A Built-In Redundancy Analysis with a Minimized Binary Search Tree

Hyungjun Cho, Wooheon Kang, and Sungho Kang

With the growth of memory capacity and density, memory testing and repair with the goal of yield improvement have become more important. Therefore, the development of high efficiency redundancy analysis algorithms is essential to improve yield rate. In this letter, we propose an improved built-in redundancy analysis (BIRA) algorithm with a minimized binary search tree made by simple calculations. The tree is constructed until finding a solution from the most probable branch. This greatly reduces the search spaces for a solution. The proposed BIRA algorithm results in 100% repair efficiency and fast redundancy analysis.

Keywords: BIRA, repair efficiency, binary search tree.

I. Introduction

As the density of memory has increased, the number of related defects has also increased [1]. To increase device yield, many manufacturers use incorporated redundancy that can be used to replace faulty modules. Therefore, the implementation of effective redundancy algorithms is essential.

Recently, a memory test methodology using built-in redundancy analysis (BIRA) algorithms was introduced [2]-[4]. If repairable, a memory with defects can be repaired using comprehensive real-time exhaustive search test and analysis (CRESTA) [2] and IntelligentSolve [3] with 100% accuracy with given redundancies. CRESTA has very short redundancy analysis (RA) time because it analyzes faulty cells using sub-analyzers during searching defect addresses. However, CRESTA incurs an enormous hardware overhead when the number of spare memories is increased. Essential spare pivoting (ESP) [4] repairs faulty cells when the number of faulty cells is more than an essential number in the same address, and local repair-most (LRM) [4] repairs faulty cells using repair-most method with a local bitmap. ESP incurs the lowest hardware overhead costs, but ESP and the LRM do not have 100% repair efficiencies. In particular, ESP has the lowest repair efficiency when the number of spares is large. IntelligentSolve, which is based on the exhaustive binary search tree, is an algorithm to reduce the number of backtracks required to search the repair solution. However, IntelligentSolve requires a long search time in nearly all serious cases. In this letter, we propose a new enhanced BIRA algorithm with a minimized partial search tree.

Manuscript received Jan. 29, 2010; revised Mar. 18, 2010; accepted Apr. 5, 2010.

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

doi:10.4218/etrij.10.0210.0032

II. Previous Works

Because redundancy analysis is NP-complete, the BIRA algorithm must use an exhaustive search method to result in 100% repair efficiency. Therefore, almost all BIRA algorithms which have 100% repair efficiencies use binary search tree methods. The depth of the binary search tree is determined by the number of spare rows (SRs) and spare columns (SCs). Orderly SRs/SCs are located against each fault site. As an example, if the fault sites are located as shown in Fig. 1(a), the result of the exhaustive binary search tree is as shown in Fig. 1(b).

A fault list consists of two tables for storing row/column faulty addresses. Using these two tables, the dynamic must-repair which makes the reduced binary search sites is performed. As an example, the first repair node is chosen using SRs because of the row-first strategy. As the number of SRs is reduced, the must-repair is performed by counting the faults of column addresses. If the faults of the column address are more than the number of SRs, the column node is covered by an SC.

