INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume3, Issue3)
Available online at www.ijariit.com

FPGA Implementation of Built In Self Repair Technique for Hard & Soft Errors in Memories

Sojy Yacob
Viswajyothi College of Engineering and Technology, Vazhakulam, P.O, Muvattupuzha, Ernakulam, Kerala
sojyyacob@gmail.com

Anu C. Kunjachan
Viswajyothi College of Engineering and Technology, Vazhakulam, P.O, Muvattupuzha, Ernakulam, Kerala
anuck_10@yahoo.com

Abstract*: Anything that changes the normal operation can be defined as an error in a memory. They can be of two types: - Soft errors & Hard errors. Soft errors are caused by external elements (such as radiation environments, electrical noises etc.) outside of the designer's control. Hard errors are mainly due to manufacturing defects or due to unknown sources. Since an error in the memory can affect the whole system, error detection & correction techniques are important. Our objective is to detect and correct errors in offline stage and in field usage. During offline test stage, BIST (Built-in-Self-Test) module will check for the errors in the entire memory. If any fault is detected, SEC-DED (Single Error Correction Double Error Detection) module will be initiated to correct one error. If there exists more than one error, reconfiguration module will remap the faulty cells with the spare memory cells. During the field usage, if any error occurs, the controller will categorize the errors to three types and apply different techniques to correct them. The whole system is implemented in FPGA and simulation results are noted.*

Keywords: BIST - Built-In Self-Test, SEC-DED - Single Error Correction Double Error Detection, FPGA - Field Programmable Gate Array, BISR - Built In Self Repair, BIST - Built-In Self-Test.

I. INTRODUCTION

Embedded memories play vital role in the semiconductor market because the system-on-chip market is booming and almost every system chip contains some types of embedded memories. The failures often happened to memory chips can be generally classified into two main classes such as those lead to hard errors and those too soft errors. The testing of embedded memories is more difficult than testing independent memories, in general, unless some built-in self-diagnosis techniques are used. In this paper, the memories are tested before and after field usage namely offline and field usage stages. The offline test is done after the memory manufacturing, here the system will check for errors and the ultimate solution of this stage is remapping of the faulty locations with the new spare ones. The field test is carried out when the memory is in an application. If there exist some errors during the field usage, according to the error type solution is made.

II. FLOW DIAGRAM OF OFFLINE TEST STAGE

The test and repair flow of offline test is shown in Fig. 1. 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 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 SEC-DEC ECC code is used to repair all the permanent SCFs. If there exist more than one errors, SEC-DED cannot correct them. Then the remapping circuit is activated to map the faulty cells with the new spare cells. Then the BIST operations are resumed to perform the remaining tests. Since the SCFs occupy the largest proportion of all the possible fault types, correcting them by ECC can increase the efficiency of space usage, and therefore, the fabrication yield and reliability can be raised significantly. Assume that during the very short BIST operation time, the probability of SEU attacks is very low. Therefore, during the offline stage, we do not have to consider the effects of soft errors, and the inherently incorporated ECC can be merely used for correcting SCF.

