PERFORMANCE ANALYSIS OF LOW POWER BUILT-IN SELF REPAIR TECHNIQUES FOR EMBEDDED MEMORIES

P. Alagappan, PG-Scholar, Department of Electronics and Communication Engineering, AVS Engineering College, Salem, alagumty@gmail.com
A. Sathishkumar, Assistant Professor, Department of Electronics and Communication Engineering, AVS Engineering College, Salem, ecesat@gmail.com

Abstract

Error correction code (ECC) and built-in self-repair (BISR) techniques by using redundancies have been widely used for improving the yield and reliability of embedded memories. The target faults of these two schemes are soft errors and permanent (hard) faults, respectively. In recent works, there are also some techniques integrating ECC and BISR to deal with soft errors and hard defects simultaneously. However, this will compromise reliability, since some of the ECC protection capability is used for repairing single hard faults. To cure this dilemma, we propose an ECC-enhanced BISR (EBISR) technique, which uses ECC to repair single permanent faults first and spares for the remaining faults in the production/power-ON test and repair stage. However, techniques are proposed to maintain the original reliability during the online test and repair stage. We also propose the corresponding hardware architecture for the EBISR scheme. A simulator is implemented to evaluate the hardware overhead (HO), repair rate, reliability, and performance penalty. Experimental results show that the proposed EBISR scheme can improve yield and reliability significantly with negligible HO and performance penalty.

Keywords: Decoder unit (3:8), Logic Unit, Comparator Unit, Response Verifier (RV), Test Generator (TG).

1. INTRODUCTION

It is well known that embedded memories are dominating the yield of system chips due to the aggressive design rules and the large occupied area proportions. Therefore, there are many proposed fault-tolerance techniques used to boost the yield and reliability. In the past, hard repair techniques by using redundancies are widely used for repairing permanent faulty memory cells. That is, spare rows (SRs)/spare columns (SCs) are used to replace faulty memory cells. One promising solution to implement the fault replacement technique is the built-in self-repair (BISR) method. To achieve the goals of BISR, three basic modules, including memory built-in self-test (BIST), built-in redundancy analysis (BIRA), and address reconfiguration (remap-ping) (AR) are usually required. The BIST module executes the adopted March test algorithms to test the memory array to see if there are faults detected. After detecting faulty bits, the BIRA module performs RA algorithms for spare allocation to replace
faulty memory bits. The AR module is used to remap faulty addresses to spare addresses.

There are two possible scenarios to add redundancies into an embedded memory array.

1) 1-D Redundancy: 1-D redundant rows or columns are added into the memory array. One of the redundant Rows/columns is used to replace a faulty row (FR)/a faulty column (FC). The main advantage of this approach is that the RA is very simple and can be implemented easily. However, the repair efficiency of this approach is limited, since only 1-D spares can be used.

2) Redundant Rows and Redundant Columns: Redundant rows and columns are both incorporated into the memory array. When a faulty cell is detected, either a redundant row or a redundant column can be used to replace it. It is more efficient than the 1-D approach when multiple faulty cells exist in the memory array. There are also many RA algorithms [2] proposed in the past. However, the optimal redundancy allocation problem becomes NP-complete.

2. LITERATURE SURVEY

2.1 Design of High Efficient Built in Self Repair Architecture using March Algorithms

J. Swarna Durga, B. Gangadhar

Built-In Self-Repair (BISR) with Redundancy is an effective yield-enhancement strategy for embedded memories. This paper proposes an efficient BISR strategy which consists of a Built-In Self-Test (BIST) module, a Built-In Address-Analysis (BIAA) module and a Multiplexer (MUX) module. The BISR is designed flexible that it can provide four operation modes to SRAM users. Each fault address can be saved only once is the feature of the proposed BISR strategy. Besides, instead of adding spare words, rows, columns or blocks in the SRAMs, users can select normal words as redundancy.

