Memory Built-In Self-Repair

Volker Schöber, Olivier Picot

Infineon Technologies

Abstract

This article describes a word oriented memory test methodology for Built-In Self-Repair (BISR). It contains memory BIST logic, wrapper logic to replace defect words , fuse boxes to store the failing addresses. This allows to use RAMs without spare rows and spare columns used in classic redundancy concepts. Faulty addresses and its expected data will be stored in the redundancy logic immediately after its detection. The BISR simply adds faulty words to the redundancy as long as spare words are available. This avoids unnecessary external or internal redundancy calculation. It is possible to add faulty addresses to faults that have been detected during former runs. The presented memory test allows a memory BISR even if parts of the redundancy is already configured. The fuse box can be connected to a scan register to stream in and out data during test and redundancy configuration. The BISR concept can be described in RTL code. Standard MBIST RTL controllers can be used and adopted to the redundancy wrapper logic. The redundancy and BIST logic is fully synthesizable and can be prepared for reuse. Therefore, it is highly flexible and can be used wit various memory types.

Keywords

BIST, Built-In Self-Repair, BISR, memory test, fuse boxes

1. Introduction

Today's deep submicron technologies allow the implementation of multiple memories on a single chip. Due to their high density memories are more prone to faults. These faults impact the total chip Yield. One way to solve this problem is to enhance the memory by redundant memory locations. The address mapping of the fault free working memory is programmable within certain limits. In order to do so, a memory test is needed to identify the faulty regions. There are basically different test solutions available as you can see in chapter 6.

The memory is tested by external test hardware or by on chip dedicated hardware (memory BIST). The second testing strategy is the preferred method for embedded memories. After memory testing the memory address map is programmed by means of volatile or non-volatile storage on or off chip. To provide the test pattern from a memory BIST a multiplexer in front of the memory is widely used. The redundant spare rows and spare columns are often included into the memory. This impacts the performance and area conditions of the memory.

The presented BIST concept prefers a redundancy logic that is placed in parallel to a memory without spare rows and spare columns. There will be no additional delay for the word redundancy logic on top of a memory in the data path of the memory. In addition, the memory layout generation procedure must not be changed. This idea discloses a new concept for a fuse box logic with test features and a memory test method which computes the address map of a memory with redundancy on the fly without penalty on total test time because the redundancies can be activated immediately within one clock cycle after detecting a failure. The principle structure is shown in Fig. 1.

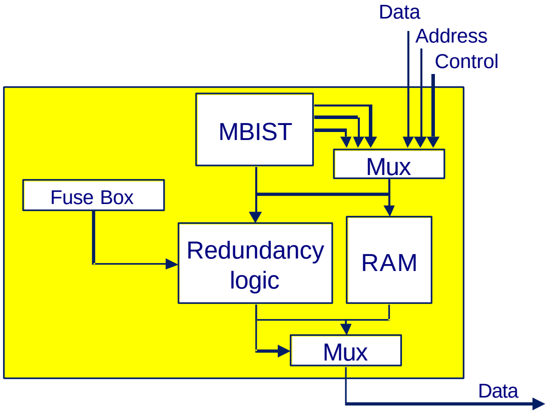

Fig. 1: The memory BIST and Self-Repair (MBISR) concept

The memory is repaired during testing by storing faulty addresses in registers. These addresses can be streamed out after test completion. Furthermore, the application can be started immediately after the memory BIST passes. The redundancy logic calculation will not increase the test time of the memory BIST.

The MBISR concept contains an interface between MBIST logic and redundancy wrapper for storing faulty addresses. This allows to use already existing MBIST solutions. The MBIST controller output must provide three signals to the wrapper logic during test.

- A fail signal to store data in the fuse register
- The expected data that is compared to the results of RAM data
- The failing address

2. The Wrapper Redundancy Logic

The redundancy logic that are wrapping the memory consists of two basic components. Spare memory locations and a way to make the address decoding programmable by disabling defective memory locations and enabling spare memory locations. For the access to a memory with redundancy two possibilities exist.