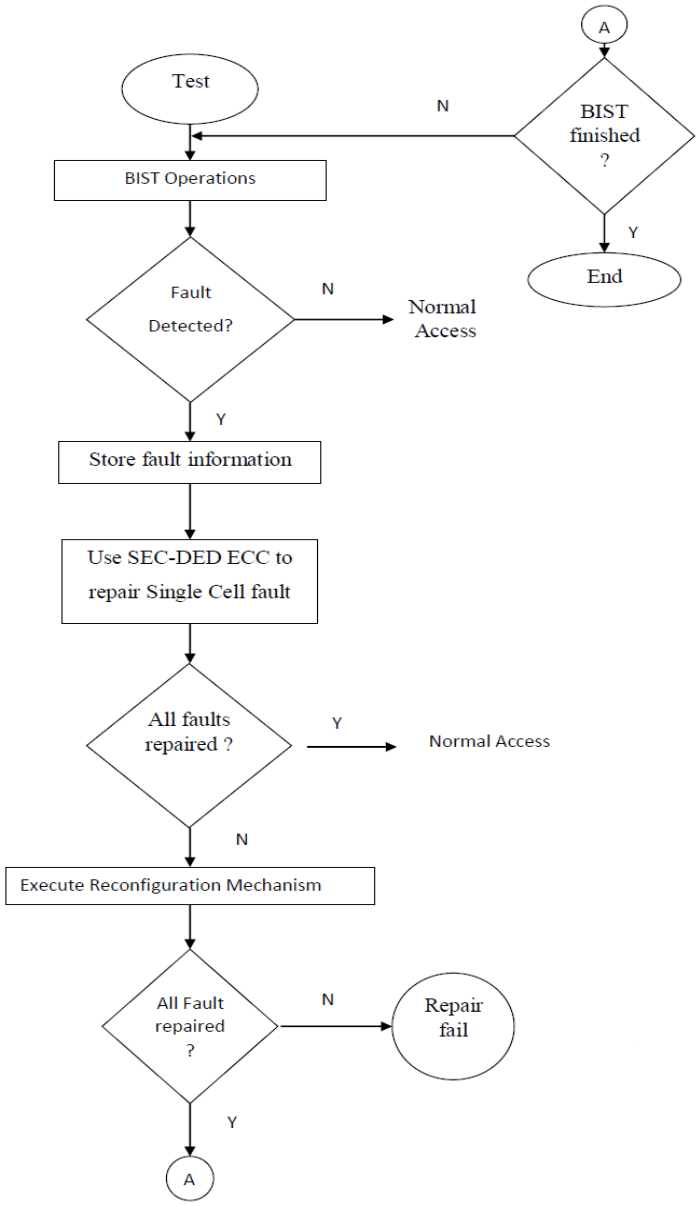

Fig. 1 Offline test and repair flow

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 spares for faulty cells. 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.

III. FLOW DIAGRAM OF FIELD TEST STAGE

The field test and repair flow are shown in Fig.2. It includes the fault detection (FD) phase, the fault deactivation (FD) phase, and the FDC (fault discrimination and correction) phase. In the FD 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.

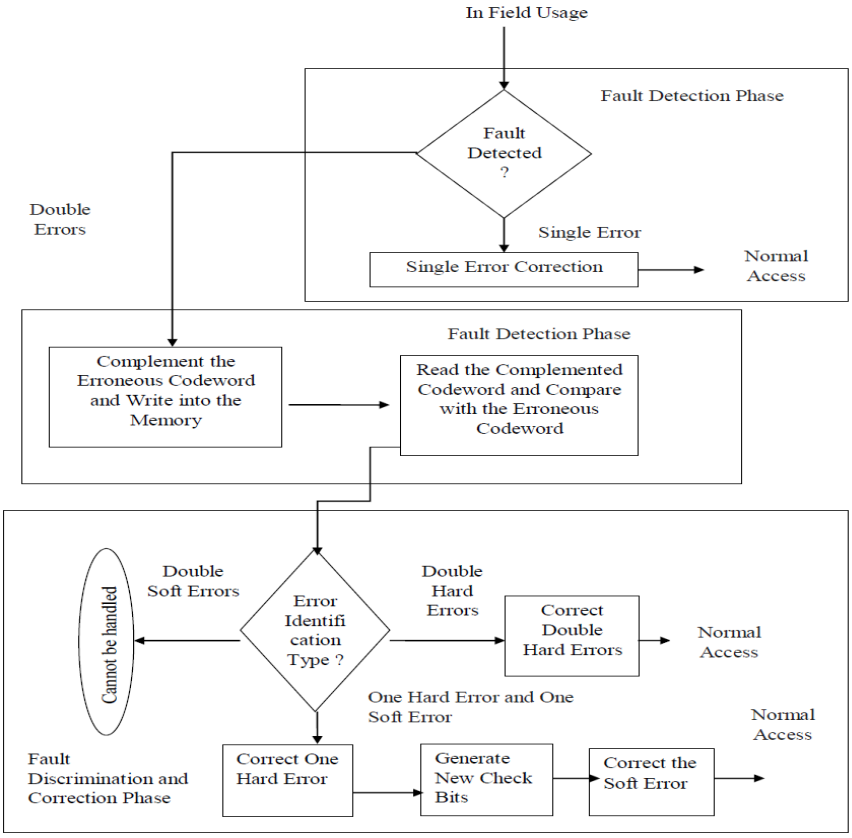

Fig. 2 The field test and repair flow

In this phase, we complement the erroneous code word and write it into the memory array again. Thereafter, we read the complemented code word again and compare it with the original erroneous code word. 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 code word 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 code word and the complemented code word to correct the double errors.

An example is shown in Table 1. In this case, assume an 8-bit code word (b7b6b5b4b3b2b1b0) is affected by two stuck-at-1 faults at bit positions b2 and b3. The correct code word is "0000 0000," as in the second row. Due to these two stuck-at faults, the readout erroneous code word will be "0000 1100," as in the third row and can be detected by the SEC-DED ECC. However, these two faults cannot be corrected by merely using SEC-DED ECC. Therefore, we complement the erroneous code word to get the complemented code word, as in the fourth row. We write back this to the memory and read the code word again from it.

