Design of Reparable Memory Systems with Shared Row Redundancies

Karen Amirkhanyan, Synopsys, Yerevan, Armenia, kamirkha@synopsys.com

Samvel Shoukourian, Synopsys, Yerevan, Armenia, samshouk@synopsys.com

Valery Vardanian, Synopsys, Yerevan, Armenia, vvardani@synopsys.com

ABSTRACT

In this paper, we proposed a method for implementation of a mechanism for "redundancy sharing" allowing to repair a fault/defect in a memory instance from a memory system with hundreds of memory instances with an available shared redundant element of another memory instance. The calculations showed that hardware is saved to a great extent with a negligible impact on memory's functional performance.

Keywords

Memory system, repair, redundancy sharing, reparable/non-reparable memory.

1. INTRODUCTION

Built-in Self-Repair (BISR) is widely used for improving memory-core yield. Although, the portion of a BISR circuit with respect to the area of the corresponding memory instance area is usually small, but due to the number of hundreds of memory instances in a memory system (MS), when each memory instance with redundancy had its own BISR circuit, the total area of the BISR circuits in an MS increased to a great extent. To decrease the overall area overhead of BISR circuits in an MS, many researchers proposed some notions of "shared BISR", grouping, reusing, multiple shared buses, etc. (see [1]-[12]). They reduced the area overhead by performing test and repair of memory instances serially that increased the time for test and repair. To reduce area overhead and test & repair time, "shared BISR" was used performing test & repair of multiple memory instances in parallel. Hence, grouping is another approach to be applied with respect to the typically great number of memory instances in an MS to make use of the property of structural identity of memory instances from the same group.

Today's complex MS usually contains hundreds of memory instances. Built-in Self-Test (BIST) is one of the main approaches for testing memories in an MS [1-12]. ARM company has introduced a shared test bus and used it for efficient test and repair purposes. Synopsys's DesignWare Multi-Memory Bus (MMB) [3], based on ARM's shared test bus, also used the bus for efficient test of multiple memory instances attached to the bus. Mentor Graphics [2] proposed to use a functional bus for its efficient testing in addition to the test bus. We suggest here an idea of using the functional bus for efficient repair of memories via "redundancy sharing" due to which time and hardware are saved during the repair [1-12].

The redundant elements remaining after manufacturing repair can be used in the field during test and repair sessions and soft repair performed periodically after power-up (see [1]). However, in all above mentioned approaches [1-12] each memory instance with redundancies used its own redundancies as local redundancies for repair within one instance only and did not have the capability to use other available redundancies allocated for other memory instances with redundancies. In other words, the redundancies of memory instances were not sharable, they could not be used by other memory instances. Although the proposed in [1] SMS (STAR (Self-Test and Repair) Memory System) test and repair solution shows high efficiency, however, there is still a possibility to develop the approach farther allowing to save much more hardware. There is an efficient way to save hardware due to an effective and flexible usage of hardware. It is based on the idea of "a sharing mechanism for redundancies". We connect the redundancies with the functional bus and use them for the repair of any fault in any reparable memory instance connected to the shared functional and test buses. BIST is performed by means of the shared test bus, and the repair is performed by means of the functional bus. For the sake of accuracy, we should mention three publications [16], [17], [18] where the expression "redundancy sharing between different memories" was mentioned. In [16], however, the text lacked in details connected with the usage, implementation and estimation of the impact on hardware and time of the chip performance. In [17], a BISR technique for multiple repairable memory instances with block-based redundancies was proposed. Redundant rows and columns were divided into row and column blocks and repair was being performed at the block level. Based on the proposed block-repair mechanism, a heuristic redundancy analysis algorithm was proposed. However, for the considered small example of only four memory instances the hardware overhead was only 7.6 % and it is not clear at all how much it will be for hundreds of memory instances. In [18], a Content-Addressable-Memory (CAM) -based shared BISR structure is proposed for test and repair of RAMs, as well as the corresponding repair strategy for the shared BISR. Although the authors claim its high efficiency with low area overhead but, however, the authors do not talk about the memory access mechanism which involves shared repair justifying this with their aim to simplify discussion.

