International Journal of Hybrid Information Technology
Vol.6, No.6 (2013), pp.269-280
http://dx.doi.org/10.14257/ijhit.2013.6.6.24

At-Speed Wordy-R-CRESTA Optimal Analyzer to Repair Word-Oriented Memories

Rahebeh Niaraki Asli, Shahin Khodadadi and Payam Habiby

University of Guilan, Rasht, Iran

niaraki@guilan.ac.ir, khodadadishahin@yahoo.com, payamhabiby@yahoo.com

Abstract

Defect probabilities in embedded memories have been increased by the scaling reduction in advanced VLSI technology. Built-In Self Repair (BIRA) circuit design is a solution to improve the yield drop and get to a high reliability. This paper will present an at-speed optimal algorithm to repair Word-Oriented memories. In spite of parallel structure in our analyzer to get high analysis speed, the area consumption of the circuit has decreased considerably due to the optimal algorithm. The repair rate of the proposed method is also 100% because of applying the binary search three as its foundation. Simulation results and comparison with other methods show the efficiency of the Wordy-R-CRESTA analyzer method.

Keywords: *Built-in redundancy analyzer, built-in self-repair, word-oriented memory, Comprehensive Real-time Exhaustive Search Test and Analysis (CRESTA)*

1. Introduction

SOC density has been considerably increased by the gate length fall and the growth of integration level. This upward trend density of chips leads to an increase in defects and likely errors in SOCs. On the other hand, the importance of data processing and saving in digital devices causes a rising demand for larger embedded memories and increases the probability of errors due to these memories. In the literature, different memory repair algorithms have been presented with various combinations of redundancies to replace the faulty parts. Yet, the rows and columns spares replacing is a common method. This process is called linear replacement. A Built-In Redundancy Analyzer (BIRA) plays a pivotal role to determine the important evaluation factors including repair rate, speed analysis and area consumption of the Built-In Self Repair (BISR) circuits used for comparison the different methods. Repair rate is expressed as follows [1]:

$$\text{Repair Rate} = \frac{\#\text{ of repaired chips}}{\#\text{ of faulty chips}} \tag{1}$$

$$\text{Normalized Repair Rate} = \frac{\#\text{ of repaired chips}}{\#\text{ of repairable chips}} \tag{2}$$

If the normalized repair rate is to be 100%, BIRA is said to achieve the optimal repair rate and the memory is repairable, this means that the BIRA can offer at least one solution to repair the memory [2].

ISSN: 1738-9968 IJHIT

On the other hand, Word-Oriented Memories (WOMs) are easy to manufacture and coding and have been vastly used in SOCs. Developments and improvements in the algorithms of memory testing have made the simultaneous announcement of several faults possible. The announced fault from BIST to the BISR circuit for WOMs is in the form of (R, W, Fail_Word) [3] where R, W and Fail_Word (FW) respectively represent row address, Word address and a string of binary which extracted by MARCH algorithm [4].

In this paper, we will present a new BIRA for WOMs with improvements in several ways. It is able to handle at-speed multiple bit failure in a WOMs. The proposed method achieves an essential reduction in area overhead in comparison to previous methods. Beside this, the repair rate of the proposed BIRA is optimum. Section II reviews the existing BIRA presented in the literature and used to repair WOMs. Section III introduced the proposed BIRA to repair WOMs. The results of the simulations and comparisons have reported in section IV. Finally, this paper is concluded in section V.

2. Previous BIRA Methods

Different BIRAs apply various methods to allocate the spare redundancies. A BIRA method called Extended Essential Spare Pivoting (EESP) was presented in [5]. In the first step, the information of faulty word is collected and saved. After reporting faults by BIST, the faulty word address is compared with data stored in the Content Addressable Memory (CAM). If the announced address matches up with the CAM's data, the corresponding match flag in CAM is set to 1 and the counter increments. After completing this procedure for all reported faults, the content of the counter is compared with a threshold number. If it is greater than or equal to the threshold number, the second part of BIRA awake to repair the faulty columns or rows. EESP algorithm has much area consumption. A 2D-redundancy using 1-D local bitmap BIRA is presented in [6]. In this method, the arrays of a memory are divided into two sub-arrays and the allocation of the redundancies is in a special way. The spare row is applied for both parts of the memory, whereas the spare columns are separately utilized for the two parts of memory. This means that one column redundancy is used to repair the errors existing in the left sub-arrays and the other column redundancy is applied for the right sub-array. A bitmap saves the fault addresses sent from BIST. This method has an efficient repair rate but can't achieve an optimal repair rate. This method also has an efficient area overhead but the speed of analysis is low. That's because for each reported fault, BIST stops through sending a test_done signal to BIRA and BIRA carries out the required analysis [7, 8].

