low and the repair efficiency should be high. The proposed algorithm require only a small array of size m x n for the storage the *local bitmap*. This reduces the silicon area overhead for the bitmap and the row/column counters. The parameters, m and n, can be constant or proportional to r and c, respectively, depending on the defect cluster distribution. We will show later that the local-bitmap technique results in a fast and small BIRA circuit with good repair efficiency and the BIRA circuit can be easily interfaced with the BIST circuit.

To illustrate LRM algorithm, consider the memory block as shown in Fig. 4, which is 8X8 size where r = c = 2(spare rows and spare columns). Also, it is assumed to perform a row-wise march algorithm during BIST. The physical cell location is also provided to facilitate the analysis. The LRM analysis process for the memory block is shown in Fig. 5. After the first five faulty cells have been detected, the bitmap is as shown in Fig. 5(a). The sixth faulty cell (i.e., cell(5,2)) has the same row address as the fourth row address tag of the current bitmap, but a column address different from any existing one. Since the bitmap is full Row 1 is selected (shaded in the figure) for repair at this stage, resulting in the bitmap as shown in Fig. 5(b). The next 3 faulty cells



	0	1	2	3	4	5	6	7
0								
1		x				x		
2				x				
3				x				
4								
5	x		x	x	x		x	
6								
7							x	

Fig 4 A Memory Block with defects

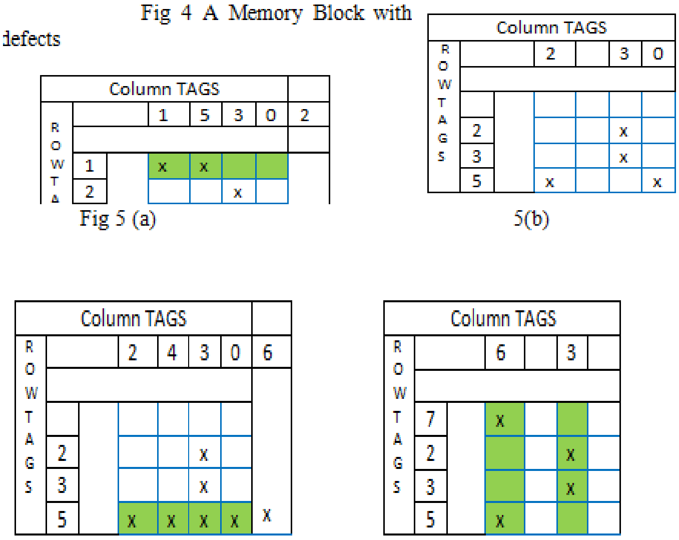


Fig 5 (a)

5(b)

5(c)

5(d)

(i.e., cell(5,4), cell(5,3), and cell(5,0)) are all located in the same row (i.e., Row 5) as shown in Fig. 5(c), so Row 5 is selected for repair next. Finally, after all faulty cells are processed, Column 6 and Column 3 are selected in sequence for repair according to their weights and the type of remaining spare lines, as shown in Fig. 5(c). In summary, Row 1, Row 5, Column 6, and Column 3 are replaced by spare lines after the repair process.

To search for the faulty line with the largest fault count, two heuristic rules are applied:

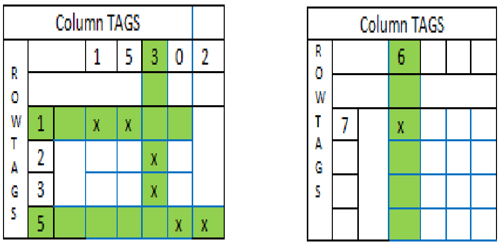
1) The address of the newly detected faulty cell is used together with the current bitmap to help determine the line with the largest fault count. It is possible that the row address is previously recorded in the bitmap but there is no room for the column address, or vice versa.