2. REDUNDANCY SHARING MECHANISM FOR REPARABLE MEMORIES

Meantime, in today's new technology memories, defects/faults are becoming less probable, and the conventional usage of redundancies in each reparable memory instance seems to be not efficient any more. The number of possible defects that can be found in memory instances in an MS can be estimated based on the current semiconductor technology and the corresponding statistical data on defect density for memory instances and the area occupied by them. Defects in memories are now being measured by new criteria and new notions, such as DPPB (Defective Parts Per Billion) (see [19]), that has been introduced recently meaning that memory devices are now very robust and reliable with very low probabilities for fault/defect emergence. We propose to share redundancies between all reparable memory instances and, thus, allocate redundant elements not within individual memory instances but within the whole MS introducing the capability of repairing any fault/defect that could be detected in any memory instance in the MS. Note that initially the MS could contain instances without redundancies, i.e. unrepairable memory instances that were initially unrepairable, but however, after applying the proposed approach of redundancy sharing they will become reparable. Thus, redundancy sharing will also increase the reliability of the MS as a whole since after the modification all memory instances will become reparable. We propose to deprive all reparable instances in the MS of all redundant elements and keep only one reparable instance with several global redundancies designated for the repair of all memory instances, and we should have a possibility for the memory instance to repair each defect in each memory instance contained in the MS. The few redundancies allocated for one reparable memory instance should be enough for repairing all possible faults in each memory instance in the MS. The defects that will be detected during the BIST session will be repaired by the BISR engine. Since it is not predictable the actual location of faults/defects in the MS, then the only requirement is to have the capability for sharing these few redundancies for repairing the possible defects/faults in all memory instances.

A similar methodology is being currently developed for the "shared repair with column redundancies". Due to the page limitations and for the sake of simplicity, we will constraint ourselves in this paper with consideration of a simple case when the MS contains only row redundancies, and all memory instances are identical instances without redundancies, and there is only one reparable memory instance of the same size with redundant rows. The MS considered has a test bus (used for test) and a functional bus (used for repair) of all memory instances in the memory system.

3. SOME DETAILS OF IMPLEMENTATION

3.1. Conventional Hardware Implementation

Consider a memory system consisting of hundreds of memory instances. Memory instances are either reparable memories with redundant rows or memories with no redundancies.

First, we apply Synopsys's DesignWare STAR memory grouping tool to group the memory instances into sub-groups according to the memory structural parameters and including the memory instances with identical structure into one group. Thus, consider a sub-group of memory instances $G_i = \{M_1, M_2, \ldots, M_s\}$ where all memory instances M_i, $i=1,\ldots, s$, have the same structure, i.e. the same number of rows and columns, all are with the same number of redundant rows, or all of them have no redundancies. Note that we exclude the special case when all the memory instances in a group G_i are all without redundancies. In this case, however, we suggest to keep the structure of the group unchanged.

Now, suppose a group G_i contains memory instances $M_1, \ldots, M_s$ where each instance M_i has two redundant row groups, each redundant row group consisting of D_x physical rows. This parameter depends on memory technology and is the same for all memory instances in a sub-group. We propose a few basic steps for implementation of the hardware. Figure 1 depicts the conventional solution of the SMS group G_i. We consider a special case when each memory instance M_i, $i=1, \ldots, s$, has two redundancy row groups where each row group R_1 and R_2, consists of D_x redundant physical rows.

Next, in Fig. 1, 1500 is the 1500 standard interface, M_i, $i=1, \ldots, s$, are the memory instances with redundancy elements, Wi are the wrapper logic of the memories M_i, $i=1, \ldots, s$, for BISR, Rl. R2, are respectively the first and second row redundancy group for M_i with D_x physical row redundancies the redundancy elements, ME_i are the enable signals of the functional bus for memory M_i. symbols "X" are examples of defects/faults in the memory instance.