Due to the effects of these two faults, the second readout code word will be "1111 1111." We conduct a XOR (compare) operation for the first and the second readout code words to get the syndrome, as in the seventh row. The second and third syndrome bits are zero in this case. Thus b2 and b3 contains stuck-at faults. Therefore, we will do a simple complement operation for b2 and b3 of the first readout code word to get the correct code word "0000 0000," as in the eighth row. Hence, these two stuck-at faults can be corrected.

Alternately, if the second readout code word contains only one error, it is a must that there is only one permanent fault. The second faulty bit in the original code word is due to the effect of a soft error since its effect is missing from the complemented code word. Based on the syndrome generated from the comparison results of the two readout code words, we need to complement the bit, which has the same value in these two code words to correct the permanent fault. Based on the corrected bit value, new check bits can be generated for the code word containing only the soft error. Then SEC-DED ECC can be reapplied to correct this soft error.

TABLE 1
EXAMPLE CODE WORD CONTAINING TWO PERMANENT FAULTS

Evolution and Operation	Codeword
Correct Codeword	0000 0000
1^{st} Read Out Erroneous Codeword	0000 1100
Complemented Erroneous Codeword	1111 0011
Write Back	1111 0011
2^{nd} Read Out Codeword	1111 1111
Syndrome	1111 0011
Complement b_2 and b_3 of the Erroneous Codeword	0000 0000

An example of this case is shown in Table 2. Here assume that *b*3 consists of a stuck- at-1 fault, and *b*2 is affected by a soft error. The correct code word is "0000 0000," as in the second row. Due to the effects of these two faults, the readout erroneous code word will be "0000 1100," as in the third row, which can be detected by the SEC-DED ECC. These two faults are still cannot be corrected by only using ECC. We can complement this erroneous code word and get the complemented code word, as in the fourth row. Write back it into the memory and read out again from the memory.
Since the effect of the soft error is eliminated during the write back and readout operations, the second readout code word will be "1111 1011."Again a XOR (compare) operation will be done for the first and the second readout code words to get the syndrome "1111 0111," as in the seventh row. Since the third syndrome bit is zero so that we can conclude that *b*3 as the faulty bit containing the stuck-at fault. Then complement *b*3 of the first readout code word to get the code word containing only the soft error, as in the eighth row. Then update the check bits after bit *b*3 is complemented. Hence, the final soft error can then be corrected by the SEC-DED ECC, as in the ninth row.

TABLE 2
EXAMPLE CODEWORD CONTAINING ONE PERMANENT FAULT AND ONE SOFT ERROR

Evolution and Operation	Codeword
Correct Codeword	0000 0000
1^{st} Read Out Erroneous Codeword	0000 1100
Complemented Erroneous Codeword	1111 0011
Write Back	1111 0011
2^{nd} Read Out Codeword	1111 1011
Syndrome	1111 0111
Complement b_3 of the Erroneous Codeword	0000 0100
Use ECC to Correct the Soft Error	0000 0000

The last case considered in this project is that if there exist no errors detected in the second readout complemented code word, we will conclude that there exit two soft errors in the original code word since their effects are eliminated in the complemented code word. Since we do not have the information about the locations of the soft errors, we will not make any correction operations. Hence it is required to retrieve this word from the next level of memory hierarchy to cure these errors.
An example of this case is shown in Table 3. Here, we assume that b2 and b3 are affected by soft errors. The correct code word will be "0000 0000," as in the second row. Due to the effects of these two soft errors, the readout erroneous code word is "0000 1100," as in the third row, which can be detected using the SEC-DED ECC. We cannot correct them by using the ECC only. We then complement this erroneous code word and get the complemented code word, as in the fourth row. Write back it into the memory and read out again from it. Since the effects of these two soft errors are eliminated during the write back and readout operations, the second readout code word will be "1111 0011." We then conduct a XOR (compare) operation for the first and the second readout code words to get the syndrome "1111 1111," as in the seventh row. The value of all syndrome bits is 1. Hence we will conclude that the original erroneous code word contains two soft errors. This case cannot be corrected by the SEC-DED ECC, and we want to retrieve this word from the lower level of memory hierarchy.

TABLE 3
EXAMPLE CODEWORD CONTAINING TWO SOFT ERRORS

Evolution and Operation	Codeword
Correct Codeword	0000 0000
1^{st} Read Out Erroneous Codeword	0000 1100
Complemented Erroneous Codeword	1111 0011
Write Back	1111 0011
2^{nd} Read Out Codeword	1111 0011
Syndrome	1111 1111
Retrieve the Codeword from the Lower Level of Memory Hierarchy	0000 0000

IV. HARDWARE ARCHITECTURE OF EBISR

A. Block Diagram Of Offline Test Stage

Fig. 3 shows the block diagram of the EBISR architecture. It mainly consists of the BIST module, the ECC encoder/decoder, the error identifier, the memory array, the spare memory, the remapping circuit, and the controller. The BIST module generates the test patterns to test the memory array and the spare memory during the offline stage when the *BIST*_ON signal is asserted. This signal is activated by either the automatic test equipment (ATE) in the offline test and repair phase.