2) Any row (column) with a fault count exceeding c_a (r_a) is regarded as a must-repair faulty row (column). Additionally, Condition 3 or 4 can be applied if $r_a = 0$ or $c_a = 0$. Condition 1 can be used for early termination if there is a fault counter for keeping the total fault number and the fault count from the must-repair faulty lines can be dynamically subtracted from the total fault count. Early termination by Condition 2 can be applied with additional address registers for the orthogonal faults.

B. Local Optimization

In LRM, selecting the faulty line with the largest fault count takes more time, because it may be required to pickup more than one line for repair (the number of lines to be repaired is a variable). Also , the repair rate of LRM is constrained by the bitmap size. To deal with this problem, another algorithm is proposed, i.e., the *Local Optimization* (LO) algorithm. The LO algorithm has a better repair rate, though it also uses the local bitmap. It searches for all possible ways of spare allocation when the bitmap is full.

Consider the same memory block as shown in Fig. 4 again. The bitmap construction procedure is the same with LRM for the first 6 faults, as shown in Fig. 6(a). when the bitmap overflows, intensive search is carried out to cover all faults recorded in the bitmap. Subsequently, Row 1, Row 5, and Column 3 are selected upon the detection of the sixth faulty cell, cell(5,2). Assume that after this search, all the remaining faults are covered except the last fault from cell(7,6). As shown in Fig. 8(b), the corresponding column, i.e., Column 6, is selected as there is only one spare column is left. The LO algorithm usually requires a more execution time than LRM for large memory sizes, as exhaustive search is performed. However, for an *unequal* spare architecture (i.e. r #= c), LO can take advantage of it by counting all combinations of along the direction with less spare lines. also the number of total iterations is reduced in this algorithm.



6(a) 6(b)

The rules and regulations of LRM still apply to LO as discussed previously. In addition, a rule which deals with orthogonal faults, based on Condition 2, can be used to improve the repair rate. The heuristic is as follows. Every incoming fault which does not match any of the row and column address tags in the local bitmap is stored in an additional *orthogonal fault register* (OFR). When a fault is detected, it is required to perform an additional check of the fault with those in the OFRs. If there is a row or column coincidence then both the matched and matching addresses are recorded in the bitmap and the one in the OFR cleared. Orthogonal faults in the OFRs are considered first after all other faults are covered. The remaining spare lines are selected to cover those orthogonal faults, one by one, using any order. This heuristics obviously complicates the control, but improves the repair rate to a great extent. Condition 2 can be applied together with this heuristics.

C. Essential Spare Pivoting

So far the above algorithms use a small bitmap instead of the full bitmap. As most algorithms depend on a bitmap as the tool for redundancy analysis, it may appear that that a bitmap is inevitable. This is not true. In this section we the *essential spare pivoting* (ESP) algorithm is proposed without using a bitmap. It will be shown later that the repair rate using ESP is still high, with a greatly simple implementation and reduced area overhead.

So the following general guidelines are proposed for redundancy analysis.

1) For any faulty row (column), if the number of faulty cells is greater than or equivalent to a threshold number E_{th}, repair it by a spare row (column).

2) An orthogonal fault can be repaired by either a spare row or a spare column. Orthogonal faults should be processed after all others.

This guideline is similar to the must-repair rule, except that the decision is based on a customized threshold number E_{th}, instead of r_a and c_a . In the analysis procedure, we maintain a counter for the number of faults in each faulty line. When the number reaches E_{th}, it is marked as an *essential line*. Assume E_{th} to be 2, then the threshold comparison is greatly simplified. The second guideline shown above states that an orthogonal fault should be recognized early but processed after all other. The reason is that, e.g., while $c_a > 0$ and $r_a > 0$, if we repair an orthogonal fault by a spare row before repairing other non-orthogonal faults, we may lose the chance to repair more faults with this spare row, as orthogonal fault can also be repaired by a spare column. With this guideline and the proposed guideline, a new algorithm called the *essential spare pivoting* (ESP)

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

algorithm, is developed as shown in Fig. 7.