Another design is presented for BIRA in [9] which uses 3-D redundancies. Spare redundancies are a little different in this method, *i.e.* a spare word is used rather than a spare column. A local bitmap is used to store error information of the faulty word. If the local bitmap cannot store any new fault information, it is full. In this case, if a spare word is available and if the number of errors is more than the threshold, the spare word is allocated to the faulty word. If the number of errors is lower than the threshold, BIRA uses spare rows and columns as a replacement. Of course it is the use of Local Repair Most (LRM) algorithm that makes the row and column allocation possible [1]. In LRM algorithm, BIRA and BIST work on a parallel basis and this increases the repair speed. Yet all these advantages are created through a reduction in optimal repair rate [10].

Most-Repair Analyzer (MRA) is another method to repair memory presented in [3]. Each CAM exploited for address storing, has an excess bit to determine faulty word. While an error is reported, the number of 1s stored in CAMs is computed by a counter. If the number of these 1s is more than the threshold, that row or column must be repaired. Thus the address of this row or column will be a section of the solution. In the end, all these addresses stored in CAMs will be analyzed by SOLVER and a suitable solution will be determined.

CRESTA and other optimized methods are from the first methods used to repair bit-oriented memories [11-13]. In [14] a CRESTA is also used to repair WOMs. This method uses a register called Column Repair Vector (CRV) to store column failure information. By analyzing CRV at the end of memory built-in self testing, analyzer determines whether a given strategy can repair the memory or not. After announcing one error, if the current spare resource is a spare row and there is a failure not covered by the CRV, a spare row is allocated to that failure. If the current spare resource is a spare column and there is a failure which is not covered by the previous rows, a spare column is reserved and CRV keep a history of all faulty columns. This is done by ORing the current CRV and the fault information. After the completion of analysis, if the number of 1s in CRV is fewer than the number of spare columns, that sub-analyzer will lead to memory repair. CRV has the same as the word width of the memory. Due to a large number of registers used in this method, the required consumption level is high [15]. In a memory with R_S spare rows and C_S spare columns, CRESTA contains $C(R_S + C_S, R_S)$ sub-analyzers, where $C(R_S + C_S, R_S)$ is the number of ways to choose R_S elements out of ($R_S + C_S$) elements. Hence, the area overhead of CRESTA drastically increases with the number of redundancies. R-CRESTA [12], [14] by reviewing the parallel sub-analyzers has achieved 25% reduction of area overhead compared with CRESTA in use of two row and two column spares.

3. The Proposed BIRA for Repairing WOMs

In the proposed algorithm, we're going to reduce the area consumption through the improvement of binary search tree structure by using spare words for WOMs as a result of considering R-CRESTA structure as the main foundation. Figure 1 shows the proposed binary search tree where the spare redundancies used consist of row and word redundancies which are represented with R and W respectively.

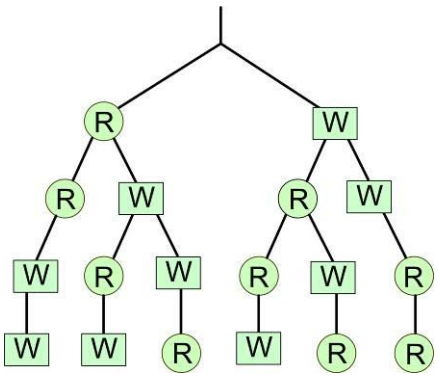

Figure 1. The proposed binary search tree

There are two spare rows and two spare words. Each branch represents a sub-analyzer consisting of four cells. Under this assumption, the possible solutions to repair a memory are as follows:

{RRWW, RWRW, RWWR, WRRW, WRWR, WWRR}

Figure 2 illustrates a faulty 8×8×4-bit WOM which has two spare rows and two spare words. The numbers in the memory show the order of announced errors. Binary numbers also shows FW. The width of the vertical redundancies equals the word width of the memory, that

is, 4. The memory repair flowchart, using the proposed algorithm, is illustrated in Figure 3. R-CRESTA uses CAM arrays to save fault addresses which are capable of comparing input addresses with already stored contents. When BIST announces a new fault address, it is sent to all sub-analyzers and is compared with the content of *CAM_R* and *CAM_W* and if this address has not been saved yet, it will be saved in the first unused CAM place. When all redundancies have been used and a new error address is reported again, the solution of related sub-analyzers is considered as an unsuccessful solution. If there is a spare row or spare word, the allocation operation will be carried out. Considering the flowchart in Figure 3 and the announced errors in Figure 2, saving the faulty addresses in cells will be as Figure 4 (a). The result is achieving WRRW strategy to repair the memory like Figure 4 (b).

Figure 2. A faulty 8×8×4-bit memory

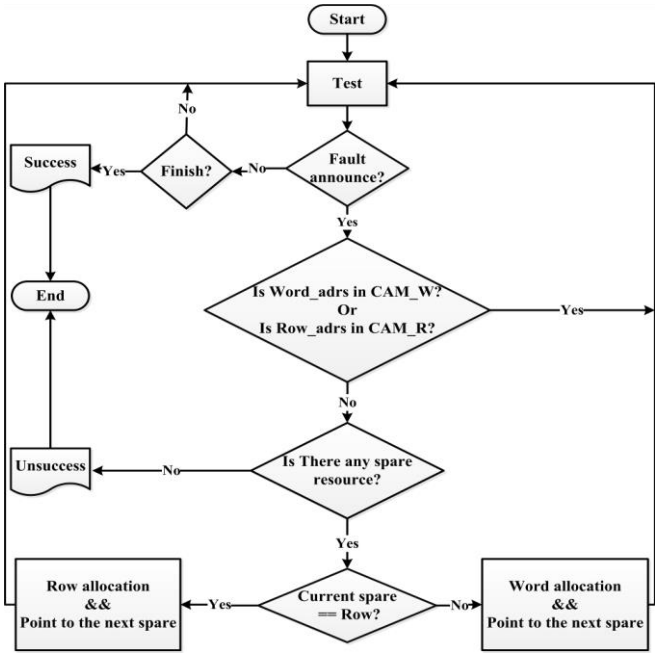

Figure 3. The flowchart of the proposed BIRA

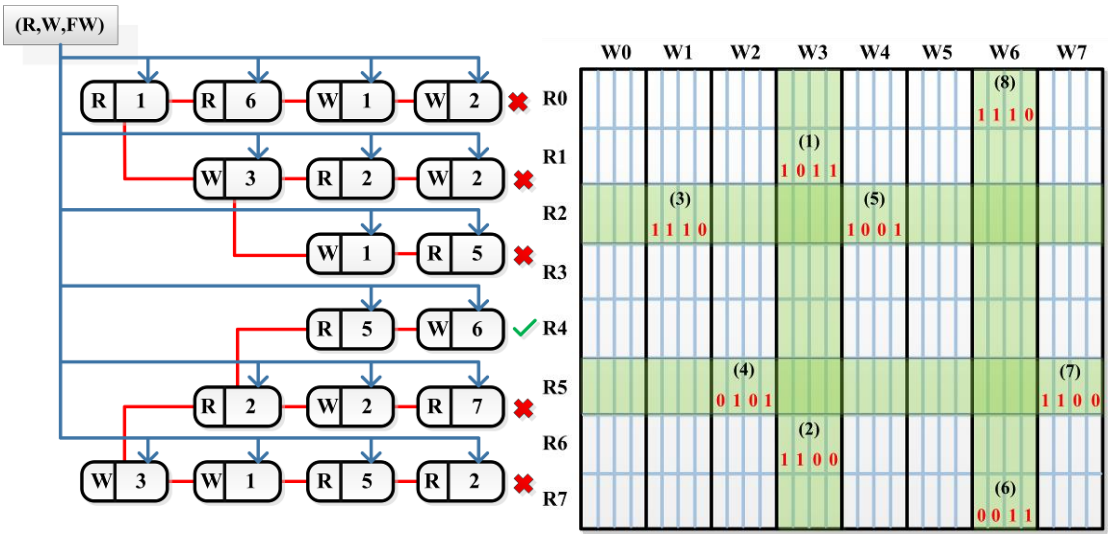

Figure 4. (a): Conceptual diagram of saving the fault addresses in the cells of the proposed BIRA. (b): Repaired memory by means of WRRW strategy