When the BIST module detects any faults in the memory, the BIST operations will be suspended. The fault information is sent to the controller module through the Faulty_addr and the Faulty_synd signals. According to the received syndromes, SCFs are not corrected by spares and are subjected to the correction capability of the ECC during online operations. The remapping circuit executes the remapping only for memory rows/columns containing multiple faults and allocates SRs/SCs to repair them. After we finish the space allocation operations, the BIST operations will be resumed unless the whole memory array is tested.

When the BIST completely tests the memory array, the remapping circuit will check whether the main memory is repairable or not by using the incorporated spares. If it is repairable, then it transfers the remapping information through the Remapping_addr and the Remapping_wen signals to the remapping circuit for address remapping. The remapping circuit then remaps the faulty addresses to their corresponding spare memory addresses. Otherwise, the remapping module will assert the Repair_fail signal to the BIST module, which then sends out the Fail signal to the external ATE or the on-chip processor to indicate that the memory cannot be repaired successfully.

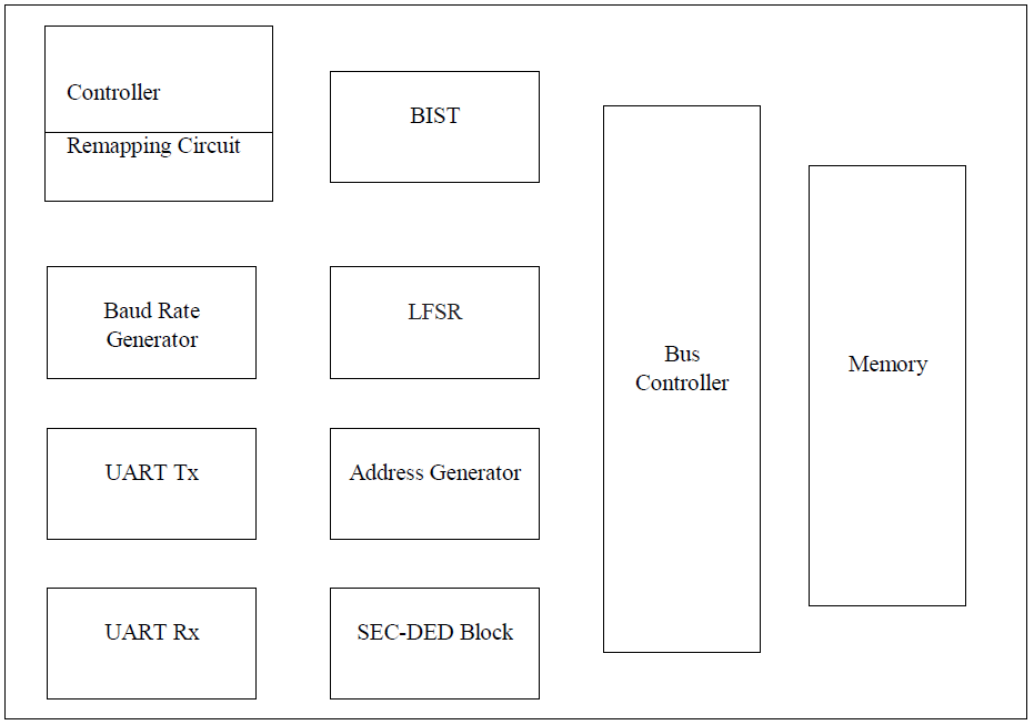

Fig. 3 Block diagram of offline test stage

B. Block Diagram Of Field Test Stage

The field test is carried out when the memory is used for an application. After receiving the DEC_Start signal sent from the ECC decoder, the error identifier then stores the erroneous code word containing two faulty bits into the ECR. This erroneous code word is complemented by the complementary and write into the memory again. We then read out the complemented code word and generate the syndrome by the syndrome generator. The syndrome is then processed by the error identifier to discriminate the fault types. Based on the identified fault type, we can perform suitable error correction operations, as described in Fig.3.4, by the error corrector to recover the original code word.

During the normal accesses of the memory array, the operations of the ECC encoder/decoder are similar to the conventional designs. As described above, if the ECC is used to correct SCFs, we should cure the situation when another hard fault or soft error attacks the same code word. Therefore, the extra error identifier module is used to conduct the required operations described in Fig. 3.4 (e.g., complement, syndrome generation, and write back) for discriminating error types and correct double errors.

