Integration, the VLSI Journal 62 (2018) 190–204

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

Enhancement of fault collection for embedded RAM redundancy analysis considering intersection and orphan faults

Štefan Krištofík [a,b,*], Peter Malík [a]

[a] Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, 845 07, Bratislava, Slovakia
[b] Institute of Computer Engineering and Applied Informatics, Institute of Informatics, Slovak University of Technology, Ilkovičova 2, 842 16, Bratislava, Slovakia

ARTICLE INFO

Keywords:
Built-in self-repair
Fault collection
Redundancy analysis
Memory repair
Yield improvement

ABSTRACT

In current semiconductor manufacturing processes, embedded memory yield is often improved by including fault tolerance techniques such as built-in self-test and self-repair. The improvement depends on the effectiveness of the memory repair algorithm. One of the existing state-of-art memory repair algorithms with high repair effectiveness is Selected fail count comparison. In this paper, further enhancement of this algorithm is proposed to increase its repair rate by considering special fault types during fault collection such as intersection and orphan faults. For 1 Mb memories with up to 10 spare rows and columns, the experimental results show a significant memory repair rate improvement of up to 5% over the original algorithm at a small area overhead cost averaging at 3,7%. Our results also show that the area overhead can be reduced by using an equal amounts of spare rows and columns, in which case it is negligible.

1. Introduction

Scaling semiconductor memory process technology below 20 nm is becoming increasingly difficult. Transistor wear out effects increase the frequency of stuck-open or stuck-on faults. Failure detection, analysis and repair during the lifetime for memories is necessary. In many current MPU and SoC designs, more than 80% of transistors are consumed by embedded Flash or DRAM memories [1]. In addition, scaling the density has a negative impact on memory yield. The incorporation of fault tolerance techniques to test and repair memories is essential to enhance embedded memory production yield.

Built-in self-repair (BISR) for memories is a well known and established fault tolerance technique based on hardware soft redundancy and memory address reconfiguration. The faulty memory cells are replaced by spare ones. At each startup, the reconfiguration for a memory (repair solution) is calculated, stored and all incoming faulty cell addresses are translated into spare addresses according to the stored solution. A typical BISR architecture consists of three basic modules. The built-in self-test (BIST) module detects and localizes faulty cells and sends fault information to the built-in redundancy analysis (BIRA) module which based on this information calculates the repair solution and stores it in the address reconfiguration module.

The key component of BIRA is the redundancy analysis algorithm (RA). It is responsible for calculating the repair solution. RA typically has two phases [2]: fault collection (FC) and spare allocation (SA). During the FC phase, fault information is collected and stored using the fault information storing mechanism. In the SA phase, spares are allocated to replace faulty cells based on the stored information. The block executing the spare allocation (SA) phase is typically referred to as the analyzer (i.e., a circuit performing the redundancy analysis). The overall memory repair effectiveness of RAs depends on the quality of their fault information storing mechanisms. RAs with a complex FC scheme generally have better repair rate (to be defined in Section 2) than RAs with a simple one.

Many FC schemes were proposed during past three decades. This work aims to improve the quality of SFCC [2], which is a state-of-art memory repair algorithm for embedded RAMs in SoC designs. Its FC scheme has been very successful in the recent years, having been utilized to repair traditional 2D memories [3], block-based 2D memories [4] and also 3D stacked memories [5]. More specifically, this work aims to improve the repair rate of SFCC by adding three enhancements to its FC scheme. The main principle behind the enhancements is to consider special types of faults during fault collection: intersection and orphan faults. Experimental results show that a significant repair rate improve-

* Corresponding author. Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, 845 07, Bratislava, Slovakia.
E-mail addresses: stefan.kristofik@stuba.sk (Š. Krištofík), p.malik@savba.sk (P. Malík).

https://doi.org/10.1016/j.vlsi.2018.02.015
Received 29 December 2016; Received in revised form 10 December 2017; Accepted 28 February 2018
Available online 13 March 2018

ment over the original scheme of up to 5% for 1 Mb memories with up to 10 spare rows and columns can be achieved at a small area overhead cost averaging at 3,7%. Higher repair rate increases memory reliability and manufacturing yield and decreases production costs.

The rest of the paper is organized as follows. RA characteristics are briefly summarized in Section 2. A survey of various existing FC schemes is also included. Section 3 is dedicated to the description of the state-of-art FC scheme. The proposed enhancements are described in Section 4. Experimental results are shown in Section 5 and Section 6 concludes the paper.

2. Related work

Many RAs were proposed during past three decades with various FC schemes. The quality of RAs is typically evaluated by four parameters [2]:

- repair rate – the percentage of all faulty memories the algorithm is able to repair,
- area overhead – the size of the area needed to implement the algorithm on chip,
- repair time,
- the ability to find the optimal repair solution – the solution using the least possible number of spares.

The ideal theoretical RA would be able to guarantee finding the optimal repair solution while having 100% repair rate, zero area overhead and repair time. Existing RAs try to find a compromise between the two most important parameters: repair rate and area overhead.