4. Experimental Results

Figure 5 shows RTL diagram of each proposed analyzing cell using Xilinx ISE 13.3 software. Each cell consists of two major components. The first component show the binary search tree result as the row and word address through *Cell_repair_out* which is saved by another component named as *Cell_comparator*. In this circuit, *cell_solution* signal selects the word or row address which is to be transferred through the MUX. After the error announcement, when *fault_notification* signal equals to 1, BIRA compares the address with the saved address in the cell. If it isn't an iterative address, it's transferred to the next empty cell. The conceptual block diagram of the proposed sub-analyzer is shown in Figure 6.

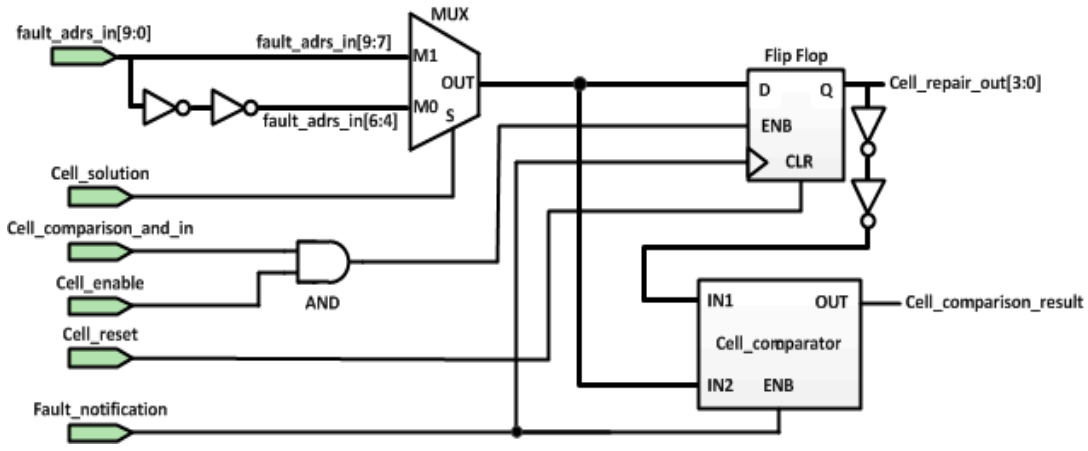

Figure 5. RTL diagram of the proposed cell of BIRA

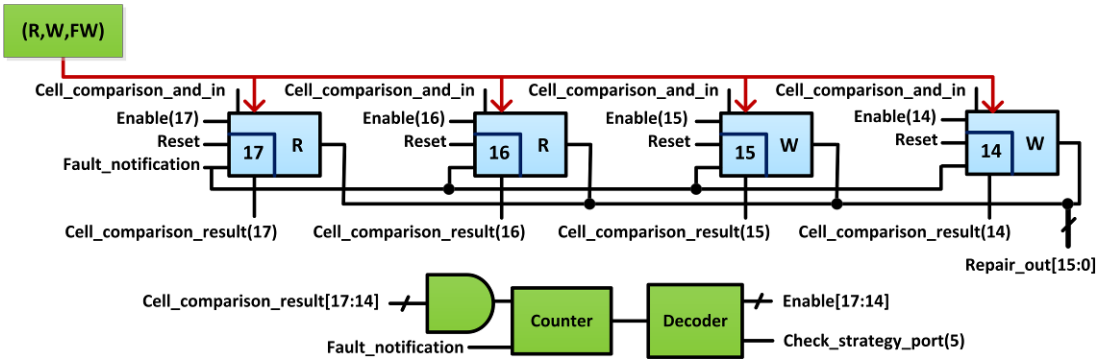

Figure 6. Conceptual block diagram of the proposed sub-analyzer

The *cell_comparison_and_in* signal in each sub-analyzer is the AND result of other cell_comparison_result which is used as an enable port for the other cells. If the counter is placed on number 4 and an error is announced, the sub-analyzer will not be able to fix the memory. The result is 4-bit *repair_out* output. In order to investigate the accuracy of the proposed BIRA, we have simulated and synthesized the designed circuit using Xilinx ISE 13.3 and programmed it on Xilinx Virtex5 XC5VSX50T FPGA. Figure 7 illustrates the result of simulation after using the fault addresses shown in Figure 2. After setting the *test_done* signal by BIST at the end of the testing process, the output *check_strategy_port* is 111011 showing WRRW strategy to repair the memory.