Usually, the access to memory locations is done as an EXCLUSIVE OR operation. With one memory address only one memory word is accessed. Hence, the address comparison with faulty addresses has to be completed before the access to the memory array starts by proper address decoding.

In this approach a parallel access to the memory and the redundancy logic is proposed. The address comparison is done in the redundancy logic. The actual address is compared to the addresses that are stored in the redundancy word lines. A multiplexer at the output of the memory and the redundancy logic decides where to take the data from. More than one redundant word line can be placed in the redundancy logic. With each new failing word all redundancy information are shifted to the next word line. The data comes from MBIST controller for each failing word individually. An overflow bit identifies that there are more failing addresses than possible repair cells. The principle is shown in Fig. 2.

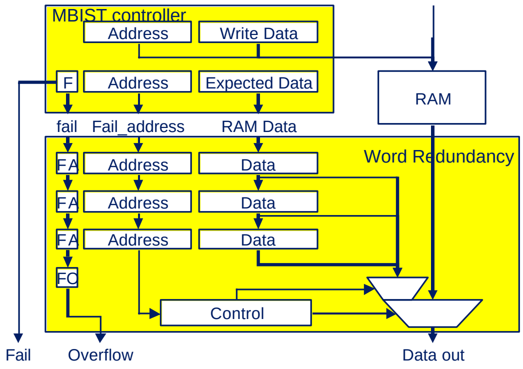

Fig. 2: An array of redundant word lines

If an address is stored in a register the FA register is set to "1" to activate the spare word. Then, the data register is used to read and write instead of the memory array. The programming of the faulty addresses are done during the memory BIST or the fuse box. The failing addresses can be read out after the memory test to program fuse boxes. In addition, already identified failures can be write into the FA and address register.

The structure for one redundancy word line is shown in Fig. 3. During test the MBIST prepares the signals for Fail, Fail_address, Expected_data. The R, WR, address (A) and DI are accessed in parallel to the memory and the redundancy logic during functional operation and test. TDI and TDO are serial interfaces for the memory. A Read signal controls the multiplexer of data DO of the memory Data out bus.

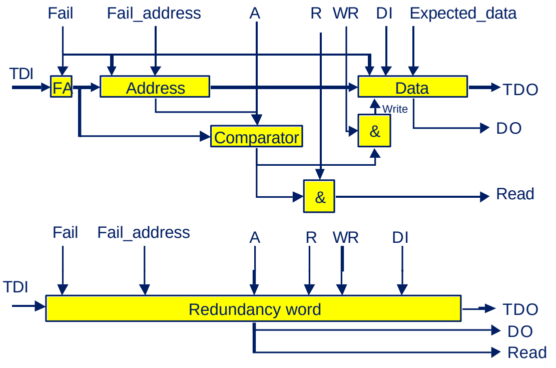

Fig. 3: Redundancy word line

3. The Fuse Boxes

To store identified failures after memory test fuse boxes can be used on and off chip. Fuses on chip are state of the art. They are blown after production test. One fuse carries one address bit. Feedback structures stores the fuse values after probing the fuse (similar to a dynamic logic). The fuse itself is nothing more than a polysilicon or metal resistor, depending on technology. In normal chip operation the fuses are probed at power on and their values are stored in feedback structures, e.g., back to back inverters. From a testing point of view three problems arise:

- The logic of the fuse box has to be tested before packaging to reduce defect probability of fuse boxes.
- An easy way to set fuse values from external source without blowing the fuse is helpful. This allows a pre fuse test and proof of the determined faulty memory locations for reliability tests, Yield improvements and diagnosis capabilities.
- A possibility to read the fuse values directly after the fuse blowing process to enhance observability of the fuse process.

The proposed fuse box which is part of the redundancy concept contains additional logic to the back to back inverter. It contains a scan Flip Flop for controlling and observing the fuse data. Therefore, two modes are added to enhance the testability.

- Test update=0: The chain of inverters is closed. The value of node B can be set to zero or one whether the fuse is blown or not.
- Test update=1: It is possible to set the internal node directly from the TDO of the scan Flip Flop. If test update goes back to 0 then the value is latched in the two inverters allowing a direct control of Fout, the output of the fuse.

