A Novel BIRA Method with High Repair Efficiency and Small Hardware Overhead

Myung-Hoon Yang, Hyungjun Cho, Woosik Jeong, and Sungho Kang

ABSTRACT—Built-in redundancy analysis (BIRA) is widely used to enhance the yield of embedded memories. In this letter, a new BIRA method for both high repair efficiency and small hardware overhead is presented. The proposed method performs redundancy analysis operations using the spare mapping registers with a covered fault list. Experimental results demonstrate the superiority of the proposed method compared to previous works.

Keywords—Built-in redundancy analysis (BIRA), embedded memory, yield, repair rate.

I. Introduction

Highly developed deep submicron technology has enabled the implementation of large system-on-chips (SoCs). Also, an SoC includes many large embedded memory cores, and the percentage of memory cores in an SoC is rapidly increasing. Due to the high density of memory devices and embedded memories, various faults are introduced, and this deteriorates the yield. Therefore, to improve the memory yield, faulty cell (line, block) repair has been introduced [1]. The widely used repair scheme uses two-dimensional (2-D) redundancy which adds both spare rows and columns to the memory for repair of detected faults. As memory size increases, the number of spare lines is increased to guarantee the desired yield of the memory devices.

However, for memories with 2-D redundancy, the optimal redundancy allocation problem becomes NP complete [2], [3]. Therefore, some heuristic redundancy analysis (RA) algorithms for built-in RA (BIRA) have been investigated. A simple RA algorithm which uses repair-most (RM) rules was proposed in [2]. Since the RM approach uses fault counters for each row and column, it is difficult to apply this approach to BIRA. The comprehensive real-time exhaustive search test and analysis (CRESTA) scheme was proposed, and it achieves optimal repair efficiency [4]. However, since the number of sub-analyzers drastically increases with the number of spare lines, the CRESTA scheme requires too large a hardware overhead. Although, the number of sub-analyzers can be reduced by serial processing [5], serial processing leads to a very long RA time. The local RM (LRM) and essential spare pivoting (ESP) schemes proposed in [6] incur a small hardware overhead and small storage for failure information. However, they cannot guarantee high repair efficiency when compared with other schemes. *IntelligentSolve* can guarantee optimal repair efficiency without using a bitmap [7]. This method requires a very long search time and content addressable memory (CAM) in order to reduce the search time.

CRESTA and *IntelligentSolve* can guarantee optimal repair efficiency; however, their hardware overhead increases drastically with the number of spare lines. Although ESP can perform RA operations with a small hardware overhead, it cannot guarantee high repair efficiency. In this letter, a new BIRA method with high repair efficiency and a small hardware overhead is proposed. The proposed method performs RA operations using the spare mapping registers with a covered fault list. To reduce the hardware overhead, spare line mapping results are directly updated to spare mapping registers for each detected fault without using a bitmap. The repair efficiency of the proposed method is similar to that of the RM approach, and the hardware overhead of the proposed method is similar to that of ESP. Therefore, the proposed method is a very efficient solution for BIRA implementation.

Manuscript received Jan. 15, 2009; revised Mar. 4, 2009; accepted Apr. 13, 2009.

Myung-Hoon Yang (phone: + 82 2 2123 2775, email: mhyang@soc.yonsei.ac.kr), Hyungjun Cho (email: chj0937@soc.yonsei.ac.kr), Woosik Jeong (email: woosik.jeong@soc.yonsei.ac.kr), and Sungho Kang (phone: + 82 2 2123 2775, email: shkang@yonsei.ac.kr) are with the Department of Electrical & Electronic Engineering, Yonsei University, Seoul, Rep. of Korea

 © 2009 ETRI

II. Proposed BIRA Technique

Conventionally, RA schemes for BIRA require storage elements for $2rc$ (r and c are the numbers of spare rows and columns, respectively). Considering the additional cost for repair using built-in self-repair (BISR), the hardware overhead is critical in practical applications. Since the proposed method uses simple spare mapping registers with a covered fault list, the hardware overhead can be significantly reduced compared with previous methods. Figure 1 shows the structure of the spare mapping registers with 2 spare rows and 2 spare columns. The number of spare row (column) mapping registers equals r (c). If the number of faults in a single row (column) exceeds the number of column (row) spares, the row (column) must be repaired using a spare row (column). Therefore, the maximal numbers of faults of the covered fault list, F_c and F_r, are c and r, respectively. We set F_c and F_r to 2, so hardware implementation was greatly simplified without degradation of the repair efficiency.