The 16 bit output *repair_out* becomes 1011001001011110 which show the address of rows and words from the memory that have to be replaced by redundancy. This signal consists of four 4-bit data because each sub-analyzer consists of four cells and each cell consists of 4-bit data which shows the type of repair and the fault address. For example "1" at the beginning of the *repair_out* signal shows W strategy and "011" tell us that the fault address is 3. So the allocation strategy must be W3. Then "0" shows the R strategy and "010" shows the second row of the memory, so the allocation strategy is R2 and in this case the rows and words which require a redundancy allocation are as follows: {W3, R2, R5, W6}.

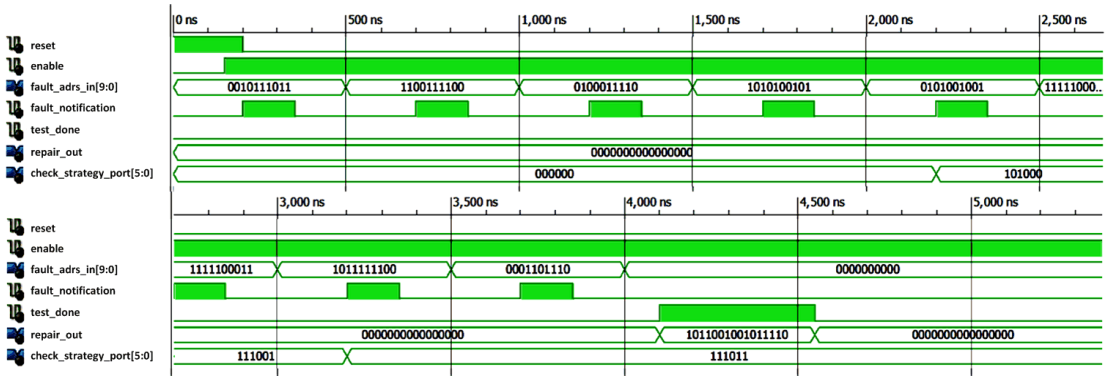

Figure 7. Simulation waveform of the proposed BIRA using fault addresses shown in Figure 2

Table 1 shows the resource usage of the proposed BIRA on XC5VSX50T FPGA for the 8×8×4-bit memory. According to the table, the number of required registers for 2×2 redundancy configuration is 72. The number of occupied slices of XC5VSX50T for implementing design is 94.

Table 1. Resource usage of XC5VSX50T chip for presented BIRA for 8×8×4-bit memory using 2×2 redundancy configuration

Logic Utilization	Usage
Number of Slice Registers	72
Number of Slice LUTs	185
Number of Bonded IOs	105
Number of Occupied Slices	94

Replacement of a single-column redundancy with a spare word is the cost of achieving the optimized repair rate and the suitable speed. The proposed BIRA uses R-CRESTA method as foundation and works based on binary search tree consequently independent of errors number it has always optimized repair rate. Figure 8 shows the comparison between the proposed BIRA's normalized repair rate with the methods existing in [3] and [6] for 8192×64-bit memory which are summarized in [7]. In order to control the cost of redundancies, our design for WOMs has the least number of spare words. In this circumstance, the proposed design is closer to R-CRESTA's BIRA for bit-oriented memories.

