High-efficiency memory BISR with two serial RA stages using spare memories

I. Kang, W. Jeong and S. Kang

As technology has become more advanced, the density of memory has increased greatly. This development has led to the need for a high-efficiency redundancy analysis (RA) algorithm to improve yield rate. Presented is a new methodology that can achieve high-efficiency repair against faults in memory. Experimental results show that the proposed built-in self-repair (BISR) method performs well.

Introduction: Owing to the development of memory design and process technology, the production of high density memories has become a large-scale industry. Moreover, the progress of system-on-a-chip (SoC) technology has increased the use of embedded memories. Since these memories require complicated designs and accurate manufacturing processes, there are possibilities for increased defects. Therefore, to analyse the defects, repair them, and fix the problems in the manufacturing process, both memory testing using built-in self-test (BIST) methods and memory repair applying a BISR circuit have recently attracted attention.

Among numerous research studies concerning memory BISR methods, there have been several successful systems including CRESTA [1], LRM [2], and ESP [2]. The CRESTA detects repairable memories with 100% accuracy under the condition of given redundancies. It analyses all possible methods, which can be applied by spare rows and columns, with parallel sub-analysers. However, CRESTA has a huge hardware overhead when the number of spare memories is increased. The LRM and ESP methods in [2] are superior BISR schemes because of their low hardware overhead and the elimination of failure bitmap. However, they have the weakness that sometimes they cannot repair memory that is repairable. In [3], the authors give fundamental guidelines for an efficient repair process. However, it needs long redundancy analysis (RA) time and a failure bitmap. The disadvantages of previous works cause ineffective waste in the RA process. To reduce RA time, decrease hardware overhead, and enlarge the normalised repair rate, we propose a new memory BISR strategy applying two serial RA stages.

Notation: To explain the proposed BISR scheme, it is necessary to define several terms. First, we can classify faults into two types according to the number of faults at the same address. Fig. 1 shows examples of the two types of fault.

- Fault (not-must): Faults that occur in the same row (column) and can be repaired not necessarily using a spare row (column). Fault (not-must) includes a fault that exists alone in a row or column. All fault addresses of this type must be saved in the fault address buffer.
- Fault (must-repair): Faults that occur in the same row (column) and must be repaired using a spare row (column). Fault (must-repair) is created when the number of faults in the same row (column) address is more than the number of available spare columns (rows). For storing the fault addresses of this type, it is required to save only one address that is the same row (column) address. The saved address will be called the ‘must-repair-address’.

There are two rules for determining the minimum size of the fault address buffer. To describe the rules, it is assumed that the memory has M spare rows and N spare columns.

- Rule 1: If the \([M + N + 1]\)-order fault, which does not have any repeated row and column addresses, occurs, the memory cannot be repaired.
- Rule 2: If there are \([2 \times M \times N]\) buffers, we can store all valid fault addresses of the targeted memory that can be repaired.

Proposed BISR architecture: The proposed BISR method classifies RA algorithms into two stages to minimise the inefficiency of conventional repair methods. The first stage is named MUST RA and the second stage is named MOST RA. For the desired operation, the proposed BISR has two store spaces: a space for storing the number of faults and another space (called a ‘fault address buffer’) for saving the addresses. The fault, which is detected by the memory BIST process, has to be written at the space for saving the number of faults and then should be saved at the fault address buffer according to the definition of the faults. These operations are performed in real time with the memory BIST. The MUST RA repairs the must-repair-address, called the Fault(must-repair) type, which needs to be repaired by the spare row (column). During the memory BIST process, if there is a must-repair-address at the row or column address, it is possible to immediately send the information related to the repair process. Throughout this process, we can reduce the total RA time and cost. This is one of the main advantages of the proposed method. After the memory BIST process, the MUST RA based on the repair-most rule [3] starts to repair the faults that have the highest priority according to the number of faults. The address that contains more faults has a higher priority.

Fig. 2 shows the implementation of the RA stages in time. At the end of the MOST RA, the entire repair process is completed.