2.2 Built-in self-repair (BISR) technique widely used to repair embedded random access memories (RAMs)

V. Sridhar, M. Rajendra Prasad

With the trend of SOC technology, high density and high capacity embedded memories are required for successful implementation of the system. In modern SOCs, embedded memories occupy the largest part of the chip area and include an even larger amount of active devices. As memories are designed very tightly to the limits of the technology, they are more prone to failures than logic. Thus, memories concentrate the large majority of defects. That is, RAMs have more serious problems of yield and reliability. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. As a matter, Built-In Self-Repair is gaining importance. Built-in self-repair (BISR) technique has been widely used to repair embedded random access memories (RAMs). If each repairable RAM uses one self-contained BISR circuit (Dedicated BISR scheme), then the area cost of BISR circuits in an SOC becomes high. This, results in converse effect in the yield of RAMs. This paper presents a reconfigurable BISR (ReBISR) scheme for repairing RAMs with different sizes and
redundancy organizations. An efficient redundancy analysis algorithm is proposed to allocate redundancies of defective RAMs. In the ReBISR, a reconfigurable built-in redundancy analysis (ReBIRA) circuit is designed to perform the redundancy algorithm for various RAMs. The ReBISR structure has been synthesized and found that the area cost when compared with the Dedicated BISR structure is very small. This paper is implemented using Verilog HDL. Simulation and Synthesis is done using Modelsim and Xilinx ISE 12.4 Tools.

2.3 An Efficient Fault Mitigating Processor Using Low Cost Tpg
D.Pachaippan, V.Sakthivel
In this paper, Built in self repair scheme (BISR) uses the built in self test (BIST) and Built in repair analysis (BIRA). BIST is used test and detect faulty memories. It sends the fault information to BIRA. BIRA is used to find the repair solutions to the faulty memories. Among hiding counter measures, essentially distinguish strategies: one based on the randomisation of the execution of cryptographic algorithms one or more Pseudo Random Number Generators (PRNGs) are also included to generate the masks, which should be updated at each step of the data path for a more efficient masking scheme. Prng used to generate key values to overcome the limitations of manual insertion values.

2.4 Test/Repair Area Overhead Reduction for Small Embedded SRAMs
Baosheng Wang and Qiang Xu
For current highly-integrated and memory-dominant System-on-a-Chips (SoCs), especially for graphics and networking SoCs, the test/repair area overhead of embedded SRAMs (e-SRAMs) is a big concern. This paper presents various approaches to tackle this problem from a practical point of view. Without sacrificing at-speed testability, diagnosis capability and repairability, the proposed approaches consider partly sharing wrapper for identical memories, sharing memory BIST controllers for e-SRAMs embedded in different functional blocks, test responses compression for wide memories, and various repair strategies for e-SRAMs with different configurations. By combining the above approaches, the test/repair area overhead for e-SRAMs can be significantly reduced. For example, for one benchmark SoC used in our experiments, it can be reduced as much as 10% of the entire memory array.

3. EXISTING SYSTEM
In previous method presents an Optimized Built-In Self-Repair for Multiple Memories. A new built-in self-repair (BISR) scheme is proposed for multiple embedded memories to find optimum point of the performance of BISR for multiple embedded memories. All memories are concurrently tested by the small dedicated built-in self-test to figure out the faulty memories, the number of faults, and irreparability. After all memories are tested, only faulty memories are serially tested and repaired by the shared built-in redundancy analysis according to the sizes of memories in descending order. Thus, the fast test and repair are performed with low area overhead. To
accomplish an optimal repair rate and a fast analysis speed, an exhaustive search for all combinations of spare rows and columns is proposed based on the optimized fault collection. The performance of the proposed BISR is located in the optimum point between the test and repair time, and the area overhead. The main purpose is to classify the memories as faulty or not, and the number of faults in each memory is stored in the dedicated wrapper. When the fault is detected by BIST, the fault information is sent to BIRA through the port Fault_info. After the test is completed or stopped, the signal Test_finish is activated and BIRA executes the RA process to find repair solutions. In the repair procedure, RA based on the exhaustive search for all combinations of spare rows and columns is performed. If the memory under test cannot be repaired, the signal Unrepair is activated, and the test and repair are finished and the SoC is determined as a faulty chip due to the irreparable memory. If the memory under test can be repaired, the repair is performed by the repair solution. In this paper, a hard repair is performed using programmable electrical fuse (eFuse) schemes. The hard repair permanently repairs the faulty memory. Electrically programmable fuses are developed to perform hard repair using an eFuse and antifuse. After the repair is done, the signal Repair_done is asserted. It is connected to the port Test_start, and test is started for the next faulty memory. A conceptual block diagram of the proposed BIST and wrapper modules. Assume that the number of memories is n. The BIST module consists of a test pattern generator (TPG), a test address generator (TAG), and a controller (CTR). The wrapper, which is dedicated to the memory, consists of a comparator (CMP) and a fault number register (FNR). When the signal Test_start is asserted, TAG and TPG generate test patterns (Test_pattern) and test addresses (Test_address) for executing the adopted March test algorithm. The overall test procedure is controlled by the signal Test_control. The clock and reset signals are provided through Clk and Reset. The CMP compares the results from the memory and expected responses to detect faults. According to the test order, the target memory under test is tested. Whenever a fault is detected, the fault information is sent through the port Fault_info. When the detected number of faults reaches the number of faults stored in the first test, the test is stopped by the signal Test_stop. Then, the signal Test_finish is activated to prevent test time from being wasted, while the test algorithm is being completed during the serial test. The test time is greatly reduced, because most faults are detected in the first few read operations of a March test. A BISR technique for multiple embedded memories is proposed. To find optimum point of the performance of BISR for multiple embedded memories, the proposed BISR scheme is proposed. All memories are concurrently tested by the small dedicated BIST to figure out the faulty, the number of faults, and irreparability. After all memories are tested, only faulty memories are serially tested and repaired by the
global BIRA according to the sizes of memories in descending order. The proposed BISR scheme finds the optimum point between the test and repair time, and the area overhead by maintaining the optimal repair rate. In addition, the verification procedure is simply conducted through the parallel test. Therefore, the proposed BISR scheme is a Solution that trades off test and repair time, and area overhead to accomplish an optimal repair rate for multiple embedded memories in the SoC.