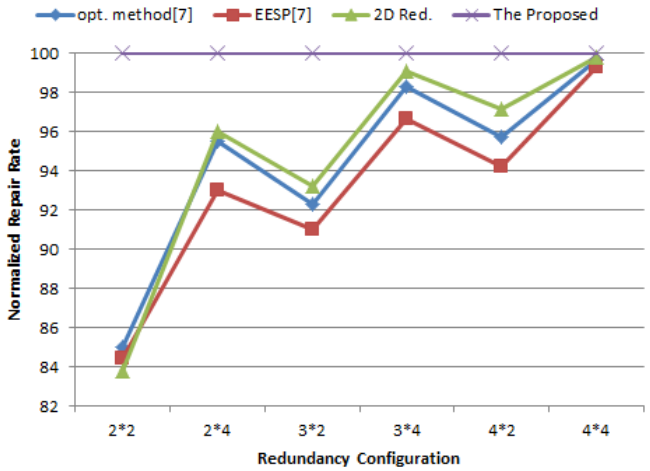

Figure 8. Comparison between the proposed BIRA's repair rate with the methods existing in [7] using 8192×64bit memory

Figure 9 compares the clock cycles of analyzers for various BIRAs to start analysis. The proposed BIRA, like bit-oriented R-CRESTA, execute redundancy analysis whenever a fault is detected by the BIST circuit and find the correct repair solution in one clock cycle due to the concurrent sub-analyzers hardware. EESP and 2D Redundancy analyzers are required to wait for finishing test algorithms to start analysis and have single redundancy analyzer hence need a longer analysis time.

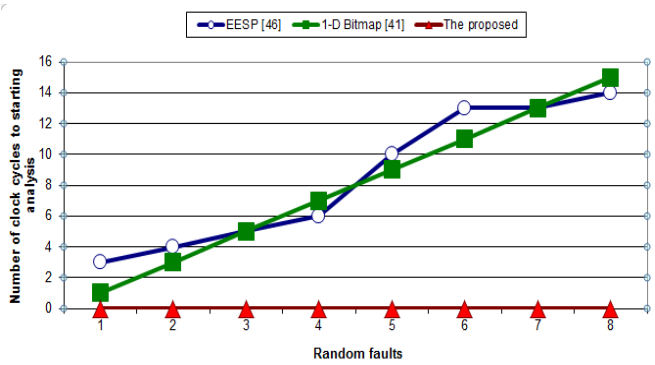

Figure 9. Comparison between the clock cycles of analyzers for various BIRAs to starting analysis

Table 2 summarizes the simulation results of the proposed BIRA circuit with 2 spare rows and 2 spare words applied to three memory sizes. The second row of the table shows the number of required registers. The worst cycle time is shown in the third row that is equal to 16.74 ns. The Critical Analysis Time (CAT) for the proposed BIRA scheme is shown in the last row. The CAT is obtained according to the longest path in the repair strategy which is 6 for the proposed wordy-R-CRESTA algorithm.

Table 2. Simulation results of the BIRA for memories with two spare rows and two spare words

Memory Size	8192×64	4096×128	2048×256
# of registers	67	65	63
Cycle Time	16.74 ns	16.74 ns	16.74 ns
CAT	6	6	6

Generally, the area overhead of the BIRA is consists of two major parts. The first part is the area of the BIRA's circuit which is directly dependent to the number of the registers. The second part of the area overhead is the reserved area that considered as healthy bits called redundancy.

For the proposed BIRA the number of registers follows an equation similar to R-CRESTA's [16]. This can be shown for M×N memory that has R_S spare rows and W_S spare words like below:

$$A_{R-CRESTA} = \left(\frac{3}{4}\right) A_{Spare\_reg} \times \frac{(R_S + W_S)!}{R_S! W_S!} \quad (3)$$

Where A_{Spare_reg} shows the registers required for spare units and can be shown as follows:

$$A_{Spare\_reg} = [(\log_2^M + 1) \times R_S + (\log_2^N + 1) \times W_S] \quad (4)$$

Table 3 shows the area overhead comparison between EESP, 1-D bitmap methods and the proposed BIRA based on required registers for a memory with M=6, N=7 and W=32. The results show, the required registers for the proposed BIRA is much smaller

than that of the EESP and 1-D bitmap methods. For example, the area reduction of the proposed BIRA area overhead in the worst case for 2×2 redundancy configuration is 78.8% and 63.6% for EESP and 1-D bitmap, respectively.