As mentioned previously, for the implementation of the proposed BISR, we have to set the spaces for saving the number of faults and the fault address buffer. In particular, for storing the number of faults, we utilise the spare memories. Through the use of the spare memories, we can largely reduce the area overhead without loss of performance. For using the spare memories as a resource to save the number of faults at the same row (column) address, there should be no faults in the spare memories. From this consideration, the architecture that we suggest in this Letter will be useful for an embedded SRAM that has small redundancies, which have a small probability of being detected faults owing to the relatively small capacity. To save the number of faults, the proposed BISR encodes the number into binary. Fig. 3 depicts how the number of faults is recorded. The spare memories have to be organised with at least two spare rows and two spare columns to use the encoding technique. The procedure used to update the number of faults is executed simultaneously with detection of the faults. If the number of faults in a row (column) address is more than the number of spare columns (rows), the faults are determined Fault (must-repair), and the address is defined as the must-repair-address.
When the address that contains the faults of Fault (must-repair) is detected, the must-repair-address is determined and it will be repaired during the MUST RA.

When the fault is discovered, it will be counted and the address of the fault will be sent to the fault address buffer. Because of the difficulty in achieving optimal performance with the fault counts only, the fault address buffer must inevitably be set. Through the RA process, the proposed BISR module counts the number of faults by the memory BIST, updates the numbers with the information from the fault address buffer, and repairs the faults with these data. Also, the remapping addresses are sent to the fault address buffer. The fault address buffer consists of row and column address bits and control bits. The row and column address bits store the fault addresses and the control bits contain the enable bit and s-bit. When the enable bit is set, the corresponding fault address in the buffer is available. When the s-bit is set, the remapping address in the fault address buffer is available. Using these rules, we can find the minimum size of the fault address buffer to repair the memories that can actually be fixed. According to these rules, when the buffer has no space to store the valid address, the memory is determined to be un-repairable if the fault that has to be written in the buffer is detected.

**Experimental results:** Figs. 4 and 5 present the experimental results of the proposed BISR scheme compared to conventional BISR approaches. The size of the targeted memories is 1024 x 256, and the faults for the experiments are scattered with a Gaussian distribution. In particular, the repair rate in this simulation was normalised to the repair rate of the optimal algorithm, the CRESTA method, which can detect 100% of the repairable memory. The area overhead and the normalised repair rate are shown in Figs. 4 and 5, respectively.

**Comparison:** The proposed BISR architecture has a low area overhead compared to other methods. Moreover, the proposed BISR scheme performs the highest repair rate except for CRESTA, which has a drastically large area overhead. Table 1 indicates comparison results with the conventional algorithms. In Table 1, we can find that the proposed method has superior performance. The proposed BISR strategy has the shortest RA time, the smaller hardware overhead and the higher normalised repair rate compared to existing BISR algorithms.

**Table 1: Performance comparison**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RA time</td>
<td>Very short</td>
<td>Short</td>
<td>Short</td>
<td>Long</td>
<td>Short</td>
</tr>
<tr>
<td>Hardware overhead</td>
<td>Very high</td>
<td>Small</td>
<td>Medium</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>Failure bitmap</td>
<td>Useless</td>
<td>Useless</td>
<td>Useless</td>
<td>Required</td>
<td>Useless</td>
</tr>
<tr>
<td>Normalised repair rate</td>
<td>1.0000</td>
<td>Less than 0.9995</td>
<td>Less than 0.9995</td>
<td>More than 0.9995</td>
<td>More than 0.9995</td>
</tr>
</tbody>
</table>

**Conclusion:** Presented is a novel BISR algorithm that can achieve high-performance repair. The proposed BISR strategy applies two serial RA stages to repair faults in memories. Through the analysis of experimental results, we can see that the proposed BISR method performs better. Consequently, the proposed BISR method exhibits excellent performance compared to the performance of conventional methods.

© The Institution of Engineering and Technology 2008
20 December 2007
Electronics Letters online no: 20083611
doi: 10.1049/el:20083611

I. Kang, W. Jeong and S. Kang (Department of Electrical and Electronic Engineering, Yonsei University, Shinchon-Dong 134 Seodaemun-Gu, Seoul 120-749, Korea)
E-mail: shkang@yonsei.ac.kr

**References**