These two operations can be implemented by an inverting multiplexer instead of an inverter. It is also possible to set the value through a pass gate without opening the inverter chain. The observability of internal nodes are assured if a scan register is connected to the output Fout of the fuse box. The ports TDI and TDO are part of a scan chain and activated during scan mode. A principle structure of the fuse box is shown in Fig. 4.

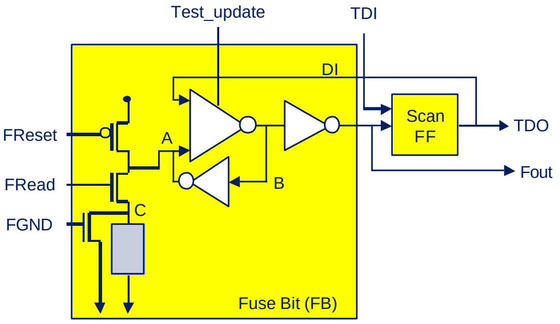

Fig. 4: Fuse Box and scan flip-flop configuration

To read out the fuse values and store the data in the inverter loop the control signals FReset, FRead and FGND are processes as is shown in Fig. 5 which can be derived from a reset signal and is shown in Fig.4.

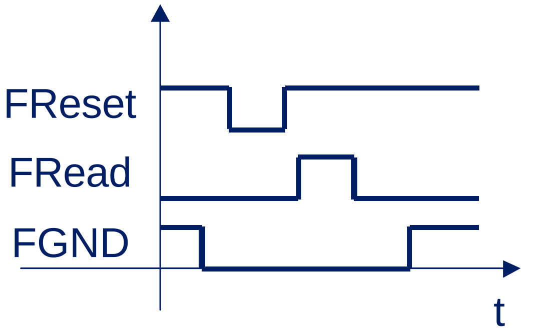

Fig. 5: Reset cycle to read out the fuse information

To setup a fuse box multiple fuses and their register are placed in parallel. There is one more fuse cell necessary to activate a programmed address after the fuses for a faulty address are blown. The scan Flips Flops are configured to a serial scan chain that can be activated during scan mode, as is shown in Fig. 6. The data out of the scan Flip Flop is connected to the input DI of the fuse box.

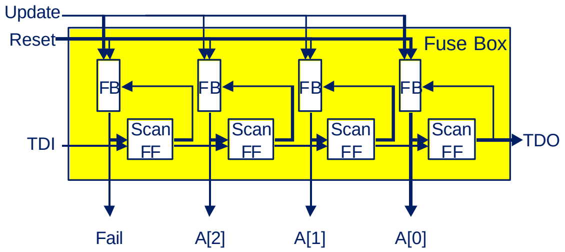

Fig. 6: 4 bit fuse box for a 3 bit address including serial load of the fuse information

Fuse boxes can be placed inside or outside the redundancy logic. If the boxes are placed outside two configurations are possible. Parallel buses are connected the fuse boxes to the address registers of the redundancy word lines. This is shown is Fig. 7. Within one clock cycle the redundancy logic can be initialized from fuse boxes.

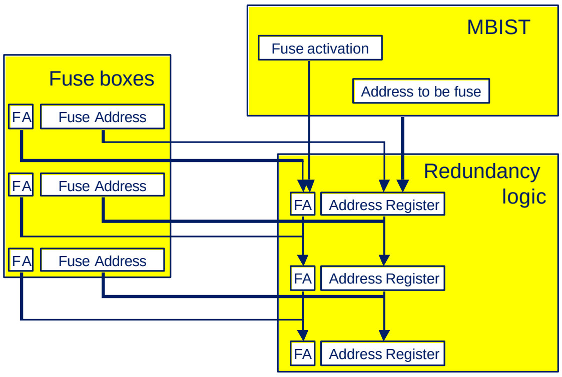

Fig. 7: Parallel access of the fuse information

Instead of parallel access it is also possible to implement serial shift logic between the fuse boxes and the redundancy word lines, as is shown in Fig. 8. This implementation avoids large busses on the chip but requires a number of additional clock cycles to shift the data. The additional Flip Flops for the serial shift operation is already implemented due to a scan based ATPG approach that is commonly used in modern designs. Therefore, a test mode to scan in and out data can be used to initialize the redundancy logic. The number of cycles will be C=K*(N+1), while K is number of redundant words and N is the size of the address bus.