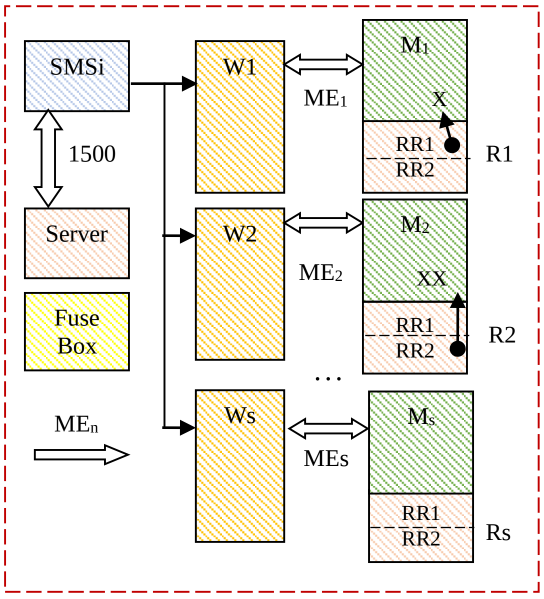

Fig. 1. Conventional structure of the SMS group

In the conventional SMS group, the repair mechanism is implemented by means of the redundant elements in a memory. Those redundancies are placed locally in the memory instance and can be used for repairing of the defects in the particular memory instance only that is a big limitation for the memory system reparability, in general.

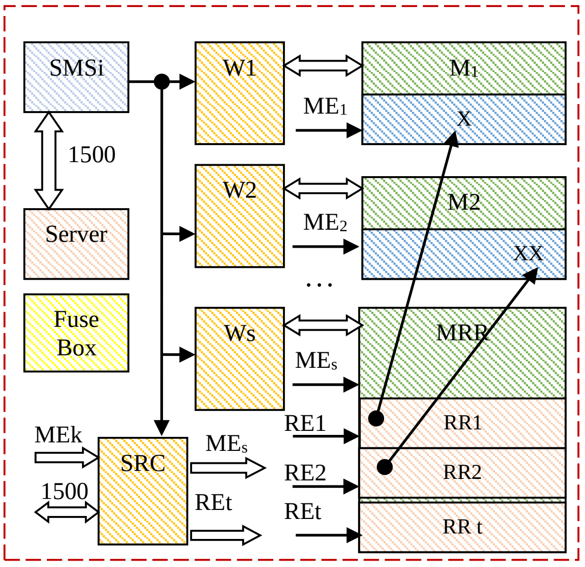

Fig. 2. *The SMS group with the shared row redundant elements*

3.2. Implementation of the Proposed Shared Row Mechanism

In Figure 2. the implementation of the memory sub-group of identical memory instances with two redundant row groups with the shared row redundant elements is presented.

SRC is the shared redundancy control RTL unit, REn_i, i=1, …, s, are the redundancy enable signals, M_i are the memory instances without the redundant rows, MRR is the memory instance with the shared row redundant elements, number of which is determined by multiplication of Defect Density and Memory Area in the memory system, the symbol.

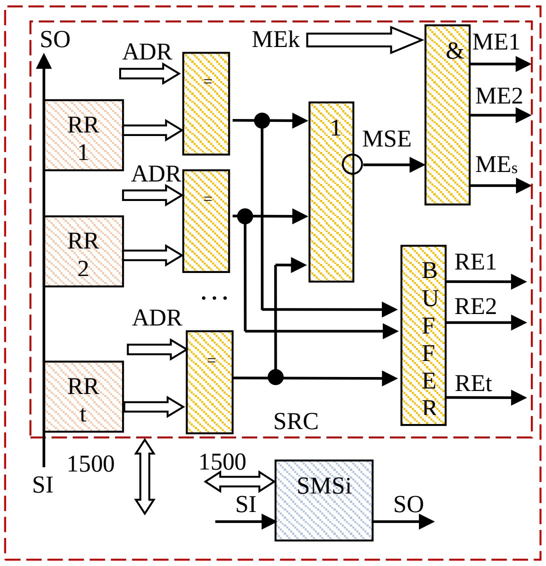

Fig. 3. *Flow Chart of the Shared Redundancy Control (SRC) unit*

4. ESTIMATIONS OF PERFORMANCE AND AREA SAVING

4.1. Performance Evaluation

The main issue related with SCR implementation is the time delay which is added additionally to the path of Memory Enable (ME) signal. This delay can be critical especially for the case of performance of the functional bus at the maximum frequency. Some SPICE simulations have been done by using the memory model of 28nm technology. The simulation results confirm the correct work of the memory with SRC unit at the maximum frequency.

In the case when the "shared redundancy mechanism" is used, the repairing of a defect in each memory in the memory system is performed by means of the row redundancy elements which are placed in the special memory instance MRR. The repairing is done through the functional bus of the memory system. The switching between a defective address and a redundant element is implemented in the SRC RTL unit. In Figure 3. the flow chart of the SRC unit is presented. The SRC block contains the redundancy registers (RRi) that are intended for storing of the address of a defective cell. The output of the register RR is joined with a comparator (=).