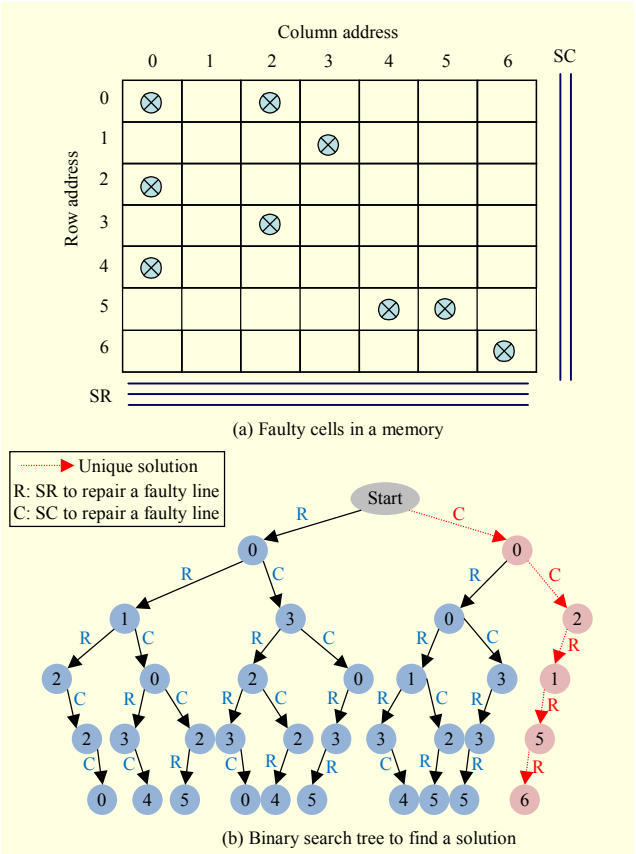

Fig. 1. Example binary search tree (no. of SR/SC: 3/2).

After this process is repeated, the search is performed for the next fault. In this process, the dynamic must-repair confirms whether the branch of the binary tree is repairable or unrepairable before the search is performed. Therefore, the dynamic must-repair reduces the binary search tree sites. In addition, the IntelligentSolveFirst (ISF) method stops when the first solution is confirmed.

III. Proposed Minimized Binary Search Tree

The ISF makes all branches in the order of stored faulty addresses. However, the proposed RA algorithm generates the minimized partial binary search tree which continues to be constructed until finding a solution from the most probable branch. In other words, the proposed algorithm reduces a search space for searching a solution. First, before the binary search tree analysis, the remaining faulty cells are reordered. The time required for redundancy analysis is determined by the number of backtracks necessary to find the solution. However, orthogonal faulty cells that do not have the same address as other faulty cells can be repaired using the remaining SRs/SCs from the last time other faulty cells were repaired. Therefore, the proposed algorithm repairs orthogonal faulty cells after

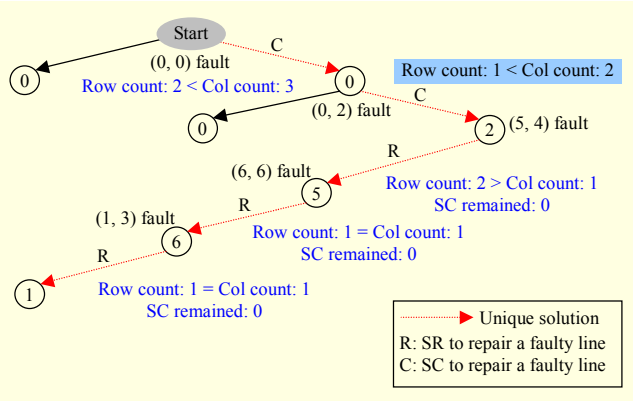

Fig. 2. Repair solution for example shown in Fig. 1 (SR/SC: 3/2).

repairing non-orthogonal faulty cells. This process reduces the depth of the binary tree. The reduced depth also makes the number of backtracks decrease.

The final binary search tree is constructed from the most probable branch. An optimal solution can most likely be found when the search tree is performed from the most probable branch. If the most probable branch is not a solution, then the proposed algorithm finds a solution by backtracking from the branch. For this final binary search tree, information about faulty cells should be acquired from the fault list. This fault list is realized by using two small content addressable memories (CAMs) of size 2SRSC, as in the IntelligentSolve method. These row/column CAMs provide the number of faults of each faulty row/column address. To repair faulty cells, the proposed algorithm decides if we use SRs or SCs through the comparison with the row and column counts. For example, if a fault that has a row address count of 2 and a column count of 1 is inserted, then the branch of a row has a higher probability for an optimal solution. Figure 2 shows the result of a minimized binary search tree solution for the example shown in Fig. 1.

(a) (0, 0) fault: the row count 2 < column count 3. Therefore, the branch of the column address 0 is chosen. The count of the row address 0 is changed from 2 to 1, and the count of the column address 0 is changed from 3 to 0.