The faulty-cell addresses are collected and stored in the P_R (*row pivot*) and P_C (*column pivot*) register files. Both have r + c registers, and all the registers are initially empty. An incoming faulty-cell address (R^, C^) is compared with the existing row pivots and column pivots in the register files. If there is a row-address match or column-address match, the matched pivot is marked as an *essential pivot* (EP). During the Fault collection phase, if the number of the pivot pairs exceeds r+c, this memory is unrepairable and the process terminates. If there is no match, the row address and column address of the current faulty cell are stored in the P_R and P_C registers, respectively. The repair rate is high when $E_{th} = 2$, and only a flag is needed along with each pivot to indicate whether it is an EP.

In the Fault allocation phase, spares are distributed according the contents of the P_R and P_C registers. It consists of two stages. In the first stage, spare rows are allotted for the essential row pivots and spare columns for the essential column pivots. After the first stage, the pivot registers contain all and only the addresses of the orthogonal faults, because they have never matched other faulty-cell addresses. These can repair these faults by either spare rows or spare columns. In ESP_SA(), we simply allocate available spare rows before spare columns.

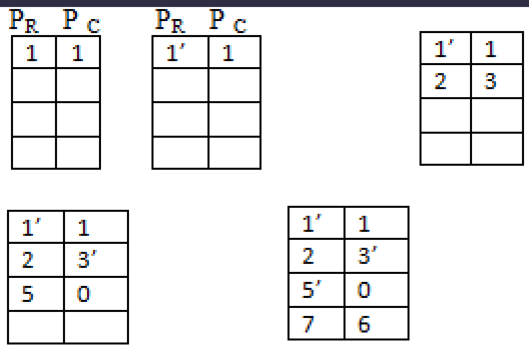


FIG 7

Now the ESP algorithm is explained by an example. The memory block under test is the same as shown in Fig. 4. The faulty cells detected are, in sequence, cell(1,1), cell(1,5), cell(2,3), cell(3,3), cell(5,1), cell(5,0), cell(5,2), cell(5,3), cell(5,4) (5,6), and cell(7,6). The FC procedure is illustrated in Fig. 7. In the figure, the and registers are shown as the left and right columns of the register array, respectively. For each faulty-cell address, the is stored in the left column and the is stored in the right column. There is a ‘ mark on a pivot if it is an essential pivot. To begin, the register array is empty, so the first address (1,1) is stored in the first row of the array directly. The second address (1,5) matches (1,1) in the row address, so the of cell(1,1) is marked as an EP. Similarly, the address (2,3) is inserted directly, while the address (3,3) matches (2,3) in its column address, thus the of cell(2,3) is marked as an EP. This procedure continues until the address (7,6) is recorded. The SA procedure is simple: first spares are allocated for the EPs—Row 1, Row 5, and Column 4, then a spare column is allocated to repair the orthogonal fault on cell (7,3). The major advantage of the ESP is mainly its simple in implementation, which results in smaller area overhead than other algorithms. The revised first guideline provides a simple search method for orthogonal faults. In the SA stage of the ESP algorithm, orthogonal faults and non orthogonal faults can be easily separated by checking their EP flags. The automatic recognition for orthogonal faults greatly increases the repair efficiency. These features make the ESP algorithm small, fast, and easily implementable.

4. SUMMARY AND CONCLUSIONS

THREE algorithms suitable for built-in redundancy analysis have been proposed. The algorithms are simple compared to conventional analysis algorithms on the memory tester. Among them, two are based on the local-bitmap idea, i.e., the LRM and LO algorithms. Our analysis shows that the LRM algorithm is highly scalable for general memory and redundancy architectures, and the LO algorithm optimizes spare allocation in the local bitmap, resulting in the best repair rate. Also, the time overhead of LO is low for a practical spare architecture. The third approach, i.e., the ESP algorithm, does not require a bitmap. It greatly simplifies the control circuit, and results in the lowest time and area overhead among the three BIRA schemes. It was shown to achieve a high repair rate for a mature fabrication process with small area overhead. The estimated area overhead of an individual LRM circuitry is below 0.3% for a 64Mbit embedded DRAM core.

