# Memory repair logic sharing techniques and their impact on yield

Benoit Nadeau-Dostie
Mentor, A Siemens Business
Ottawa, Canada
benoit_nadeau-dostie@mentor.com

Luc Romain
Mentor, A Siemens Business
Ottawa, Canada
luc_romain@mentor.com

2020 IEEE International Test Conference (ITC) | 978-1-7281-9113-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/ITC44778.2020.9325280

***Abstract*— Techniques for sharing memory repair logic amongst memories are described. The techniques allows reducing silicon area and loading time of repair information upon power up. The impact on yield is predicted using two different methods based on defect density and clustering or past silicon experience.**

***Keywords*— Memory repair, silicon area, repair time, yield**

## I. Introduction

The amount of memory embedded in integrated circuits continuously increases. In [1], it is reported that as much as 400Mbits had been observed at 28nm and 500Mbits at 16nm. This memory is distributed over thousands and even tens of thousands of memory instances depending on the application. The number of bits per instance does not necessarily increase. It is the number of instances which tends to increase in order to achieve parallel processing. For example, increasing the number of cores in a processor chip or increasing the number of channels of a telecommunications chip.

Memory repair has become essential to achieve sufficient yield on circuits with such high memory content. However, only a small percentage of memories actually need to be repaired on any one chip. This fact allows to significantly compress the memory repair information stored in a Non-Volatile Memory (NVM) as shown in [2] and [3], for example.

In [4], sharing of the repair analysis logic calculating a repair solution for each memory is described to reduce area. However, test time is not reduced because memories sharing the repair analysis logic need to be tested one at a time. Also, there is no reduction of the power up time since the number of repair registers to be loaded is not reduced. Authors in [5] address the first limitation and explain in more detail how the repair analysis logic can be shared for memories of different dimensions and different physical address mappings. This is done by using an "align and merge" logic module allowing the repair analysis logic to calculate a solution in a unified repair format. Again here, the number repair registers is not reduced and the area of the repair analysis logic increases linearly with the number of memories tested in parallel.

The purpose of our work is to further reduce the amount of repair logic and the time required to load repair registers upon power-up. We propose to achieve both goals by merging failure information coming from different memories or segments of memories tested to calculate a common repair solution which can be broadcasted. A method to limit the potential yield loss resulting from using a common repair solution which might not be optimal is also proposed.

Section II describes the hardware modifications required to share the repair logic. Section III analyzes in detail the potential impact on yield followed by some conclusions.

## II. Repair Logic Optimization

Each Memory BIST controller is typically shared by several memories. Tens or even hundreds of memories can be tested by a single controller. These memories can be different in term of sizes, port configurations (single/multi-port), physical organization (with/without column multiplexing) and repair method (row-only, column-only, row and column). Access to these memories can be direct or indirect through a shared bus such as in [6]. Memories can be tested in parallel, in series or a combination of the two. All these aspects are considered in our proposed solution for sharing the repair logic.

The proposed method does not require modifying existing partitioning algorithms used to assign memories to controllers. These algorithms are mainly driven by parameters such as power domains, clock domains, memory placement, test time, power levels, and others. The proposed method creates repair logic sharing groups after this initial partitioning. These groups are local to each controller. The groups are automatically defined based on different criteria explained later and a user-defined limit defining the maximum size of each group. For example, a group might consist of at most 100K bits of memory. This could represent several memories, a single memory or a portion of a large memory. The built-in repair analysis (BIRA) module associated to a group must be able to compute a common repair solution even if the group is composed of memories with different characteristics (size, physical organization, pipeline depth, etc…) similar to what the memory BIST controller itself can do. This means that the column address range processed by the BIRA module can be defined by one memory and the row address range defined by a different memory.

A BIRA module allocates at most one unique spare row unit and one unique spare column unit within a group. This means that if a memory has multiple segments, each with their own spare resources, the same repair solution is applied for all segments. A typical example is a memory with two segments, left and right, and each segment has one spare IO which can be allocated independently if repair sharing is not enabled.