Table 4 shows the percentage of the required redundancies area overhead to memory size for 6×7×32-bit memory. As it can be seen, it's highly probable to considerably increase the area consumption for BIRA implementation through considering a wider redundancy. Undoubtedly, different redundancy configurations play an important role to ascertain the proportions. According to the table, in the worst case, at the cost of at speed testing, when we use two spare rows and two spare words, the proposed BIRA reserves 52.4% whilst EESP and 1-D bitmap occupy 34.2% of the 6×7×32-bit memory as redundancies. On the other hand, as we can see in Table 3, the proposed BIRA implementation needs only 67 registers whilst EESP and 1-D bitmap need 316 and 184 registers, respectively. In other words, reserving 18.2% more memory cells as redundancy make the BIRA's area overhead 78.8% and 63.6% smaller than EESP and 1-D bitmap, respectively, and at speed testing has been possible. Moreover, the proposed BIRA can cover more faults because of the wider wordy redundancies.

Table 3. The area overhead of different BIRAs based on required registers for 6×7×32-bit memory

(R,C) for EESP & 1-D Bitmap/ (R,W) for the Proposed BIRA	EESP [7] A	1-D Bitmap [7] B	The proposed C	The area reduction $(1-\frac{C}{A})\%$	The area reduction $(1-\frac{C}{B})\%$
(1,1)	158	92	12	92.4%	86.9%
(1,2)	237	138	23	90.3%	83.3%
(1,3)	316	184	46	85.4%	75%
(2,1)	237	138	25	89.4%	91.9%
(2,2)	**316**	**184**	**67**	**78.8%**	**63.6%**
(3,1)	316	184	44	86.1%	76.1%

Table 4. The percentage of the required redundancies area overhead to memory size for 6×7×32-bit memory

(R,C) for EESP & 1-D Bitmap/ (R,W) for the Proposed BIRA	$\frac{(R,C)\text{ area}}{\text{memory size}}\%$ A	$\frac{(R,W)\text{ area}}{\text{memory size}}\%$ B	$B-A$
(1,1)	17.1%	28.5%	11.4%
(1,2)	17.6%	40.5%	22.9%
(1,3)	18%	52.4%	34.4%
(2,1)	33.8%	42.8%	9%
(2,2)	**34.2%**	**52.4%**	**18.2%**
(3,1)	50.5%	57.1%	6.6%

5. Conclusion

In this paper, at speed BISR analyzer design applicable to word-oriented memories has been presented which handles multiple-failure in a word. The design is simulated by Xilinx 13.3 and implemented on Virtex5 XC5VSX50T FPGA. The repair rate, irrespective of the number of memory faults, is always at its maximum because the new strategies presented in our method for WOMs have been designed according to binary search tree. The simulation results show, due to the lack of bitmaps and at the cost of wider redundancies in our analyzer, the number of the required registers is less than EESP and 1-D bitmap methods. In the worst case, for a 2×2 redundancy configuration, the proposed BIRA area reduction is equal to 78.8% and 63.6%, respectively. These comparisons approve the design performance.