REFERENCES

[1].Lee, H., Kim, J., Cho, K., & Kang, S. (2018). Fast Built-In Redundancy Analysis Based on Sequential Spare Line Allocation. *IEEE Transactions on Reliability*.

[2].Patnaik, S., & Ravi, V. (2018). A Built-in Self-Repair Architecture for Random Access Memories. In *Nanoelectronic Materials and Devices* (pp. 133-146). Springer, Singapore.

[3].Gebregiorgis, A., Bishnoi, R., & Tahoori, M. B. (2018). A Comprehensive Reliability Analysis Framework for NTC Caches: A System to Device Approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[4]. Huang, C. T., Wu, C. F., Li, J. F., & Wu, C. W. (2003). Built-in redundancy analysis for memory yield improvement. *IEEE transactions on Reliability*, *52*(4), 386-399.

[4]. M. Lin, Ed., “1997 Semiconductor Industry Annual Report,” (in Chinese), Industrial Technology Research Institute (ITRI), Hsinchu,Taiwan, ITIS project report, 1997.

[5] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang, “A programmable BIST core for embedded DRAM,” *IEEE Design & Test of Computers*, vol. 16, pp. 59–70, Jan.–Mar. 1999.

[6]. “Semiconductor industry association,” in *International Technology Roadmap for Semiconductors (ITRS)*, Dec. 2000.

[7]. K. K. Saluja, S. H. Sng, and K. Kinoshita, "Built-in self-testing RAM: A practical alternative," *IEEE Design & Test of Computers*, vol. 4, pp. 42–51, Feb. 1987.

[8]. M. Franklin and K. K. Saluja, "Built-in self-testing of random-access memories," *IEEE Computer*, pp. 45–56, Oct. 1990.

[9]. B. Nadeau-Dostie, A. Silburt, and V. K. Agarwal, "Serial interface for embedded-memory testing," *IEEE Design & Test of Computers*, vol. 7, pp. 52–63, Apr. 1990.

[10]. P. Camurati, P. Prinetto, M. S. Reorda, S. Barbagallo, A. Burri, and D. Medina, "Industrial BIST of embedded RAMs," *IEEE Design & Test of Computers*, vol. 12, pp. 86–95, 1995.

[11]. S. Tanoi, Y. Tokunaga, T. Tanabe, K. Takahashi, A. Okada, M. Itoh, Y. Nagatomo, Y. Ohtsuki, and M. Uesugi, "On-wafer BIST of a 200-Gb/s failed-bit search for 1-Gb DRAM," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1735–1742, Nov. 1997.

[12]. J. Dreibelbis, J. Barth, H. Kalter, and R. Kho, "Processor-based built-in self-test for embedded DRAM," *IEEE J. Solid-State Circuits*, pp. 1731–1740, Nov. 1998.

[13]. R. P. Treuer and V. K. Agarwal, "Built-in self-diagnosis for repairable embedded RAMs," *IEEE Design & Test of Computers*, vol. 10, pp. 24–33, June 1993.

[14]. C.-W. Wang, C.-F. Wu, J.-F. Li, C.-W. Wu, T. Teng, K. Chiu, and H.-P. Lin, "A built-in self-test and self-diagnosis scheme for embedded SRAM," in *Proc. Ninth IEEE Asian Test Symp. (ATS)*, Taipei, Dec. 2000, pp. 45–50.

[15]. P. Mazumder and J. S. Yih, "A novel built-in self-repair approach to VLSI memory yield enhancement," in *Proc. Int. Test Conf. (ITC)*, 1990, pp. 833–841.