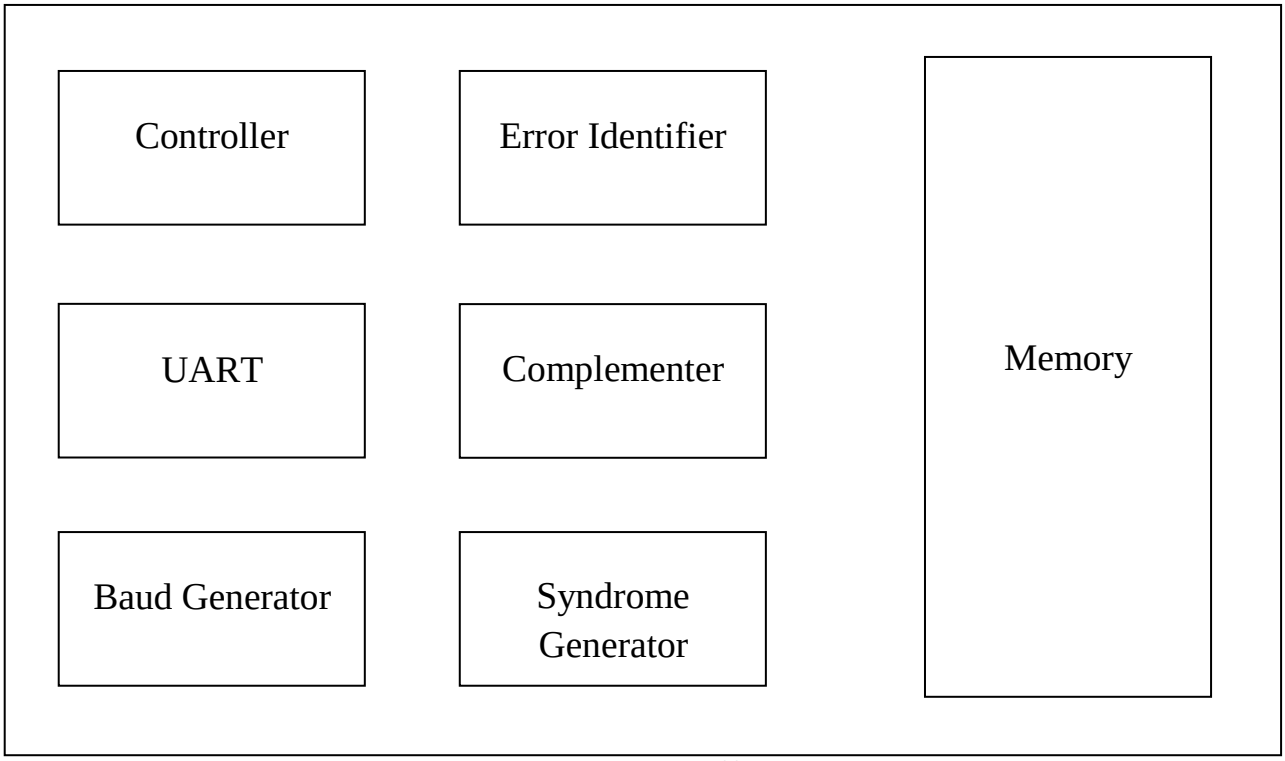

Fig. 4 Block diagram of field test stage

During a read operation, if the *Data_Output* signals from the memory array contain only one faulty bit (either affected by a hard fault or a soft error), it can be corrected by the ECC decoder as usual. Alternately, if there are two faulty bits in the *Data_Output*, the ECC decoder can detect double faults and assert the *DEC_Start* signal to the error identifier module, as shown in Fig. 4.

V. RESULTS

This section contains the simulation results obtained in MATLAB and Isim, and implementation result obtained in FPGA.

A. SEC-DED Encoder And Decoder Results

The SEC-DED encoder and decoder are implemented in FPGA. The simulation results of SEC- DED encoder and decoder are shown in Fig. 5 and 6 respectively. The extended Hamming code is used as the SEC-DED code here.

