A Fast Redundancy Analysis Algorithm in ATE for Repairing Faulty Memories

Hyungjun Cho, Wooheon Kang, and Sungho Kang

Testing memory and repairing faults have become increasingly important for improving yield. Redundancy analysis (RA) algorithms have been developed to repair memory faults. However, many RA algorithms have low analysis speeds and occupy memory space within automatic test equipment. A fast RA algorithm using simple calculations is proposed in this letter to minimize both the test and repair time. This analysis uses the grouped addresses in the faulty bitmap. Since the fault groups are independent of each other, the time needed to find solutions can be greatly reduced using these fault groups. Also, the proposed algorithm does not need to store searching trees, thereby minimizing the required memory space. Our experiments show that the proposed RA algorithm is very efficient in terms of speed and memory requirements.

Keywords: Redundancy analysis (RA), fault groups.

I. Introduction

To increase device yields, many manufacturers use incorporated redundancy that can be used to replace faulty cells. Several studies have shown that the redundancy algorithm is NP-complete. Therefore, many proposed redundancy analysis (RA) algorithms [1]-[5] are based on exhaustive binary search trees and require a long search time for nearly all of the serious cases. A new RA algorithm is needed to reduce the analysis time. If the memory space required for the search tree is greater than the usable memory space in the automatic test equipment (ATE), the memory repair should be stopped. Therefore, the RA algorithm should create a solution using the limited memory space in ATE.

The repair-most (RM) algorithm [1] is greedy, and, although it is not optimal, it is very simple and has thus been adopted as an RA algorithm in many cases. Kuo and Fuchs proposed the branch-and-bound algorithm, which is a heuristic method based on the exhaustive binary search tree [1]. The intelligent solve first (ISF) algorithm [2], minimized binary search tree algorithm [3], and BRANCH algorithm [4] are built-in self-repair (BISR) algorithms that use a binary search tree structure. The ISF algorithm has an optimal repair rate, but its RA speed is much slower than that of other binary search algorithms. To reduce the search time, [3] minimizes a binary search tree and BRANCH concurrently analyzes all nodes of a branch for the binary search tree. As a result, [3] and BRANCH enable the algorithm to find the solution faster than ISF. However, due to the large hardware overhead needed for BISR, many memories are generated without it. For this reason, ATEs with the RA algorithms are used to test and repair memories. PAGEB [5] creates a solution that transforms a spare allocation problem into Boolean functions, and the renowned binary decision diagram is used to manipulate them. PAGEB consumes less memory space and time when compared to the branch-and-bound methods. However, because PAGEB should always determine the defect function (DF), constraint function (CF), and repair function (RF), it is time-consuming and requires significant memory space. The DF is a Boolean function that encodes the locations of all faulty cells, and the CF is a Boolean function encoding all combinations of faulty lines replaceable by spare lines. In the end, RF is a Boolean function that encodes all repair solutions for a space allocation problem. Therefore, PAGEB must calculate the RF generated by the Boolean-AND operation between the stored DF and CF.

Manuscript received Sept. 2, 2011; revised Oct. 20, 2011; accepted Nov. 3, 2011.

This work was supported by a grant from the National Research Foundation of Korea (NRF), funded by the Korean government (MEST) (No. 2010-0024707).

Hyungjun Cho (phone: +82 2 2123 2775, chj0937@soc.yonsei.ac.kr), Wooheon Kang (sudal@soc.yonsei.ac.kr), and Sungho Kang (corresponding author, shkang@yonsei.ac.kr) are with the Department of Electrical & Electronic Engineering, Yonsei University, Seoul, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.12.0211.0378

A fast RA algorithm using a simple calculation is proposed in this letter and compared with the above RA algorithms structured via C-language. The proposed algorithm determines a solution for repairing memory faults with a time of nearly zero and is also the most easily implemented algorithm for ATE tests. Also, since the proposed algorithm does not need to store the search spaces, it can greatly reduce the amount of memory used.

II. Proposed RA Algorithm

The proposed algorithm creates fault groups that are irrespective of each other. The faulty bit map and the fault group information are simultaneously generated during the detection of faults by the ATE.

Definition (fault group). A fault group is defined as a set of single or multiple faults. A single fault group is the same as an orthogonal fault [6]. A multiple fault group is defined as a set of faults that has the same row or column address as other faults.

Figure 1 shows an example with a single fault group and two multiple fault groups. All faults are grouped into single or multiple fault groups when they are detected in the ATE. For example, the table to the right in Fig. 1 shows a faulty bit map with a fault group number. The fault group list is determined as the ATE searches the faults in order. Each multiple fault group has at least two faults that have the same row or column address irrespective of other fault groups.