References

[1] C. -T. Huang, C. -F. Wu, J. -F. Li and C. -W. Wu, "Built-in redundancy analysis for memory yield improvement", IEEE Trans. Reliability, vol. 52, no. 4, **(2003)** December, pp. 386–399.
[2] 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. VLSI System, vol. 17, no. 12, **(2009)**.
[3] J. Chung, J. Park and J. A. Abraham, "A built-in repair analyzer with optimal repair rate for word-oriented memories", IEEE Trans. VLSI Systems, **(2012)**.
[4] N. A. Zakaria, W. Z. W. Hasan, I. A. Halin, R. M and X. Wen, "Fault Detection with Optimum March Test Algorithm", IEEE Third International Conference on Intelligent Systems Modelling and Simulation, **(2012)** pp. 700-704.
[5] S. K. Lu, Y. -C. Tsai, C. -H. Hsu and K. -H Wang, "Efficient built-in redundancy analysis for WordOriented embedded memories with hierarchical redundancy", IEEE Trans. VLSI System, vol. 14, no. 1, **(2006)**, pp. 34-42.
[6] T. -W. Tseng, J. -F. Li and D. -M. Chang, "A built-in redundancy-analysis scheme for RAMs with 2D redundancy using 1D local bitmap", In Proc. IEEE DATE Conf., **(2006)**.
[7] T. -W. Tseng and J. -F. Li, "A Low-Cost Built-In Redunddancy-Analysis Scheme for Word-Oriented RAMs With 2-D Redundancy", IEEE Trans. On VLSI Systems, vol. 19, no. 11, **(2011)**, pp. 1983-1995.
[8] T. -W. Tseng and J. -F. Li, "ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs", IEEE Trans. On VLSI Systems, **(2010)**, pp. 921-932.
[9] Y. -J. Chang, Y. -J. Huang and J. -F. Li, "A Built-In Redundancy-Analysis Scheme for RAMs with 3D Redundancy", IEEE Int. Test Conf., **(2011)**.
[10] P. Habiby and R. N. Asli, "An Improved BIRA for Memories with Optimal Repair Rate Using a Flipping Analyzer", IEEE, 20th ICEE, **(2012)**.
[11] T. Kawagoe, J. Ohtani, M. Niiro, T. Ooshi, M. Hamada and H. Hidaka, "A built-in self-repair analyzer (CRESTA) for embedded DRAMs", IEEE Int. Test Conf., **(2000)**, pp. 567-574.
[12] W. Jeong, T. Han and S. Kang, "An advanced BIRA using parallel sub-analyzers for embedded memories", IEEE Int. Conf., **(2009)** on SOC Design, pp. 249-25.
[13] P. Habiby and R. N. Asli, "Design and Implementation of a New Symmetric Built-In Redundancy Analyzer", IEEE Int. Symposium on CADS, **(2012)**, pp. 99-103.
[14] D. Xiaogang, S. M. Reddy, W. -T. Cheng, J. Rayhawk and N. Mukherjee, "At-speed built-in self-repair analyzer for embedded word-oriented memories", in Proc. Int. Conf. VLSI Design., **(2004)**, pp. 895–90.
[15] T. Chen, J. Li and T. Tseng, "Cost-Efficient Built-In Redundancy Analysis With Optimal Repair Rate for RAMs", IEEE Trans on CAD of ICS, vol. 31, no. 6, **(2012)**.
[16] W. Jeong, J. Lee, T. Han, K. Lee and A. Kang, "An Advanced BIRA for Memories with an Optimal Repair Rate and Fast Analysis Speed by Using a Branch Analyzer", IEEE Trans. On CAD of IC and system, vol. 29, no. 12, **(2010)**.

Authors

Rahebeh Niaraki Asli

She received her B.S. and M.S. degrees in Electronic Engineering from the University of Guilan, Rasht, Iran, in 1995 and 2000, respectively. Also, she received Ph.D. degree in electrical engineering from the Iran University of Science and Technology, Tehran, Iran, in 2006.

From 1995 to 2002 she has worked in electronic laboratories of the Department of Electrical Engineering in the University of Guilan. During 2002 to 2006, she was with design circuit research group in the Iran University of Science and Technology electronic research center (ERC) and CAD research group of Tehran University. Since 2006, she has been an Assistant Professor in Department of Electrical Engineering, Engineering faculty of Guilan University. Her current research interests include design for testability, embedded memory testing, diagnosis, and repair, VLSI CAD design, and soft errors.

Shahin Khodadadi

He received his B.S. degree in Electronic Engineering from University of Guilan, Iran, in 2011.He is also accepted in M.S. degree in Electronic Engineering in University of Guilan, Iran, in 2011. His current research interests include design for testability, memory testing, diagnosis and repair.

Payam Habiby received his B.S. degree in Electronic Engineering from University of Guilan, Iran, in 2007. He also received his M.S. degree in Electronic Engineering from University of Guilan, Rasht, Iran, in 2012. He is currently working as an electronic engineer at Iranian Oil Terminal Company (IOTC). His research interests include embedded memory test and repair and VLSI CAD design.