4. PROPOSED SYSTEM

As the VLSI technologies keep shrinkage and min-imization of devices, the attacks due to single event upsets (SEUs) become prominent and cannot be neglected. The SEUs are caused by the variations of operating environments or radioactive α particle. In general, memories are usually very sensitive to SEUs, and the probability of causing soft errors increases with the increasing of the size of memories. Fortunately, there are many online concurrent repair techniques for protecting memories from soft errors. The most popularly used technique is by using the error correction codes (ECCs). We can add additional parity bits to the information bits to detect and/or correct the faulty bits in a code-word, which consists of the check bits and the information bits.

Hamming code and Hsiao code can correct single errors and detect double ones in a given codeword, known as the single-error-correction double-error-detection (SEC-DED) code. They can be used in high-speed memories due to the low-complexity and high-speed operations of the encoding and decoding circuitries. For protecting burst faults, the Bose–Chaudhuri–Hocquenghen and Reed–Solomon codes can be used. They are mainly used for protecting CD-ROM and hard disk storage devices.

Conventionally, the target faults of the hard repair and the ECC schemes are permanent faults and soft errors, respectively. That is, these methods consider soft errors and permanent faults separately. For today’s advanced memories, we usually equip them with both ECC and BISR. If we can plan and integrate the protection capabilities of these two schemes and consider soft errors and permanent faults simultaneously, the fabrication yield and reliability can be further improved. In synergistic fault-tolerance schemes combing both ECC and BISR are proposed. The yield model of a memory incorporated with ECC and the bit/word line redundancy is derived. In, the permanent faults and soft errors can be
distinguished by the integration of the ECC circuit and the BISR circuit.

The basic scenario of these methods is to use the SEC-DED code to correct permanent single cell faults (SCFs), since the occurrence probability of this fault type is usually higher than 97%. For other fault types, e.g., FRs/FCs, spare elements can be used to repair them. Since most SCFs are corrected by ECC, the usage of spares can be reduced greatly. Unfortunately, since the ECC protection capabilities of the code words containing SCFs are used for repairing permanent faults, the ability of these codeword for protecting further soft error attacks and hard faults due to aging or process variations will become inefficacious. The reliability during in-field usage thus will be sacrificed. Therefore, we have to seek for efficient techniques to deal with this situation to maintain the original reliability.

In order to cope with this dilemma described above, we propose an ECC-enhanced BISR (EBISR) technique. It mainly consists of two stages—the production/power-ON test and repair (PTR) stage and the online test and repair (OLTR) stage. The PTR stage uses the SEC-DED code to repair single permanent faults first and spares for the remaining faults during production test and the power-ON test. The main target is the permanent faults. During the OLTR stage, the main targets are soft errors and unintentional permanent faults due to aging and process variations. The OLTR stage can detect the second fault or error by the incorporated SEC-DED code. Thereafter, the second fault/error can be corrected by the fault discrimination and corrections (FDC) phase in this stage. Unlike the conventional techniques that integrate both ECC and hard repair, the protection capability of the SEC-DED code will not be compromised even it has been used to correct a permanent fault in a codeword. Therefore, the original reliability can be maintained when a second fault occurs.

