An Advanced BIRA Using Parallel Sub-analyzers for Embedded Memories

Woosik Jeong, Taewoo Han and Sungho Kang
Department of Electrical and Electronics Engineering
Yonsei University
Seoul, Korea
{woosik.jeong, twhan, and shkang}@yonsei.ac.kr

Abstract—Although many built-in redundancy analysis (BIRA) algorithms which use parallel sub-analyzers have optimal repair rate and a fast analysis speed, they suffer from a large area overhead. To reduce the area overhead, a new BIRA analyzer is proposed which reconstructs the content addressable memory (CAM) structure of the parallel sub-analyzers like a binary searching tree. Experimental results show that the proposed BIRA analyzer achieves 25% reduction of area overhead in case an embedded memory has 4 spares with optimal repair rate and zero analysis speed.

Keywords—built-in self repair (BISR); redundancy analysis (RA); optimal repair; embedded memory; area overhead

I. INTRODUCTION

With the advance of technology, the capacity and density of embedded memories on systems-on-chip (SOC) have been continuously increased while the probability of memory faults has also increased, resulting in yield drop. Recently, built-in redundancy analysis (BIRA) algorithms have been widely used to improve the quality and the yield for SOC by replacing faulty cells with healthy redundant cells. Many BIRA approaches for various spare architectures have been introduced and most redundancy analysis algorithms for BIRA are based on 2D spare architecture using row and column redundancies [1-14]. Optimal redundancy allocation problem becomes NP-complete [4], [12]. From the middle of 1980s, various redundancy analysis algorithms for 2D spare architecture have been developed [1]-[3], [8], [9], [12]-[14]. Among them, CRESTA [1], IntelligentSolveFirst [2], ESP [3], and LRM [3] are most well known BIRA algorithms. There are 3 key features of a BIRA algorithm to reflect its performance: the repair rate, which should be optimal for an improved yield; the area overhead, which should be low to reduce silicon cost; and the analysis time, which should be short for increased productivity. However, the main drawback of CRESTA is the large area overhead due to its parallel sub-analyzers while its analysis time is almost zero [1]. Since IntelligentSolveFirst searches for all possible solution cases with a single redundancy analyzer, its major drawback is the long analysis time [2]. Meanwhile, some other BIRA algorithms such as ESP and LRM have been developed to reduce area overhead but they have to suffer from a loss of repair rate [3].

II. PARALLEL SUB-ANALYZER

A. Observation on Previous Work

To reduce the area overhead of CRESTA, a close observation is required. CRESTA has parallel sub-analyzers and the total number of sub-analyzers is the same as the total number of branches in the binary searching tree. Figure 1a shows an example of a memory block which has 2 row spares and 2 column spares. In this figure, there are 8 faults and their detecting order is shown in Fig. 1b. Only one set of repair solution exists in this example (i.e., row 1, row 3, column 4, and column 6).

Fig. 1 An example of memory block with some faults
Fig. 2 An example of memory block and redundancy analysis by CRESTA analyzer.

addresses invalid. During a test, the parallel sub-analyzers are running every detected faulty address by BIST. These faulty addresses are compared with the previously stored address in the CAM arrays of each sub-analyzer, if each stored address is found to be valid, from top to bottom (e.g., R1 → R2 → C1 → C4 for sub-analyzer #1 in Fig. 2a). Thereafter, the faulty address is stored into the unused upmost CAM array when there is no same row or column address. If there is no more unused CAM array for any sub-analyzer to store a new faulty address, the sub-analyzer is turned out to be un-repairable. The compare-then-store action is executed for each sub-analyzer, concurrently. In Fig. 2a, Unsuc represents the analysis result of each sub-analyzer. Initially, the value of Unsuc is ‘L’ but it is set to ‘H’ if the sub-analyzer failed to get a proper repair solution. According to Fig. 2a, the memory is repairable and a set of address values stored in sub-analyzer #1 is the proper repair solution (i.e., row 1, row 3, column 4, and column 6) while the other sub-analyzers are failed.

Meanwhile, in this example, some groups of CAM arrays such as {R11, R21, and R31}, {C41, C51, and C61}, {C22 and C32} and {R42 and R52} store the same address values for each group as row 1, column 0, column 1, and row 3, respectively. If it is possible that a group of CAM arrays may be substituted for a representative CAM array, this may be a good opportunity to reduce the area overhead of CRESTA by removing any repetitive CAM arrays. This is the primary focus of our newly proposed BIRA, R-CRESTA.

B. Proposed Parallel Sub-analyzer

Before removing any repetitive CAM arrays, the connection of input-output signals between the redundant CAM array and the descendent must be verified. In Fig. 2a, Rij (Cij) indicates the jth row (column) CAM array of ith sub-analyzer to store row (column) address. Each sub-analyzer consists of 4 CAM arrays in this example. Each CAM array has three kinds of output; a stored address (ADDRij), an enable flag (Eij), and a match flag (Mij) as shown in Fig. 2b. A stored address represents the previously stored address of a CAM array. An enable flag represents the validity of the stored address. A match flag represents whether the stored address is the same as a detected faulty address. These three outputs of CAM array Rij (Cij) can be expressed as shown below.

\[
ADDR_{ij} = E_{ij} \cdot \bar{M}_{ij} \cdot \text{faulty address} + ADDR_0
\]

\[
E_{ij} = E_{ij-1} \cdot \bar{E}_{ij} \cdot M_{ij-1} + E_{ij}
\]

\[
M_{ij} = E_{ij} \cdot \text{compare}\{\text{faulty address}, ADDR_{ij}\} + M_{ij-1}
\]

If a is equal to b, compare (a,b) in Equation (3) returns 1 else it returns 0.