Whenever the built-in self-test (BIST) detects a faulty cell, the address of the faulty cell is latched. The BIRA checks whether the faulty cell can be repaired by the currently used spare lines. If both F_0 and F_1 bits of a spare mapping register which covers this fault are set to 1, this spare line is designated a "must-repair" line, and bit M is set to 1. If only the F_0 bit is 1, bit F_1 is set to 1, and F_1_*Col_Addr* or F_1_*Row_Addr* is filled with the column (row) address of the faulty cell. If any used spare lines cannot cover this faulty cell, one of the unused spare lines is allocated for the faulty cell. In this case, for one of the unused spare mapping registers, the U bit is set to 1, and the row (column) address of the faulty cell is recorded at *Row_Addr* or *Col_Addr*. Also, bit F_0 is set to 1, and the column (row) address of the faulty cell is recorded at F_0_*Col_Addr* or F_0_*Row_Addr*. If there is no unused spare line, used spare lines

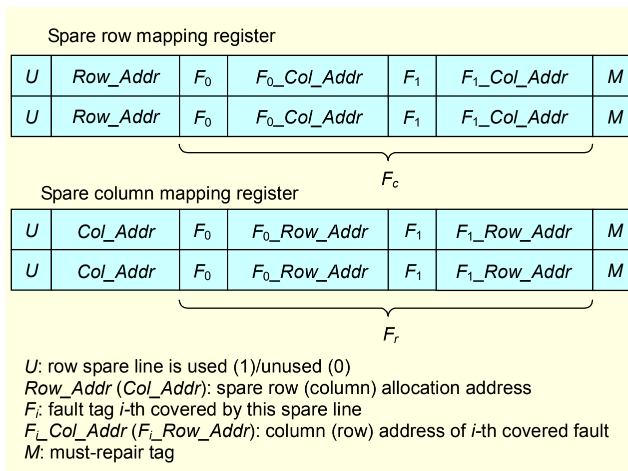

Fig. 1. Structure of the spare mapping registers.

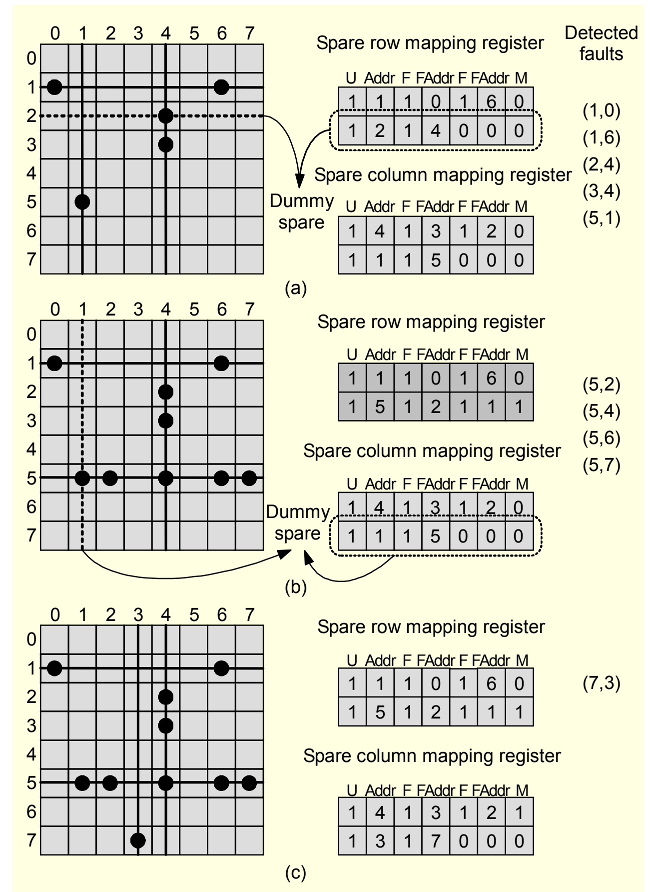

Fig. 2. Example redundancy analysis procedure.

are checked to find a dummy spare line. If all faults in the covered fault list of a spare row (column) are covered by used spare columns (rows), the spare row (column) must be the dummy spare line. The must-repair line should be excluded from the search space for the dummy line. If there is no dummy spare line by this method, the faulty cell can be repaired by swapping row and column spare lines which cover only one faulty cell. If the existent faulty cell covered by these row and column spare lines shares a row or column address of a newly detected faulty cell, these spare lines can be swapped and cover the newly detected faulty cell.

Figure 2 shows an example of an RA procedure of the proposed method. In this example, we assume that the memory size is 8×8, r=c=2, and F_r=F_c=2. After 5 faults are detected, all spare lines are used, and the procedure to find a dummy spare line is performed. As shown in Fig. 2(a), the spare row allocated to address 2 (dotted line) is a dummy spare line. This spare row is allocated to the sixth faulty cell (5, 2). After nine faulty cells are covered, there is no unused spare line as shown in Fig. 2(b). Since the spare column allocated to address 1 is a dummy spare line, this spare column is allocated to the last faulty cell (7, 3) as shown in Fig. 2(c).