The comparator compares in real time the value of the RR with the address value on the address of the functional bus (ADR signals). If the address coincides with the address of RR then SRC unit blocks the memory enable signal ME_i of the corresponding memory instance on AND element(s) and simultaneously activates the Redundancy enable (RE_k) signal for the corresponding redundant element. The number of RR registers is the same as the number of the redundant elements in the MRR memory instance. Information in RR registers can be updated through serial input/output ports (SI and SO signals) by SMS processor or by Server by means of standard 1500 interface.

4.2. Area Saving

Introduce the following notations:

D_x - number of physical rows in a redundant row group,
D_M - defect density of an SRAM memory,
A_M - area of a memory with redundancies,
D_M . A_M – number of possible defects in a memory with redundancies,
A_b – area of a bit-cell b in a memory,
m – number of memories with redundancies on the functional bus of the memory system,
k – number of columns in a memory,
l – number of rows in a memory.

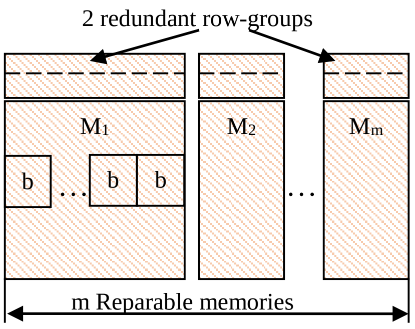

Fig. 4. Group of identical memory instances with two redundant row-groups

According to our proposal of "shared row mechanism", all the memory instances with redundancies in a memory group of identical memory instances are proposed to replace with corresponding instances with no redundancies but retaining only one reparable instance with the number of row redundancies equal to the multiplication of memory defect density by the area of the memory system. As a result, since row redundancies of the other instances are removed, and since the number of such instances with excluded row redundancies may be a few hundred in modern SoCs then the area saving is significant. Based on Figures 4 and 5, the percentage of the redundancy area saving can be estimated by the following formula:

$$\Delta R = ((R_M - R^*_M)/RM)\ \ 100\ \%\ ,$$

where R_M is the area of the redundant rows in the given memory system, and R^*_M is the area in the modified circuit after excluding the unnecessary redundant rows. Then area saving can be estimated as follows:

$$\Delta R = (1 - DMAM/m)\ 100\ \% = (1 - lkAbDM)\ 100\%\ .$$

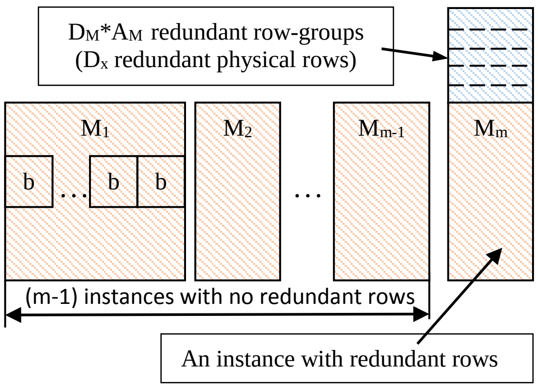
DM*AM redundant row-groups
(Dx redundant physical rows)
M1
M2
Mm-1
Mm
b
b
b
. . .
(m-1) instances with no redundant rows
An instance with redundant rows

Fig. 5. Group of m identical memory instances with two redundant row-groups

Our calculations for different configurations of memory instances contained in the memory system showed significant saving of hardware due to the sharing mechanism of row redundancies in memory instances resulting in exclusion of a great amount of redundant area in memory instances.

5. CONCLUSION

In this paper, we proposed a "redundancy sharing mechanism" for the repair of a fault/defect in a memory instance from a memory system with hundreds of memory instances with an available shared redundant row of another memory instance. The calculations showed that hardware is saved to a great extent with a negligible impact on memory's functional performance.

In the future, we are planning to extend this research for the cases of redundant columns and 2D redundancy when both redundant columns and rows are available in memory instances.

REFERENCES

[1] Y. Zorian and S. Shoukourian, ''Embedded-Memory Test and Repair: Infrastructure IP for SoC Yield,'' *IEEE Design & Test of Computers*, vol. 20, no. 3, pp. 58-662003.