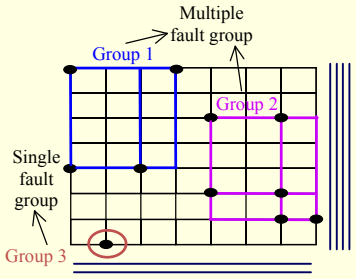
The proposed algorithm provides a solution for repairing faulty memories using the following fault group properties:

Property 1. In a multiple fault group, there are no faults that have the same row/column address as faults in other fault groups.

Due to Property 1, the proposed algorithm can provide repair solutions in each fault group irrespective of the solutions for other groups in the remaining spare cells. Since single fault groups can be repaired by any redundant cells regardless of the row or column, they are repaired by remaining redundant cells after all faults in the multiple fault groups have been repaired.

Property 2. If the number of groups is more than R_S+C_S, the memory cannot be repaired (an early termination solution).

At least one row/column spare cell is needed to repair the faults in a fault group. To repair a fault group, one or more redundant cells are needed. Therefore, if the number of groups is the same or less than R_S+C_S, then the RA algorithm should determine whether the memory is repairable or irreparable. This feature allows the irreparable memories to be found early and then terminated before the RA algorithm is performed. Property 2 is applicable to other RA procedures that occur prior to the solutions searching process.



Fault no.	Fault address		Group no.
	Row	Column	
1	0	0	1
2	0	3	1
3	2	4	2
4	2	6	2
5	4	0	1
6	4	2	1
7	5	4	2
8	6	4	2
9	6	6	2
10	6	7	2
11	7	1	3

Fig. 1. Example of fault groups.

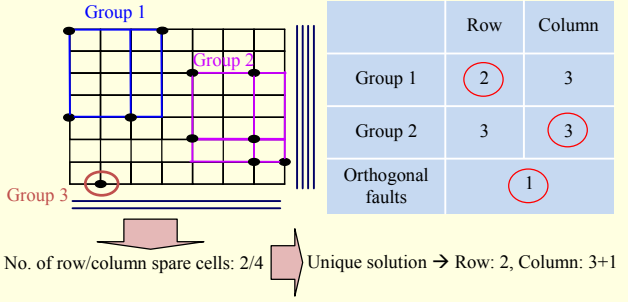


Fig. 2. Example of FAST algorithm.

The proposed RA algorithm is called the FAST algorithm. The FAST algorithm generates a solution using Properties 1 and 2 of the fault groups. Initially, if the number of fault groups is more than the number of redundancy cells, the faulty memory cannot be repaired using other RA algorithms. Therefore, with the use of this early termination solution, irreparable memories are terminated prior to the execution of an RA algorithm. Moreover, the FAST algorithm checks the number of faulty row/column lines in all of the multiple fault groups. Subsequently, it combines the number of faulty row/column lines in all of the multiple fault groups. As a result, a solution is generated using a combination of the number of faulty row/column lines, excluding the number of single fault groups. The sum of the number of row/column lines of the solution and the number of single fault groups must be less than the number of remaining redundancy cells. Finally, the proposed FAST algorithm repairs the single fault groups using the remaining redundancy cells.

Figure 2 shows an example of a repair solution found via the FAST algorithm. Fault group 1 has two different row addresses and three different column addresses. Fault group 2 has three different row and column addresses. Fault group 3 has only one address, which means that the fault for group 3 is a single fault group. The only single fault group can be repaired after finding solutions for the multiple fault groups. From the above information, solutions can be determined using a combination

of the number of each row/column addresses in the fault groups. In the example shown in Fig. 2, (row: 2, column: 3+1) is a unique solution. Similarly, the FAST algorithm determines a solution for repairing memory with a nearly zero central processing unit (CPU) time after collecting fault groups.

III. Experiment Results

A 1 gigabit (1,024 blocks×1,024×1,024) memory was used for the experiments in this study to ensure a fair comparison. Each experiment was repeated 10,000 times with randomly generated addresses for the faulty cells. These experiments did not consider sharing redundancy cells in other blocks. The experiments were performed for different redundancy configurations and numbers of random faults. The generated faults included a single faulty cell, a row/column line of faulty cells consisting of several adjacent faulty cells in a row/column, or a rectangle of faulty cells affecting 2×2 cells. The table in Fig. 3 shows the generated distributions of the fault types. The experiments were simulated using the "row first strategy," where, if possible, the faulty cells were repaired by row spare cells, regardless of the use of spare cells. Faults were scattered throughout the whole memory area.

