Abstract

In general, the external automatic test equipment (ATE) is used to progress a redundancy analysis (RA) process in industrial semiconductor memories. However, many researches have pointed out that a high price of the external ATE can be a huge burden on testing costs. This paper presents a new concept which reduces the failure bitmap size in the external ATE without any extra hardware overhead. Despite the reduction of the failure bitmap size, converting some parts of the faulty information to a partial solution search tree prevents unwanted yield drops.

Keywords: yield, testing costs, redundancy analysis (RA), bitmap and automatic test equipment (ATE)

Introduction

To gain a reasonable manufacturing yield, one thing that cannot be missed in the memory repair area is a redundancy analysis (RA) technique. At the beginning, all the faulty information is transferred to the external automatic test equipment (ATE) and stored in the failure bitmap. Then based on the properly selected RA algorithm, solution addresses are collected from the external ATE. A faulty memory is then repaired by allocating fault-free spare lines to solution addresses. Various RA algorithms have been studied steadily. Repair-most (RM) is a simple algorithm that allocates spare cells from the most faulty line [1]. Intelligent-solve (IS) and intelligent-solve-first (ISF) use a compressed binary search tree with backtracking processes [2].

The most crucial problem of using the external ATE is that the cost of the device is way too expensive. Various research topics have been studied to solve this problem. In [3], the author says that the external ATE can be substituted with built-in self repair (BISR) which consists of built-in self test (BIST) and built-in redundancy analysis (BIRA) by performing the functions of the ATE in the chip itself. Although this previous work economically relieves the burden of the external ATE, preparing a space in the chip for the extra built-in logics can be a trouble to chip designers.

This paper presents a simple concept which can reduce the failure bitmap size using a partial solution search tree based on an exhaustive search RA algorithm. Because a RA algorithm is inserted into the ATE in a form of software, any extra logic is not required. The rest of this paper shows details of the proposed idea.

Proposed Idea

The key point of the proposed idea is converting some faulty addresses to a binary search tree so that some part of the failure bitmap can be omitted. Fig. 1 is an example of an 8×8 faulty memory block with 2 row spares and 2 column spares. This memory block is divided into four quadrants. First and third quadrants of the failure bitmap, shaded areas on Fig. 1, are the target areas to be omitted.

Faults are sorted into three groups. Faults which are placed in first or third quadrants belong to group 1. Because first and third quadrants of the failure bitmap will be omitted, group 1’s faulty addresses should be stored in a different way. Whenever faults in group 1 are detected, a partial solution search tree is updated. As a result of the fault collection phase, a partial solution search tree which contains all kinds of solutions that can cover group 1’s faults is completed. The faulty information of group 1 is no longer needed because each branch of a partial solution search tree can be a solution of group 1. After fault collection, faults in second or fourth quadrants are split into two types. Faults which cannot be covered by the information of a partial solution search tree belong to group 2. Remaining faults and then belong to group 3.

The proposed idea uses a simple RA process. Since a partial solution search tree cannot cover group 2’s faults, another binary solution search tree is needed for group 2. Then, the ATE finds a solution between the combination of two solution search trees using the exhaustive search. There are some conditions that a solution must be satisfied. The number of used spare lines should not exceed that of prepared spare lines. And if the combination of two solution search trees cannot cover all group 3’s faults, remaining spare lines should cover group 3. Fig. 2 describes the entire process of the proposed idea.
Fig. 2 Solution search process of Fig. 1. (a) Fault grouping. (b) Partial solution search tree of group 1. (c) Solution search tree of group 2.

Fig. 2(a) is a consequence of the fault grouping process. Faulty cells on (3, 6) and (5, 0) are included in group 1 since they are placed in first or third quadrants of the failure bitmap. Based on group 1, the combination of (row 3, column 6) and (row 5, column 0) becomes a partial solution search tree which is shown in Fig. 2(b). The dotted line on Fig. 2(a) denotes the paths that might be covered by a partial solution search tree. Faulty cells on (1, 1), (6, 5) and (7, 5) are then gathered in group 2. Fig. 2(c) shows a solution search tree for group 2. The rest of faulty cells belong to group 3. After all solution search trees are constructed, the ATE finds a final solution that meets the conditions. At first, the ATE checks all the candidates which use fewer spare lines than prepared spare lines. There are three candidates which are depicted with solid lines in Fig. 2(b) and (c). In this case, all the candidates, (R3, R5, C1, C5), (R3, C0, R1, C5) and (C6, R5, R1, C5), use 2 row spares and 2 column spares. Because there is no extra spare lines, only (R3, R5, C1, C5) can cover group 3’s faults among the candidates. So a final solution for this example is row three, row five, column one and column 5. It is depicted with bold lines in Fig. 2(b) and (c).

Another strength of the proposed idea is that the rate of reduction can be adjustable. In the above example, only 50% of memory cells are omitted. But the failure bitmap size can be reduced further while core cells are maintained. Any cells can be selected to core cells but the whole set of core cells should contain every address in a memory block at least one time. However, the more the bitmap area is saved, the more the analysis time is spent. So the rate of reduction should be carefully chosen to make the best outcome. Fig. 3 is one kind of the reduction process. For better visual image, diagonal cells are selected to core cells. Every row(or column) address can be covered by diagonal cells. While cells on (0, 0) to (7, 7) are maintained, shaded areas can be freely selected for saving the bitmap size.

Fig. 3 Example of the additional reduction process of the failure bitmap

Fig. 4 Comparison of test time and repair rate (Row / Column spares = 5 / 5).

Experimental Results

In the experiments, a 1024×1024 memory array is assumed. Fig. 4 shows test time and repair rate when a memory has 5 row and column spares. Each RA algorithm was repeated 10,000 times with the faults using polya-eggenberger distribution. Though RM has the smallest test time, its repair rate is quite low. Unlike with RM, ISF can achieve 100% repair rate. However, both previous algorithms should use the full bitmap. Otherwise, a graph of the proposed algorithm in Fig. 4 uses the completely compressed failure bitmap. This result means that the proposed idea greatly lessens the burden of the memory size and maintains 100% repair rate at the same time by spending some additional analysis time for RA.

Conclusion

A novel method which reduces the failure bitmap size in the external ATE by changing some faulty information to a partial solution search tree was proposed in this paper. At first, target cells are selected and faults are classified into three groups. And then, using a generated partial solution search tree, an exhaustive search algorithm is applied to the RA phase. Eventhough it took a little more RA time, effective results of reduction were gained without additional built-in logics. This idea leads semiconductor industries to the conclusion that the low cost ATE is very helpful to save the overall cost of test.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012R1A2A1A03006255).

References