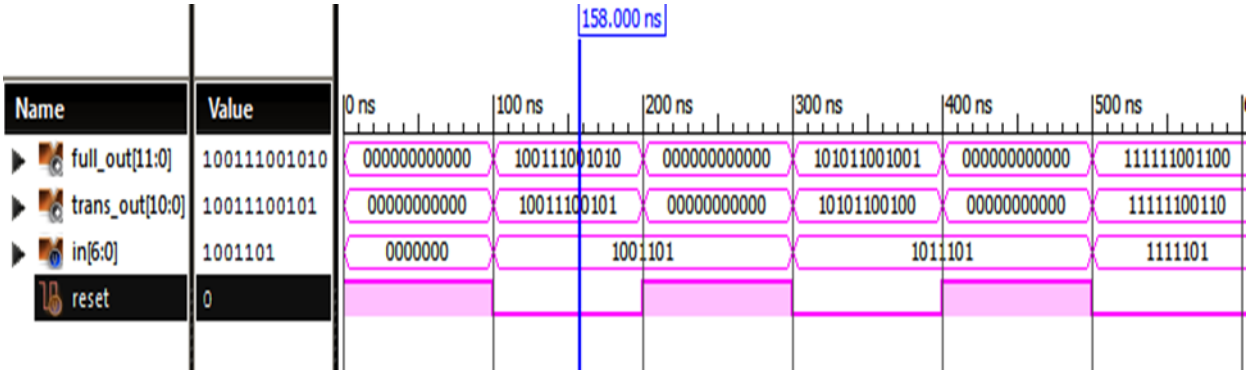

Fig. 5 Simulation result of SEC-DED Encoder

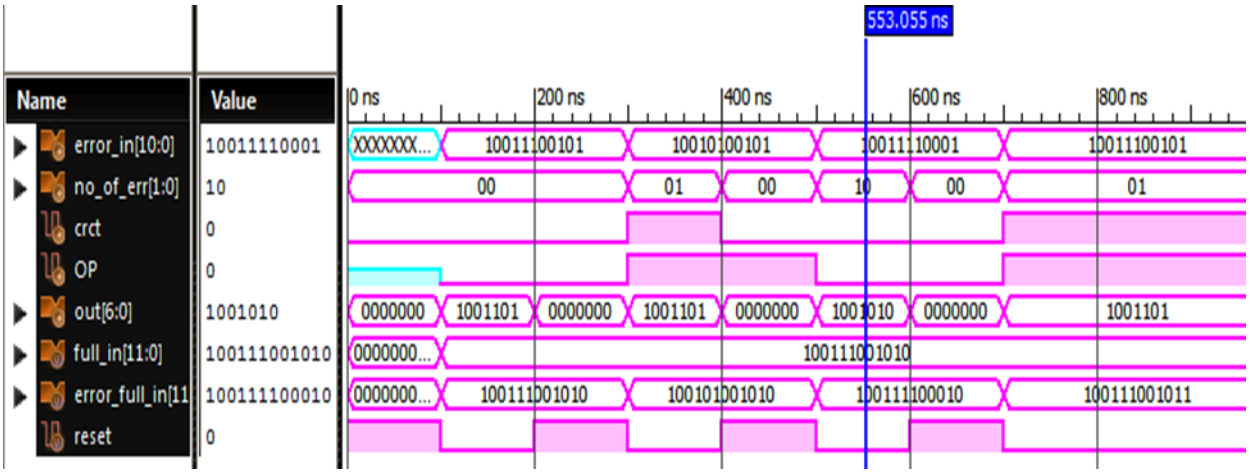

Fig. 6 Simulation result of SEC-DED Decoder

B. Offline Test Stage Results

In the offline stage, the BIST will check for the errors. If there exist no errors the memory will be given to normal usage. If any fault is detected, the SEC-DED module will be invoked to repair the single cell faults. SEC-DED will detect double errors and corrects only one error. Thus if the SEC-DED module cannot correct all the errors, remapping module will be initiated and the faulty cells are reallocated to spare memory locations. After reallocation, still there exits errors, the memory will not be used for any applications.

The simulation results are shown in Fig. 7. Matlab is used to display the error locations and remapped locations as shown in Fig. 8. Error is injected into certain memory locations and results are noted