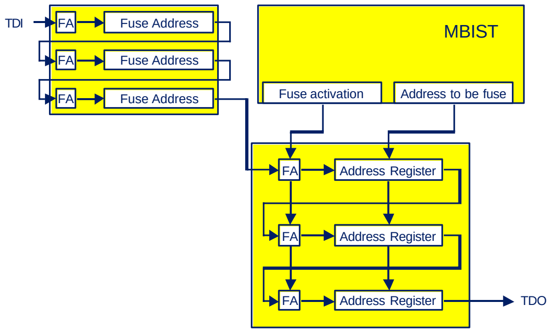

Fig. 8: Serial access of the fuse information

The memory BIST controller is able to activate additional redundant word lines after the fused values are shifted into the wrapper logic. This will be explained in a following chapter.

4. Memory BIST Redundancy Principle

The memory BIST including redundancies is divided into a memory BIST controller part and the redundancy logic. The redundancy logic can be used with a standard memory BIST controller. The presented concept is independent to other implementation strategies of the memory BIST and its algorithm. Therefore, the concept can be adopted to company wide BIST flows or memory BIST generators. The MBIST controller had to provide only the following internal signals for the redundancy logic.

- The expected data that is used to compare the test results from the memory inside the MBIST controller.
- The failing address of the faulty word need to be provided.
- A fail signal that can be used as a write enable for the redundancy wrapper.

Fig. 9 shows these signals that enables the programming of the redundancy logic.

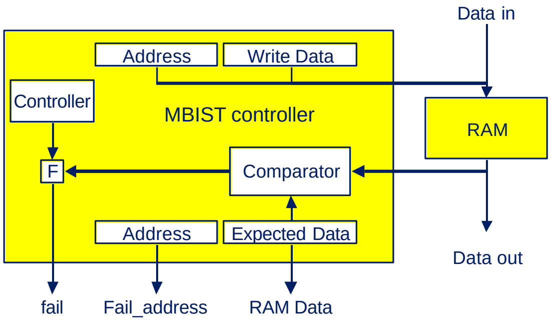

Fig. 9: MBIST structure with its interface to the redundancy logic

An on chip memory test runs through the address space of the memory and does write and read operations in a given order (depending on test algorithm). The memory output is compared with the expected data. If both words are not equal parts of the respective memory word is faulty. In this case, the faulty address must be stored in the redundancy logic. In addition, the redundancy wrapper needs an activation signal and the expected data to activate a word line redundancy with one cycle. This allows an at speed redundancy calculation. It is possible to store the faulty address immediately after detecting an error.

MBISR with reset: The failing address is stored in the redundancy wrapper and is therefore activated. By doing so, the faulty address is fixed immediately to proceed. In the next step, the MBIST starts again from the beginning. It runs until the next failures is found or the test is finished. This concept can also be used when the test control is placed in a CPU or in a tester while the redundancy wrapper is implemented in the chip closed to the memory. This approach is a more time consuming including a reset after each failure procedure than the following description.

MBISR without reset: To proceed the test without interruption the expected data are also stored into the redundancy wrapper logic. Then, the redundancy logic contains the faulty address and the correct data of the failing memory address. This avoids a restart of the memory test algorithm. To validate that the redundancy logic does not contain any failures a scan based ATPG test is proposed before the memory test. Depending on the implementation a few clock cycles might be needed for storing the faulty address and the expected data.

The test program flow to activate the redundancy is shown in Fig. 10. Three different results are possible.

- A software repair is done before or after a hardware repair. This degree of freedom allows a flexible usage during fabrication test and system test during the debugging phase and in field applications.
- A hardware repair is done including the process to blow the fuses. This is normally done a wafer level test because most of the fuses are activated with laser before packaging.
- An repair overflow when too many failures can been encountered. This allows the test to identify failures that cannot be repaired.