Figure 3 shows the average time needed to search for a solution according to changes in the number of faults. These experiments were performed using four and five spare rows/columns. As shown in Fig. 3, the FAST algorithm creates a solution with a time of nearly zero (maximum 0.39 s) regardless of the number of spares and faults. It is evident that even if PAGEB is much faster than the branch-and-bound algorithms [5], it requires a long time to calculate the DF, CF, and RF. Specifically, the calculation time of the CF exponentially increases as the number of spares increases. Therefore, we can see that the graph for PAGEB has an exponentially increasing curve and the graph for the RM has a linearly increasing curve. The RM algorithm is faster than PAGEB, but the repair rate of the RM is very low. As shown in Table 1, even if the FAST algorithm does not always maintain a 100% repair rate, the FAST algorithm has a 100% repair rate when the PAGEB algorithm has a 100% repair rate according to the number of faults. Since the FAST algorithm does not need to store the braches of trees or graphs for Boolean equations, it consumes a minimum amount of memory space. Therefore, it is highly advisable to use the FAST algorithm to test and repair large memories using the ATE. Additionally, if a user desires a 100% repair rate, it can be achieved using branch-and-bound algorithms after performing the FAST algorithm with a time of nearly zero. In other words, if a user wants a 100% repair rate, the FAST algorithm can be used as an early-termination method.

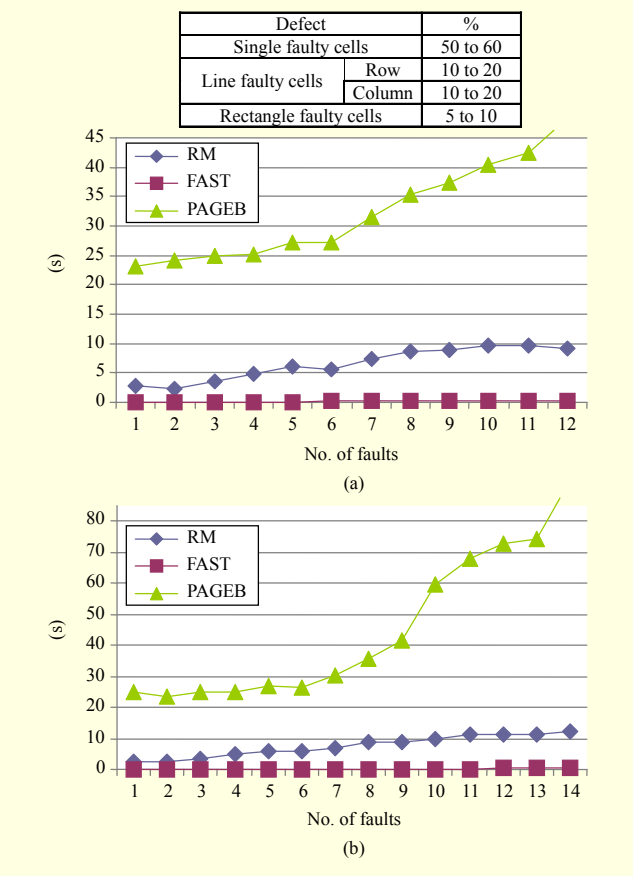


Fig. 3. Average time for searching solution: (a) R_S/C_S: 4/4 and (b) R_S/C_S: 5/5.

Table 1. Repair rate comparison (R_S/C_S: 5/5).

RA algorithm	No. of faults			
	11	12	13	14
RM	100%	100%	95.27%	55.23%
PAGEB	100%	100%	100%	65.34%
FAST	100%	100%	100%	61.84%

IV. Conclusion

As memory size increases, the testing/repair time and amount of memory space used for repairing memories increases accordingly. Therefore, reduced time consumption and required memory space are necessary for memory repair using ATEs. The proposed FAST algorithm repairs faulty cells in the memory with a CPU time of nearly zero. Moreover, it requires almost no memory space when searching for the solution. Therefore, the proposed RA algorithm is useful when simultaneously testing and repairing large memories with many redundant cells.

References

[1] S.-Y. Kuo and W.K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," *IEEE Design Test Computers*, vol. 4, no. 1, Feb. 1987, pp. 24-31.

[2] P. Öhler, S. Hellebrand, and H.-J. Wunderlich, "An Integrated Built-in Test and Repair Approach for Memories with 2D Redundancy," *Proc. IEEE European Test Symp. (ETS)*, May 2007, pp. 91-96.

[3] H. Cho, W. Kang, and S. Kang, "A Built-in Redundancy Analysis with a Minimized Binary Search Tree," *ETRI J.*, vol. 32, no. 4, Aug. 2010, pp. 638-641.

[4] W. Jeong et al., "An Advanced BIRA for Memories with an Optimal Repair Rate and Fast Analysis Speed by Using a Branch Analyzer," *IEEE Trans. CAD*, vol. 29, no. 12, Dec. 2010, pp. 2014-2026.

[5] H.-Y. Lin, F.-M. Yeh, and S.-Y. Kuo, "An Efficient Algorithm for Spare Allocation Problems," *IEEE Trans. Reliability*, vol. 55, no. 2, June 2006, pp. 369-378.

[6] C.-T. Huang et al., "Built-in Redundancy Analysis for Memory Yield Improvement," *IEEE Trans. Reliability*, vol. 52, Dec. 2003, pp. 386-399.