[16]. A. Tanabe, T. Takeshima, H. Koike, Y. Aimoto, M. Takada, T. Ishijima, N. Kasai, H. Hada, K. Shibahara, T. Kunio, T. Tanigawa, T. Saeki, M.Sakao, H. Miyamoto, H. Nozue, S. Ohya, T. Murotani, K. Koyama, and T. Okuda, "A 30-ns 64-Mb DRAM with built-in self-test and self-repair function," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1525–1533, Nov. 1992.

[17]. T. Chen and G. Sunada, "Design of a self-testing and self-repairing structure for highly hierarchical ultra-large capacity memory chips," *IEEE Trans. VLSI Systems*, vol. 1, pp. 88–97, June 1993.

[18]. P. Mazumder and Y.-S. Jih, "A new built-in self-repair approach to VLSI memory yield enhancement by using neural-type circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 124–136, Jan. 1993.

[19]. I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L.

Lweandowski, “Built in self repair for embedded high density SRAM,” in *Proc. Int. Test Conf. (ITC)*, Oct. 1998, pp. 1112–1119.

[20]. D. K. Bhavsar, “An algorithm for row-column self-repair of RAMs and its implementation in the alpha 21 264,” in *Proc. Int. Test Conf. (ITC)*, 1999, pp. 311–318.

[22]. N. Park and E. Lombardi, “Repair of memory arrays by cutting,” in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, San Jose, Aug. 1998, pp. 124–130.

[23]. S.-K. Lu and C.-H. Hsu, “Built-in self-repair for divided word line memory,” in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, 2001, pp. 13–16.

[24]. T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada, and H. Hidaka, “A built-in self-repair analyzer (CRESTA) for embedded DRAMs,” in *Proc. Int. Test Conf. (ITC)*, 2000, pp. 567–574.

[25]. J. Ohtani, T. Ooishi, T. Kawagoe, M. Niiro, M. Maruta, and H. Hidaka, “A shared built-in self-repair analysis for multiple embedded memories,” in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, vol. 4, 2001, pp. 187–190..

[26]. R.W. Haddad, A. T. Dahbura, and A. B. Sharma, “Increased throughput for the testing and repair of RAMs with redundancy,” *IEEE Trans. Computers*, vol. 40, pp. 154–166, Feb. 1991..

[27]. M. Choi and N. Park, “Dynamic yield analysis and enhancement of FPGA reconfigurable memory system,” in *Proc. 18th IEEE Instrumentation and Measurement Technology Conf. (IMTC)*, vol. 1, Budapest, Hungary, May 2001, pp. 386–391.

[28]. M. Choi, N. Park, F. Meyer, F. Lombardi, and V. Piuri, “Reliability measurement of fault-tolerant onboard memory system under fault clustering,” in *Proc. 19th IEEE Instrumentation and Measurement Technology Conf. (IMTC)*, vol. 2, Anchorage, AK, May 2002, pp. 1161–1166.

[29]. C.-F.Wu, C.-T. Huang, C.-W.Wang, K.-L. Cheng, and C.-W.Wu, “Error catch and analysis for semiconductor memories using march tests,” in *Proc. IEEE/ACMInt. Conf. Computer-Aided Design (ICCAD)*, San Jose, Nov. 2000, pp. 468–471.

[30] V.R.Seshagiri Rao, Dr. M.Asharani “Reliability measurement of memory system using spare blocks ,” International journal of electrical engineering & technology (ijeet) Vol 9, Issue 1, Jan-Feb 2018, pp. 18–25; ISSN Print: 0976-6545 and ISSN Online: 0976-6553

[31] V.R.Seshagiri Rao, Dr. M.Asharani “Redundancy Algorithm Using Line Based Method for Memories achieving Less Area

Overhead ," "International Journal of Engineering and Technology (UAE) (IJET) ISSN: 2227-524X

[32] V.R.Seshagiri Rao, Dr. M.Asharani "Global Spare Blocks for Repair of Clustered Fault Cells in Embedded Memories ," Journal of Computational and Theoretical Nano science printed in the United States of America Vol. 16, No. 1, 924–933, 2019.