Operations used in the above Equations such as \(\cdot\) and \(\pm\) imply logical AND and logical OR, respectively. In Equation (1) and (3), a faulty address indicates a newly detected address from the BIST which is not a pair of addresses but a row (column) address for the row (column) CAM array. In Equation (1), the faulty address is stored into a CAM array when its previous enable flag is 1 (i.e., \(E_{ij-1}=1\)) and its own enable flag is 0 (i.e., \(E_{ij}=0\)). This means that the CAM array is the unused upmost one. In addition, the previous match flag must be 0 (i.e., \(M_{ij-1}=0\)), which means that there is no same address from the first CAM array to the (j-1)th CAM array. According to stored addresses in CAM arrays in Fig. 2a, a group of CAM arrays such as {R11, R21, and R31} store the same row address when the first faulty address (i.e., (1,0)) is detected. Because of this, the previous enable flag and the previous match flag of the first CAM array of all sub-analyzers are fixed to 1 and 0, respectively. Then the enable flags of
{R_{11}, R_{21}, and R_{31}} have the same value (i.e., E_{11}=E_{21}=E_{31}=1). When the second fault (i.e., (3,1)) is detected, the match flags of {R_{11}, R_{21}, and R_{31}} have the same value (i.e., M_{11}=M_{21}=M_{31}=0) since the stored addresses of {R_{11}, R_{21}, and R_{31}} are the same. Therefore, some groups of CAM arrays have the same outputs when they have the same type (i.e., row CAM or column CAM) and the same sequence from the first CAM array for each sub-analyzer (e.g., R_{21} vs. R_{31}).

This means that some redundant CAM arrays such as {R_{21}, and R_{31}} can be removed from {R_{11}, R_{21}, and R_{31}} and outputs from the representative CAM array R_{11} can be simply broadcasted to the input signals of their descendents (e.g., R_{12}, C_{22}, and C_{32}) without any additional circuitry to compensate the removed CAM arrays. Figure 3 shows the proposed structure of parallel sub-analyzers for the example depicted in Figure 1a. In this case, R_{31}, C_{41}, C_{51}, C_{32}, and R_{42} were removed and input-output signals were connected as shown in Figure 3.

III. EXPERIMENTAL RESULTS

Normalized repair rate represents the ability of a RA algorithm to find a correct repair solution and it was introduced in [3]. The Definition of the normalized repair rate is as follows:

\[
\text{normalized repair rate} = \frac{\# \text{ of repaired chips}}{\# \text{ of repairable chips}}.
\]

In this definition, the repairable chip means a chip is able to be repairable and the repaired chip means a chip is turned out to be repairable by the result of a RA algorithm. Optimal repair rate is used when the normalized repair rate is 100%. To verify the performance of the proposed BIRA, comparison of the repair rate of various BIRAs is shown in Figure 4. We developed a C-language based simulation tool named RepairSim which generates 1000 different cases of random faults for each x-axis values of Figure 4. In Fig. 4, the proposed BIRA, CRESTA, and IntelligentSolveFirst achieve a 100% normalized repair rate except for ESP.

Figure 5 shows the comparison of the clock cycles of RA analysis of various BIRAs. CRESTA and the proposed BIRA execute redundancy analysis whenever a fault is detected by the BIST during test algorithms running. The redundancy analysis is also completed at the same time of finishing all test algorithms. IntelligentSolveFirst requires a long analysis time because it has single redundancy analyzer and recalculation after finishing test algorithms is inevitable to find a proper repair solution.

Storage requirement, the total number of CAMs, is the dominant part of the area overhead of the entire BIRA [3]. Figure 6 shows the comparison of the storage requirement ratio which is calculated as follows:

\[
\text{Storage requirement ratio} = \frac{\# \text{ of CAMs of a BIRA}}{\# \text{ of CAMs of CRESTA}}.
\]

ESP has the smaller area overhead than CRESTA for most cases but the repair rate of ESP is not optimal. Except for ESP, CRESTA, IntelligentSolveFirst, and the proposed BIRA have optimal repair rate so the comparison of storage requirement ratio is is shown in Figure 6.
ratio was executed only for these three BIRAs. The proposed BIRA shows the lower storage requirement ratio than CRESTA when the number of spares is greater than 3. Four spares (e.g., 2 row spares and 2 column spares) are sufficient for many cases because the size of most embedded memories is still small. The storage requirement of the proposed BIRA shows only 75% of CRESTA in case a memory block has 4 spares. Furthermore, the larger number of spares is used, the lower area overhead is required on the proposed BIRA compared with CRESTA. The proposed BIRA also shows the lowest storage requirement for a memory block has 4 spares compared with both CRESTA and IntelligentSolveFirst.

According to experimental results on repair rate, analysis speed and storage requirement, the proposed analyzer successfully reduces the area overhead of CRESTA by reconstructing the CAM structure of parallel sub-analyzers while it still achieves optimal repair rate and zero analysis time.

IV. CONCLUSION

The proposed BIRA, achieved a reduction of the area overhead against the original CRESTA algorithm while maintaining both optimal repair rate and zero analysis time. The reduction of area overhead is successfully accomplished by removing redundant CAM arrays from the parallel sub-analyzers without addition of any other circuitry. Therefore, it can be a valuable solution for small memory systems such as the embedded memory on SOC.

ACKNOWLEDGMENT

The authors would like to thank all members of Computer Systems and the Reliable SOC Laboratory from Yonsei University, Seoul, Korea, for their insightful feedback and comments on this research. The authors would like to appreciate CAD tool support from IC Design Education Center (IDEC).

REFERENCES