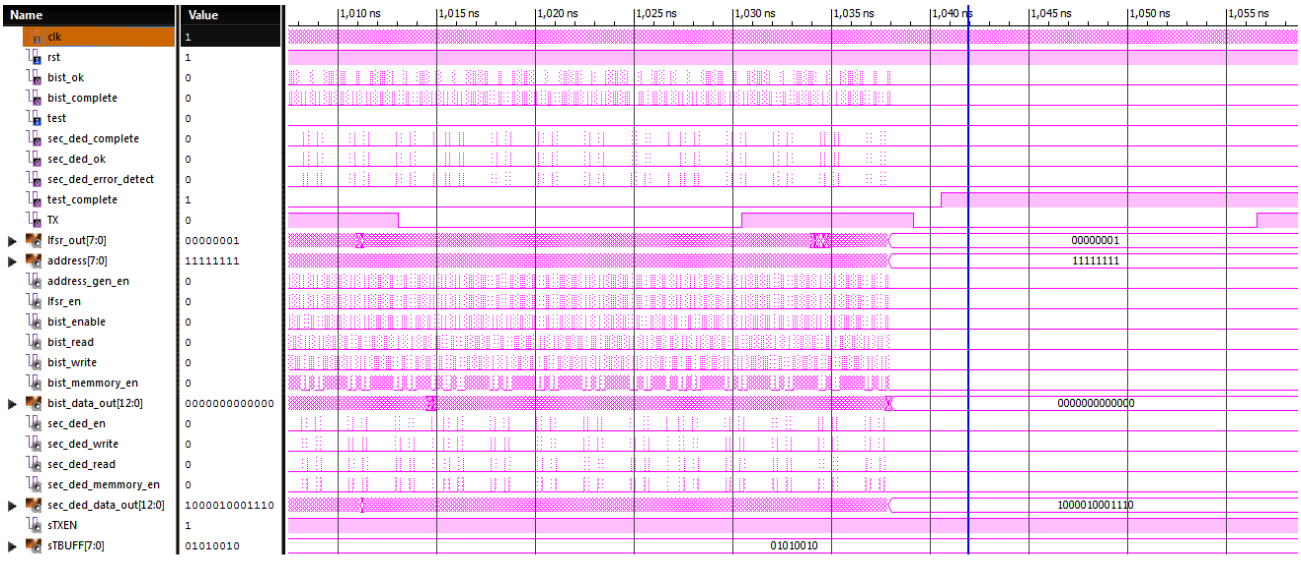

Fig. 7 Simulation results of offline test stage

.


```
                built in Self Test

Press Reset button of the FPGA
                              done

press test button to start the test

Enter 1 to get Details1

Bist Error found and Locations are :
8   9   12  13  24  25  28  29  40  41  44  45  56  57  60  61
72  73  76  77  88  89  92  93  100 104 105 108 109 120 121 124
125 136 137 140 141 152 153 156 157 168 169 172 173 180 184 185
188 189 200 201 204 205 216 217 220 221 232 233 236 237 248 249
252 253


Sec-ded cannot correct at Locations :
100 180


remaped location are :
100 : 255       180 : 254
```

Fig. 8 Matlab result of offline test stage

C. Field Test Results

In the field test, if any fault is detected then the single errors will be corrected by the SEC-DED module and if more than single errors, fault discrimination module is initiated. In the fault discrimination, errors categorized into double soft errors, double hard errors and with one hard and one soft error. All these three errors are treated separately to correct them. Fig 9 shows the Matlab results of field test stage. Here the location 67 is set to have one permanent and one soft error. 34 is written into that location. Even though there exit errors, correct value will be received after reading.

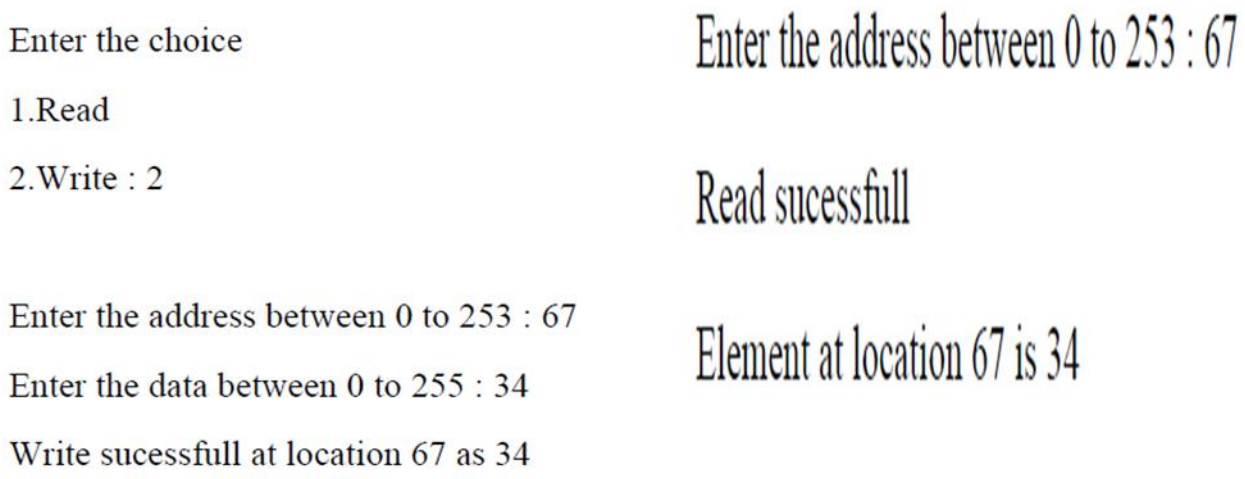

Fig. 9 Matlab results of field test stage

D. Implementation Results In FPGA

The offline test stage and the field test stage are implemented in the FPGA. In the offline test stage, BIST will check for the errors and the SEC-DED code is used to correct only one error. For more than one error, the faulty memory cell will be remapped to the spare memory. With the help of UART and Matlab the address locations and remapped locations are displayed. The design summary of offline test stage is shown in Fig. 10. Mimas Spartan 6 FPGA and Xilinx ISE Design suit 14.7 is used for Verilog programming and implementations.