The EBISR is based on the time redundancy concept. When a second faulty bit is detected (by the inherent ability of the SEC-DED code), extra read and write operations are required. According to the first and the second readout code words, our EBISR can discriminate the fault categories (permanent faults or soft errors) and, then, correct the second faults. Although the SEC-DED code can only correct an SCF in a codeword, however, EBISR can correct up to two faulty bits with almost negligible performance degradation and hardware overhead (HO). Therefore, the reliability will not be reduced even when the ECC is used to protect permanent SCFs. However, if a codeword already contains a hard fault and Corrected by the SEC-DED code, it is really difficult for us to detect further double soft errors, since the codeword contains three faulty bits. If we also want to protect this situation, we have to use other coding techniques, which have higher protection capabilities.

The corresponding hardware architecture of the EBISR is also proposed. A simulator is developed to evaluate the HO, repair rate, reliability, and performance penalty. Experimental results show that the fabrication yield and reliability can be enhanced significantly with negligible HO and performance penalty.
The rest of this paper is organized as follows. The basic test and repair flows of PTR and OLTR stages are given in Section II. Examples are given for describing the proposed flows. The hardware architecture of the memory with the EBISR technique incorporated is presented in Section III. Experimental results that include repair rate, reliability, HO, and performance penalty are given in Section IV. Finally, the conclusions are given in Section V.

4.1 TEST AND REPAIR FLOW OF EBISR

The test and repair flow of the PTR stage is shown in Fig. 4.2. It mainly consists of the fault detection phase, the ECC repair phase, and the hard repair phase. In the fault detection phase, the BIST module executes the adopted March test algorithms to test the memory array. If no faults are detected, the memories can be shipped or used for normal access. If there are any faults detected, the BIST operations will be suspended, and the fault information including the fault addresses and the fault syndromes, should be stored. The fault syndromes can be used to identify if the faulty words contain SCFs or multiple cell faults. After storing the fault information, the BIST operations are resumed to perform the remaining tests. When the entire memory is completely tested, we then enter the ECC repair phase. The SEC-DEC ECC code is used to repair all permanent SCFs. Since the SCFs occupy the largest proportion of all the possible fault types, correcting them by ECC can increase the efficiency of spare usage, and therefore, the fabrication yield and reliability can be raised significantly. We assume that during the very short BIST operation time, the probability of SEU attacks is very low. Therefore, during the PTR stage, we do not have to consider the effects of soft errors, and the inherently incorporated ECC can be merely used for correcting SCFs.

If all permanent faults are SCFs, then the memory array can be repaired successfully without using any redundancies. The memory can then be shipped or accessed normally. However, if there are other types of faults (e.g., FRs, FCs, and cluster faults), we should enter the hard repair phase to deal with these faults. In this phase, we can use a suitable RA algorithm (e.g., the essential spare pivoting (ESP) algorithm for allocating spares.)

![Fig. 4.2. Test and repair flow of the PTR stage.](image)

![Fig. 4.3. Test and repair flow of the OLTR stage.](image)
According to the analysis results, SRs/SCs can be used to replace FRs/FCs. The addresses of FRs/FCs are stored for address remapping. If the incorporated redundancies are sufficient to repair all FRs/FCs, the memory array can be repaired successfully. That is, the memory array can be shipped or subjected to normal access. However, if all spares are used and there are still faulty memory rows/columns, the memory cannot then be repaired successfully and should be discarded directly. Since most of the SCFs are repaired by ECC, the probability of repair fail in this phase can be reduced greatly. The test and repair flow of the OLTR stage is shown in Fig. 4.3. It includes the concurrent fault detection (CFD) phase, the fault deactivation (FD) phase, and the FDC phase.

<table>
<thead>
<tr>
<th>Evolution and Operation</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Codeword</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1st Read Out Erroneous Codeword</td>
<td>0000 1100</td>
</tr>
<tr>
<td>Complemented Erroneous Codeword</td>
<td>1111 0011</td>
</tr>
<tr>
<td>Write Back</td>
<td>1111 0011</td>
</tr>
<tr>
<td>2nd Read Out Codeword</td>
<td>1111 1111</td>
</tr>
<tr>
<td>Syndrome</td>
<td>1111 0011</td>
</tr>
<tr>
<td>Complement $h_1$ and $h_2$ of the Erroneous Codeword</td>
<td>0000 0000</td>
</tr>
</tbody>
</table>

**EXAMPLE CODEWORD CONTAINING TWO PERMANENT FAULTS**

In the CFD phase, single errors (either permanent faults or soft errors) can be corrected, and double errors can be detected by the inherent correction/detection feature of the SEC-DEC ECC. However, if the ECC detects double errors, we should enter the FD phase. In this phase, we complement the erroneous codeword and write it into the memory array again. Thereafter, we read the complemented codeword again and compare it with the original erroneous codeword. Based on the results of the comparison operation, we then enter the FDC phase. In this phase, the error types are identified first. If the readout complemented codeword still contains double errors, we can conclude that there are two permanent faults. Based on the comparison results, we can just complement the two bits, which are identical in the original erroneous codeword and the complemented codeword to correct the double errors.