Based on the way RAs execute their FC and SA phases, they can be divided into static (sequential execution), dynamic (parallel execution) and hybrid (combination of the previous two). In modern BISR applications, only dynamic and hybrid RAs are feasible solutions because static RAs incur too much area overhead on chip to be feasible and have long repair times [2].

There are many aspects to consider prior to RA on-chip implementation [4]. This paper focuses on fault information storing mechanisms. Existing fault storing mechanisms can be divided into three categories: no fault storing, fault bitmaps and CAM memories.

2.1. FC schemes

Some earlier simple dynamic RAs do not use any form of fault storing. Spares are allocated for faults immediately after fault detection. Examples include all pseudo-random spare allocation strategies where spares are allocated according to some predetermined order, e.g., row-first (all rows first, then all columns), column-first (all columns first, then all rows) and balanced (alternating rows and columns) [6]. RAs that do not use any form of fault storing are simple, fast and incur low area overhead. However, spare allocation performed immediately after fault detection may often lead to low repair rates. Such hasty allocation decisions may not be the most effective way to allocate spares for memories. Therefore, these RAs can not guarantee finding optimal repair solutions. They are best used in cases where low memory fault density is expected.

The other approach is to use fault bitmaps for storing fault information. A fault bitmap represents a mapping of memory cells storing the information whether the cells are faulty or not. There are three types of fault bitmaps:

- Complete bitmap – a 1:1 mapping, i.e., the bitmap is the same size as the memory.
- Compressed bitmap – a small sized, reduced form of the complete bitmap.
- Maximal-sized bitmap – its size is set to a minimal value that can still guarantee finding the optimal memory repair solution.

Complete bitmaps are used by some earlier static RAs and are stored off-chip in the memory of the external tester. After the FC phase is finished, the bitmap contains all fault information. Next, the SA phase is executed on the tester and the repair solution is calculated by software based on the information in the bitmap. Examples of RAs using complete bitmaps include Repair-Most (RM) [7] (repair the row or column containing the most faults) and the strategies utilizing either comprehensive search trees [8] or branch-and-bound algorithms [9] in order to traverse the space of possible repair solutions and to find the optimal one. RAs that use a complete bitmap are more complex and take more time than RAs without fault storing. They also incur very high area overhead. Therefore, they are not feasible in BISR solutions. However, their repair rates are very high and they do guarantee finding optimal memory repair solutions. They can be implemented easily in software and are best used with external test equipment or as a benchmark to evaluate repair rates of other built-in RAs.

The compressed local bitmap was proposed to reduce high area overhead incurred by complete fault bitmaps [10]. Because its size is usually very small, it can not contain the same amount of fault information as the complete bitmap. In case the compressed bitmap is full, it needs to be emptied before it can be used again, i.e., the SA phase is executed based on the information stored in the full bitmap. This may lead to situations where some fault information is not accounted for during SA, because it could not be stored in the full bitmap. This in turn will lead to decreasing in repair rates. Some examples of RAs using compressed bitmaps are mentioned next. Local Repair-Most (LRM) first proposed the compressed local bitmap [10]. Extended LRM in Ref. [11] is an extension of LRM for block memories (the differences between traditional and block-based memories are explained in Ref. [12]). The 3D repair scheme proposes to utilize spare IO's in addition to traditional rows and columns [13]. The Range Checking First Algorithm (RCFA) divides the bitmap virtually into two parts [14]. All mentioned examples are hybrid RAs. RAs that use a compressed bitmap are similar to RAs that use a complete bitmap in terms of complexity and repair speed, but the reduced bitmap size allows them to be used in built-in solutions. However, the bitmap size reduction comes at the cost of decreased repair rates.

The maximal-size bitmap was proposed in Ref. [15] to cope with the disadvantages of complete and compressed fault bitmaps. Instead of keeping the size of the bitmap as small as possible, the size is calculated based on the numbers of used spares before on-chip implementation so that the algorithm can guarantee finding the optimal memory repair solution. RAs that use a maximal-size fault bitmap have higher repair rates than RAs using a compressed bitmap while still being feasible for built-in solutions. They also do guarantee finding optimal memory repair solutions. However, the size of the bitmap scales steeply with the number of spares, e.g., for 5 row and 5 column spares, which are the typical numbers of spares, the bitmap size would be 1,2 Kb. In addition to the fault bitmap itself, RA needs an additional storage elements, such as row and column address registers, fault counters and other logic, to keep various auxiliary information. This further increases the area overhead. Published RAs with this type of bitmap are hybrid.

The third approach is to use CAM memories for storing fault information. The aim is to completely eliminate the need for fault bitmaps and all associated circuitry such as row and column address registers, fault counters and other auxiliary registers and logic [15]. Instead, the fault information is stored in a group of small CAM memories which offer all standard memory operations. In addition, they have the advantage of a very quick search operation [16]. CAMs are periodically searched through and the stored data are frequently changed and updated during RA. Spares are allocated based on the contents of CAMs. Some examples of RAs using CAMs are listed next. Essential Spare Pivoting (ESP) allocates spares to repair faulty rows and columns immediately when the second fault in them has been detected [10]. Modified ESP (MESP) is an extension of ESP for block memories [12]. Comprehensive Real-time Exhaustive Search Test and Analysis (CRESTA) sacri-