[2] D. Sargent, "Viewpoint: Memory BIST for shared-bus applications", *EDN Network*, February 16, 2012, http://www.edn.com/design/test-and-measurement/4389500/Viewpoint-Memory-BIST-for-shared-bus-applications .

[3] C.-L. Su, R.-F. Huang, and C.-W. Wu, ''A Processor-Based Built-in Self-Repair Design for Embedded Memories,'' *Proc. 12th Asian Test Symp. (ATS 03),* IEEE CS Press, pp. 366-371, 2003.

[4] R.C. Aitken, ''A Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories,'' *Proc. Int'l Test Conf. (ITC 04)*, IEEE CS Press, pp. 997-1005, 2004.

[5] T.-W. Tseng, J.-F. Li, and C.-C. Hsu, ''ReBISR: A Reconfigurable Built-in Self-Repair Scheme for Random Access Memories in SOCs,'' *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 18, no.6, pp. 921-932, 2010.

[6] C.-D. Huang, J.-F. Li, and T.-W. Tseng, ''ProTaR: An Infrastructure IP for Repairing RAMs in System-on-Chips,'' *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1135-1143, 2007.

[7] T.-W. Tseng, J.-F. Li, and C.-S. Hou, "A Built-in Method to Repair SoC RAMs in Parallel", *IEEE Design & Test of Computers*, vol. 27, no. 6, pp. 46-57, 2010.

[8] N. B. Singh, A. Bhat, A. Anand, R. Tiwari, A. Kothiala, "Method and Apparatus for Multiple Memory Shared Collar Architecture", *US Patent application 20160062864*, 2016.

[9] C.-L. Su, R.-F. Huang, and C.-W. Wu, "A Processor-Based Built-In Self-Repair Design for Embedded Memories," *Proc. 12th Asian Test Symposium (ATS'03)*, pp. 366-371, 2003.

[10] K. Darbinyan, G. Harutyunyan, S. Shoukourian, V. Vardanian, Y. Zorian, "A Robust Solution for Embedded Memory Test and Repair", *IEEE Asian Test Symposium (ATS)*, pp. 461-462, 2011.

[11] S. Bahl, B. Singh, "A Novel Method for Silicon Configurable Test Flow and Algorithms for Testing, Debugging and Characterizing Different Types of Embedded Memories through a Shared Controller", *Records of the 2004 Int'l Workshop on Memory Technology, Design and Testing (MTDT'04)*, pp. 78-83, 2004.

[12] N. Bhushan Singh et al, "Method and Apparatus for Multiple Memory Shared Collar Architecture", *US Patent Application US2016/0062864 A1*, 2016.

[13] Synopsys Press Release: "Synopsys STAR Memory System Multi-Memory Bus Processor Enables 10 Percent Die Size Reduction for Marvell SoC," http://news.synopsys.com/2014-10-21-Synopsys-STAR-Memory-System-Multi-Memory-Bus-Processor-Enables-10-Percent-Die-Size-Reduction-for-Marvell-SoC .

[14] Synopsys Press Release: "Imagination Technologies Adopts Synopsys STAR Memory System for Embedded Memory Test and Repair for New MIPS Processor," http://news.synopsys.com/2016-11-15-Imagination-Technologies-Adopts-Synopsys-STAR-Memory-System-for-Embedded-Memory-Test-and-Repair-for-New-MIPS-Processor .

[15] S. Adham, P. Saggurti, "TSMC's Use of Synopsys STAR Memory System: Features and Capabilities", https://webinar.techonline.com/19700?keycode=CAA1CC .

[16] S. Bahl, "A Sharable Built-in Self-repair for Semiconductor Memories with 2-D Redundancy Scheme", *22nd IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 331-339, 2007.

[17] S.-K. Lu, Z.-Yu Wang, Yi-Ming Tsai, and Jiann-Liang Chen, "Efficient Built-In Self-Repair Techniques for Multiple Repairable Embedded RAMs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 4, pp. 620-629, 2012.

[18]. G. Wang, C. Chang, "Design and Implementation of Shared BISR for RAMs: A Case Study," IEEE Autotestcon, Anaheim, CA, pp. 1-7, 2016.

[19] D. Park, "Move ICs from defects per million to defects per billion", http://www.edn.com/design/test-and-measurement/4443160/Move-ICs-from-defects-per-million-to-defects-per-billion .