**5. DESIGN AND IMPLEMENTATION**

The following steps to implements

**Step 1:** Read the original image matrix [OR]

**Step 2:** Construct the Binary Matrix [BM] and Grayscale Matrix [GSM] based on the following steps

**Step 3:** Compare each pixel in the matrix [OR] with the previous pixel in the same matrix as shown in below.

Step 4: The binary matrix elements are calculated as follows:

$$[BM]_{k,j} = \begin{cases} 0 & \text{if } [OR]_{k,j} = [OR]_{k,j-1} \\ 1 & \text{otherwise} \end{cases}$$

Step 5: First element in [GSM] is set to be equal to the value of the first pixel of [OR]

Step 6: The rest of the elements of [GSM] are calculated as follows:
Step 7: The original image can be reconstructed as follows:

\[
\text{[GSM]}_k = \begin{bmatrix}
\text{nul} & \text{if} & [\text{OR}]_{i,j} = [\text{OR}]_{i,j+1} \\
[\text{OR}]_{i,j} & \text{otherwise}
\end{bmatrix}
\]

5.1 PERFORMANCE PARAMETERS

An example is shown in Table I. In this case, we assume that the 8-bit codeword \((b_7b_6b_5b_4b_3b_2b_1b_0)\) is affected by two stuck-at-1 faults at bit positions \(b_2\) and \(b_3\). The correct codeword is “0000 0000,” as shown in the second row. Due to these two stuck-at faults, the readout erroneous codeword becomes “0000 1100,” as shown in the third row and can be detected by the SEC-DED ECC. However, these two faults are uncorrectable by merely using SEC-DED ECC. Therefore, we complement the erroneous codeword and get the complemented codeword, as shown in the fourth row. We write back it into the memory and read the codeword again. Due to the effects of these two faults, the second readout codeword becomes “1111 1111.” We conduct an XOR (compare) operation for the first and the second readout codeword and get the syndrome, as shown in the seventh row. The second and third syndrome bits are both zero. We can conclude that \(b_2\) and \(b_3\) contains stuck-at faults. Therefore, we can make a simple complement operation for \(b_2\) and \(b_3\) of the first readout codeword to get the correct codeword “0000 0000,” as shown in the eighth row. Therefore, these two stuck-at faults can be corrected.

Alternately, if the second readout codeword contains only one error, it is necessary that there is only one permanent fault. The second faulty bit in the original codeword is due to the effect of a soft error, since we cannot find its effect from the complemented codeword. Based on the comparison.
We assume that the BIST, BIRA, remapping circuit, spare memories, and ECC are mandatory modules, which should be inherently designed into the memories. This assumption is practical, since today’s large size embedded memories incorporate these techniques to enhance the reliability and yield. Therefore, we only have to add the extra EEC module for implementing the proposed EBISR technique. The internal architecture of the EEC module is shown in Fig. 5.2. The sizes of the components in the EEC module depend on the code-word length. For two memories with different sizes and the same codeword length, there EEC modules are basically the same. We use the TSMC 90-nm 1P9M technology to synthesize three memory sizes, 256 × 352 × 22, 256 × 624 × 39, and 256 × 1152 × 72, with the EBISR technique incorporated. The codeword lengths are 22, 39, and 72, respectively. The number of code words of each memory is 4096.

6. RESULT
In this Error correction code (ECC) and built-in self-repair (BISR) techniques by using redundancies have been widely used for improving the yield and reliability of embedded memories are designed using Xilinx.

7. CONCLUSION
Error correction code (ECC) and built-in self-repair (BISR) techniques by using redundancies have been widely used for improving the yield and reliability of embedded memories. The target faults of these two schemes are soft errors and permanent (hard) faults, respectively. In recent works, there are also some techniques integrating ECC and BISR to deal with soft errors and hard defects simultaneously. This design is achieved by a Xilinx software.

8. REFERENCES
[1] S. E. Schuster, —Multiple word/bit line redundancy for semiconductor memories, *IEEE*