(b) (0, 2) fault: as in (a), the row count 1 < column count 2. So, a spare column is chosen. The count of the row address 0 is changed from 1 to 0, and the count of column address 2 is changed from 2 to 0.

(c) (2, 0), (3, 2), and (4, 0) faults already have a solution.

(d) (5, 4), (6, 6), and (1, 3) faults: in (a) and (b), all spare columns were used. Therefore, three SRs should be chosen.

(e) No faults remain at this point. Therefore, a final unique solution is constructed with row addresses of 1, 5, and 6, and column addresses of 0 and 2.

Because of the reduced SC after (a), dynamic must-repair

can be performed. If dynamic must-repair is performed, then row address 5 must be repaired by one SR. Therefore, row address 5 is chosen after the (a) process is performed. In this example, the proposed algorithm makes an optimal solution without any backtracks. In this way, the proposed algorithm can dramatically reduce the number of backtracks, and the time required for the memory test is greatly reduced.

IV. Experimental Results

1024×1024-bit memory was used for the experiments to guarantee fair comparisons with other RA researches. Each experiment was repeated 1,000 times with randomly generated faulty addresses. The distribution of defects was limited to 1/64 of the memory area to obtain optimal repair rates. The trend of the simulated results is nearly identical in comparison to the experimental results with defects scattered around the whole memory area. Figure 3 shows repair efficiencies as a function of the number of defects. The ESP method has much lower repair efficiencies when the number of defects is large. Since ESP does not have 100% repair efficiency, the proposed algorithm is compared with the ISF algorithm such that both algorithms have 100% repair efficiencies.

For comparison of RA times, Figs. 4 and 5 show the number of clock cycles of the BIRA algorithms. It can be seen that the proposed algorithm, which stops when the first solution is confirmed, requires fewer clocks than that of the ISF algorithm. Figure 6 shows the hardware overhead costs of the RA modules. The hardware overhead costs are calculated by equations for storage of the BIRA modules. Among the existing RA algorithms, ESP has the lowest hardware overhead cost. However, the ESP and LRM methods do not have 100% repair efficiencies. The proposed algorithm has the same hardware overhead as the ISF, but it has a lower hardware overhead than the CRESTA and LRM methods as shown in Fig. 6. Table 1 shows the performance comparison results with the conventional algorithms. As shown in Table 1, the proposed algorithm has superior performance. The proposed algorithm

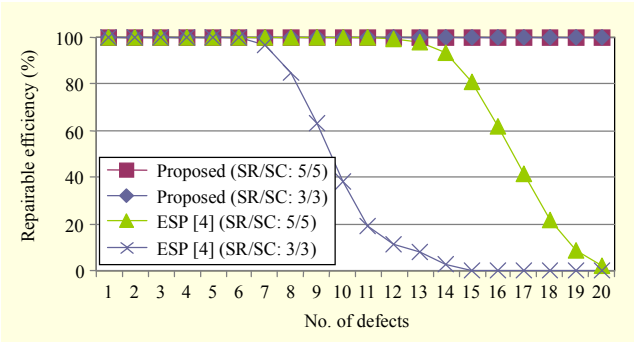

Fig. 3. Comparison of repairable percentages.

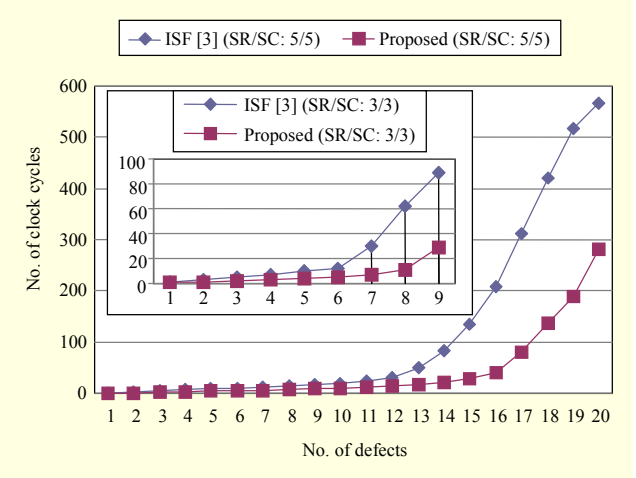

Fig. 4. Comparison of the number of clock cycles (SR/SC: 3/3, 5/5).