III. Experimental Results

In our experiments, we assumed a 1024×1024 memory cell array with $r=c=4$, and $F_r=F_c=2$. To guarantee the effectiveness of the proposed method on various processes, each experiment for the number of randomly injected faults was repeated 10,000 times. Injected faults were scattered with Gaussian distribution so as to introduce various fault types (single faulty cell, faulty row, faulty column, cluster fault). Figure 3 shows the average repair efficiency for each scheme according to the number of faults inserted at random locations. Although the proposed method cannot guarantee optimal repair efficiency, the repair efficiency of the proposed method is similar to that of the RM approach and much higher than that of ESP.

The hardware overhead was estimated using the number of storage cells because each scheme needs storage cells for repair procedures, and the storage cells dominate the silicon area of BIRA circuits [6]. Figure 4 shows the hardware overhead of the proposed scheme compared with that of previous schemes. We assumed that the size of the memory cell array was 1024×1024 bits, with (r, c) ranging from (2, 2) to (6, 6), and $F_r=F_c=2$. The hardware overhead of the proposed method is much smaller than that of CRESTA and *IntelligentSolve* and similar to that of ESP. Also, since the proposed method can be implemented with simple registers (without CAM), the implementation is very simple.

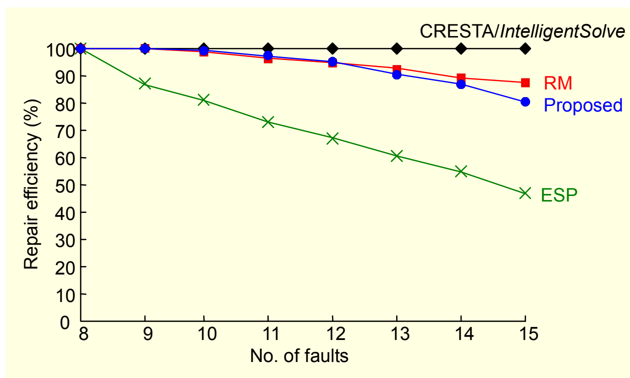

Fig. 3. Comparison of repair efficiency.

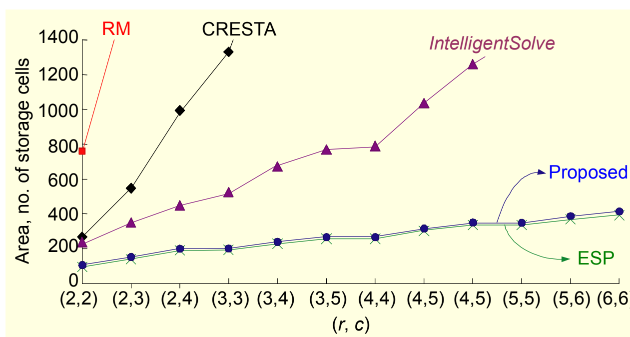

Fig. 4. Comparison of hardware overhead.

The previous BIRA methods have trade-offs between repair efficiency and hardware overhead. As shown in Figs. 3 and 4, the newly proposed method has advantages in both of these areas. The proposed method can achieve almost optimal repair efficiency with a very small hardware overhead. Therefore, the proposed method is a very useful BIRA solution for embedded memories.

IV. Conclusion

The proposed BIRA method with both high repair efficiency and small hardware overhead was presented. To reduce the number of storage elements, spare line mapping results are directly updated to the spare mapping registers for each detected fault. The proposed method is a very efficient BIRA method for embedded memories which require both high repair efficiency and a small hardware overhead. Experimental results show that the proposed method can guarantee high repair efficiency with a very small hardware overhead.

References

[1] R.P. Cenker et al., "A Fault-Tolerant 64K Dynamic Random-Access Memory," *IEEE Trans. Electron Devices*, vol. 26, no. 6, June 1979, pp. 853-860.

[2] J.R. Day, "A Fault-Driven Comprehensive Redundancy Algorithms," *Proc. Int'l Test Conf.*, 1985, pp. 35-44.

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

[4] T. Kawagoe et al., "A Built-in Self-Analyzer (CRESTA) for Embedded DRAMs," *Proc. Int'l Test Conf.*, 2000, pp. 567-574.

[5] S. Shoukourian, V. Vardanian, and Y. Zorian, "An Approach for Evaluation of Redundancy Analysis Algorithm," *Proc. IEEE Memory Technology, Design and Testing Workshop*, 2001, pp. 51-55.

[6] C.-T. Huang, C.-F. Wu, and C.-W. Wu, "Built-in Redundancy Analysis for Memory Yield Improvement," *IEEE Trans. Reliability*, vol. 52, no. 4, Dec. 2003, pp. 386-399.

[7] 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.*, 2007, pp. 91-96.