memmory_test_top Project Status (04/30/2017 - 13:17:00)			
Project File:	bist_xilinx.xise	**Parser Errors:**	No Errors
Module Name:	memmory_test_top	**Implementation State:**	Programming File Generated
Target Device:	xc6slx9-2tqg144	**• Errors:**	
Product Version:	ISE 14.7	**• Warnings:**	
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Performance Summary			
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Fig. 10 Design Summary of offline test stage

The corrected memory is used for field applications. But when the memory is in field usage, there are chances to have errors. If any fault is detected, one of them will be corrected by the SEC-DED module and the other one will be corrected on the basis of its type. This reduces the spare memory usage and thus decreases hardware complexity. The design summary of field test stage is shown in Fig. 11.

memmory_field_top Project Status			
Project File:	bist_xilinx.xise	**Parser Errors:**	No Errors
Module Name:	memmory_field_top	**Implementation State:**	Programming File Generated
Target Device:	xc6slx9-2tqg144	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Fig. 11 Design Summary of field test stage

The two test stages are implemented in Mimas Spartan 6 FPGA. UART module is used to communicate with the FPGA to display the results in the screen.

CONCLUSION AND FUTURE SCOPE

The EBISR technique is successfully implemented, which integrates both ECC and hard repair techniques to increase the fabrication yield and reliability of embedded memories. Based on the proposed EBISR techniques, the reliability will not be compromised if the ECC is also used to correct permanent faults. The EBISR technique can be applied during the offline test stage and the field test stage. In the offline test stage, BIST and LFSR will check for the errors. If any error exits SEC-DED module is activated and correct one error. The extended Hamming code is used as the SEC-DED code. After SEC-DED stage if there exit errors, reconfiguration mechanism will reallocate the faulty cells with spare cells. In the field test stage, the SEC-DED code is used to correct the errors and if more than one error exits, categorize the errors into three types and apply different schemes. The EBISR is implemented in the FPGA and simulation results are noted. The error details are shown with help of Matlab. The proposed technique can be easily integrated into the conventional BISR architecture. This project is mainly focusing on single cell faults and it can be extended to the detection and correction of multiple cell upsets.

REFERENCES

[1] SHYUE-KUNG LU, CHENG-JU TSAI, AND MASAKI HASHIZUME *(2016)* ENHANCED BUILT-IN SELF-REPAIR TECHNIQUES FOR IMPROVING FABRICATION YIELD AND RELIABILITY OF EMBEDDED MEMORIES, *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 24*

[2] Jaeyong Chung, Joonsung Park, Jacob A. Abraham (2013) A Built-In Repair Analyzer With Optimal Repair Rate for Word-Oriented Memories, *IEEE Trans. On VLSI Systems*, Vol. 21, no. 2.

[3] Jing Guo, Liyi Xiao, Zhigang Mao, Qiang Zha (2014) Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 22, no. 1.

[4] P. Oehler, S. Hellebrand, and H.-H.Wunderlich (2007) An integrated built-in test and repair approach for memories with 2D redundancy, *in Proc. Eur. Test Symp*., 91–96

[5] P. Oehler, A. Bosio, G. D. Natale, and S. Hellebrand (2008) A modular memory BIST for optimized memory repair, *Proc. Int. On-Line Test.* Symp.,171–172
[6] W. Jeong, I. Kang, K. Jin, and S. Kang (2009) A fast built-in redundancy analysis for memories with optimal repair rate using a line-based search tree, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 17, no. 12, 1665–1678
[7] Costas Argyrides, Dhiraj K Pradhan, Taskin Kocak (2011) Matrix-based codes for Reliable and Cost Efficient Memory Chips, *IEEE Trans. On VLSI Systems*, vol. no. 19, no. 3
[8] C. Argyrides, R. Chipana, F. Vargas, and D. K. Pradhan (2011) Reliability analysis of H-tree random access memories implemented with built-in current sensors and parity codes for multiple bit upset corrections, *IEEE Trans. Rel., vol. 60, no. 3, pp. 528–537*
[9] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro (2012) Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement, *IEEE Trans. Device Mater.Rel.*
[10] Reviriego, M. Flanagan, J. A. Maestro (2012) A (64, 45) triple error correction code for memory applications, *IEEE Trans. Device Mater.Rel.*, vol.12, no.1,101–106
[11] S. Hamdioui, G. Gaydadjiev, and A. J. van de Goor (2004) The state of the art and future trends in testing embedded memories, *IEEE Int. Workshop Memory Technol., Design Test. (MTDT)*, pp. 54–59.