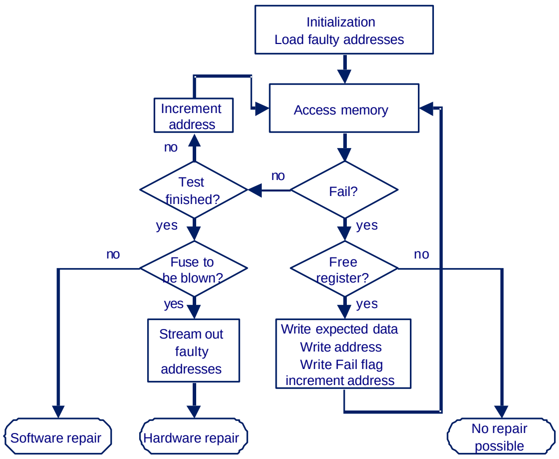

Fig. 10: Test flow to activate the redundancy

5. Conclusion

A new memory Built-In Self-Repair concept has been presented that uses spare words instead of spare columns and rows. It allows to proceed a software repair before and after the fuses of a redundancy wrapper are blown. In addition, the results of a software repair can be used to blow on-chip fuses. This allows to use the MBISR during wafer and package test. Because of its open architecture it is capable to read out all necessary diagnostic information. In addition, it allows to repair memories during field application as long as spare words are available when a MBISR test finds failures.

The word redundancy can be programmed at speed of the memory BIST. There will be no additional delay for the redundancy wrapper because the redundancy is implemented in logic that is faster than the memory access. The redundancy and MBIST logic can be tested with a traditional scan based ATPG approach together with the embedded logic of the chip.

The implementation of the BIST is based on RTL code and can be used together with standard memories that do not have any redundancy capabilities. The number of the spare words for the redundancy wrapper is scalable in the RTL code.

6. Literature

[85Day] J. R. Day, „**A Fault-Driven Comprehensive Redundancy Algorithm**“, IEEE Design & Test of Computers, pp.35-44, June 1985.

[92ChenS] T. Chen, G. Sunada: “**A Self-testing and Self-Repairing Structure for Ultra-Large Capacity Memories**”, International Test Conference, pp. 623-631,1992.

[92TanaTK] A. Tanabe, T. Takeshima, H. Koike, Y. Aimoto, M. Takada, et. al.: “**A 30 ns 64-Mb DRAM with Built-In Self-Test and a Self-Repair Function**”, IEEE Journal on Solid-State Circuits, Vol. 27, November 1992.

[93TreuA] R. Treuer, V. Agarwal: “**Built-In Self-Diagnosis for Repairable Embedded RAMs**”, IEEE Design & Test of Computers, pp.24-33, June 1993.

[96NordON] P. Nordholz, J. Otterstedt, D. Niggemeyer: “**A Defect-Tolerant Word-Oriented Static RAM with Built-In Self-Test and Self-Reconfiguration**”, Innovative Systems in Silicon, pp. 124-132, October 1996.

[97YounP] L. Youngs, S. Paramanandam: “**Mapping and Repairing Embedded Memory Defects**”, IEEE Design & Test, pp. 18-24, January,1997.

[98KimZK] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. Higgins, J. Lewandowski: “**Built-In Self-Repair for Embedded High Density SRAM**”, International Test Conference, pp. 1112- 1119,1998.

[99Day] D. K. Bhavsar, **„An Algorithm for Row-Column Self-Repair of RAMs and its Implementation in the Alpha 21264“**, International Test Conference, pp. 311- 318,1999.

[99NakaHK] S. Nakahara, K. Higeta, M. Kohno, T. Kawamura, and K. Kakitani, „**Built-In Self-Test for GHz Embedded SRAMs Using Flexible Pattern Generator and New Repair Algorithm**", International Test Conference, pp. 301- 310,1999.

[00KawaON] T. Kawagoe*, J. Ohtani, M. Niiro, T. Ooisih, M. Hamada, H. Hidaka: „**A Built-in Self-Repair Analyzer (CRESTA) for Embedded DRAMs“**, International Test Conference, pp. 567-574, 2000.