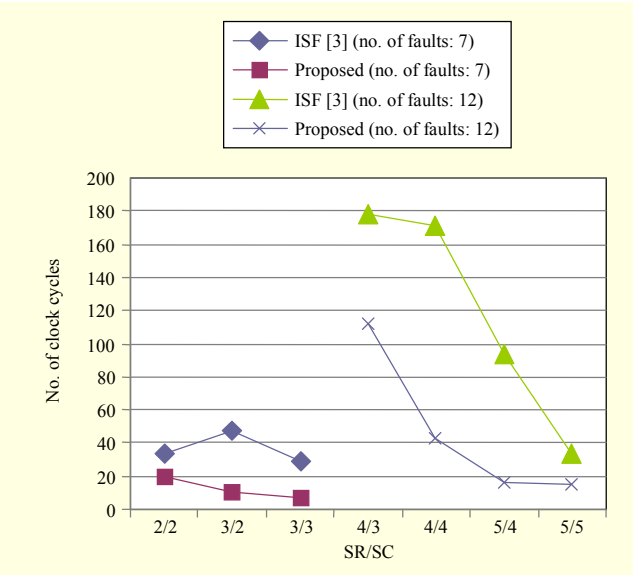

Fig. 5. Comparison of the number of clock cycles.

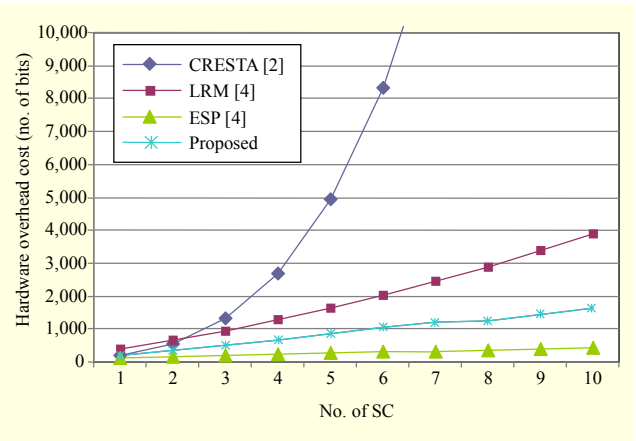

Fig. 6. Comparison of hardware overhead costs incurred by different algorithms (SR: 3).

Table 1. Performance comparison of the proposed algorithm with other BIRA algorithms.

Algorithm	CRESTA [2]	ESP [4]	LRM [4]	ISF [3]	Proposed
RA time	Very short	Short	Short	Medium	Short
Hardware overhead	Very high	Small	Medium	Small	Small
Repair efficiency	100%	<100%	<100%	100%	100%

has 100% repair efficiency, smaller hardware overhead, and a minimized RA time.

V. Conclusion

The main idea of the proposed algorithm is to identify a solution in the shortest possible time through reducing the search space. The minimized binary tree continues to be built until the solution is found from the most probable branch. Therefore, the proposed BIRA algorithm has 100% repair efficiency and minimizes the RA time. According to the experimental results, the proposed algorithm generates solutions with the minimal number of clocks when searching a minimized binary search tree. Therefore, application of the proposed algorithm can result in increased memory manufacturing yield and reliability.

References

[1] Y. Park et al., "An Effective Test and Diagnosis Algorithm for Dual-Port Memories," *ETRI J.*, vol. 30, no. 4, Aug. 2008, pp. 555-564.

[2] T. Kawagoe et al., "A Built-in Self-Repair Analyzer (CRESTA) for Embedded DRAMs," *Proc. Int. Test Conf. (ITC)*, 2000, pp. 567-574.

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

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