International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 14 (2017) pp. 4098-4105

Block Sparse and Addressing for Memory BIST Application

Mohammed Altaf Ahmed[1], D Elizabath Rani[2] and Syed Abdul Sattar[3]

[1]Dept. of Electronics & Communication Engineering, GITAM Institute of Technology, GITAM University, Visakhapatnam, A.P, India.

[2]Dept. of EIE, GITAM Institute of Technology, GITAM University, Visakhapatnam, A.P, India.

[3]Dept. of ECE, Royal Institute of Technology and Sciences. Hyderabad, A.P, India.

[1] ORCID: orcid.org/0000-0003-0355-7835

Abstract

Large volumes of data storage and data exchange are observed in current applications. With the developments of new devices, the information is captured or retrieved at very high rate. During the process of high volume memory applications, occurrence of fault data retrieval is commonly observed, wherein these faults are introduced by different means, and a large probability of fault condition is seen at memory locations. Memories are the prime storage elements in all digital devices. Data are addressed, stored and fetched in/from these memory elements. As information is stored in these memory locations, the reliability or fault tolerance in such device is a prime requirement. To develop a fault tolerance approach, these memory elements are incorporated with Built in self-testing (BIST) and Built in self-repairing (BISR) logic. However these approaches are operated in Bit or word level. The fault tolerance is achieved by sparsing the faulty bit/word to a temporary sparse location, where the address unit and the sparse memory are buffered. However, the memory reading operation under high fault condition is observed to be an exhaustive process, where each fault address is to be interpreted isolately. This address mapping results in low memory throughput and higher power consumption. To eliminate this problem, a new coding approach to fault memory addressing using redundant addressing is proposed in this chapter. The redundant parameter of the memory addressing is utilized to optimize the testing and reading operation.

Keywords: Memory fault testing, BIST, BISR, redundant address coding, Block sparsing

INTRODUCTION

DUE to rapid growth in VLSI technology, it is possible to integrate several billions of transistors on a single chip. Process variations causes failures like read failure, access failure, hold stability failures and write failure. During the process of high volume memory applications, occurrence of fault data retrieval is commonly observed. Wherein, these faults are introduced by different means, a large probability of fault condition is seen at memory locations. As part of the development of VLSI technology, System-on chip (SoC) is developed which is capable to integrate an entire system on a single chip. It may possible to integrate many processor cores with embedded memory, interconnect infrastructure and application specific circuits embedded on a single chip which reduces the size of the system from a board to chip. SoCs gives high performance and higher reliability at low cost with low power consumption. According to international technology roadmap for semiconductor (ITRS) over 90% of the chip is occupied by embedded memories in System-on chip (SOC) [7]. Testing is a measurement of defects and quality level. A circuit is tested once and for all, with the hope that once the circuit is verified to be fault free it would not fail during its expected life-time, it is known as off-line testing. However, this statement does not correct for modern ICs, based on deep sub-micron technology, because within expected lifetime they may develop faults even during its operation. To solve this problem sometimes redundant circuitry are reserved on-chip which replace the faulty portions. To enable replacement of faulty circuitry, the ICs are tested before each time they startup. If a fault is found, a part of the circuit (having the fault) is replaced with a corresponding redundant circuit part (by re-adjusting connections).Testing a circuit each time before its startup, is called Built-In-Self-Test (BIST). Once BIST finds a fault, there adjustment in connections to replace the faulty part with a fault free one is a design problem. BIST reduces the time to test a chip. Additionally, the violent design rules make the memory arrays prone to faults [4]. Therefore, the memory yield plays a major impact on SoC yield and is dominated in the overall SoC yield, thus, optimizing the memory yield plays a critical role in the SoC environment. For yield improvement, memory arrays are commonly placed with spare elements [13],[1] and externally test equipment has been used to test the memory arrays and configure the allocated spare elements. However, in

the SoC environment, the overall test time is excessively increases if the test response information from the memory arrays is sent to the external testing equipment. On the other hand, the SoC environment, combined with shrinking technology, allows us more space for on-chip test infrastructure at lower cost than earlier, which makes probable a variety of built-in- self-test (BIST) and built-in-self-repair (BISR) techniques for reducing the test time. In this paper, design of dynamic Built-in Self-Repair for Embedded SRAM is proposed. Built-in-Self-repair is used to enhance the yield for embedded memories for effective memory diagnosis and fault analysis. BISR mainly consists of Built-in Self-test, Built-in fault-analysis and Multiplexer (MUX). In the proposed BISR, each fault can be saved only once .The main aim of the proposed BISR to repair a fault using redundancy by forming one–to-one mapping of a faulty location to redundancy location. By dynamic redundancy architecture we can repair more number of faults by replacing even single bit fault with single bit redundancy bit. To present the stated approach the rest of the paper is organized as follows, section II describes the methodologies. The experimental results are illustrated in section III and IV. The past approaches are outlined in discussion section V and the conclusion is outlined in section VI.

METHODOLOGY

A. Built-In Self Repair (BISR) Logic

In order to improve fabrication yield and in-field reliability, built-in self-repair (BISR) is considered a promising solution [15]. The fault tolerance is made to achieve a fault free data access in/from memory units. The BISR logic is used as a monitoring and processing block in memory application for fault tolerance. The conventional model of BISR operates in two modes of operation. The operational modes for an SRAM unit are stated in Table I.

TABLE I
SRAM OPERATION MODES

Modes	Repair Selection	Operation
Test Mode (tes_h=1)	Default: repair (bisr_h=1) Don't repair (bisr_h=0)	Access normal words Repair faults and test Access normal words Test only
Access Mode (test_h=0)	Repair (bisr_h=1) Don't repair (bisr_h=0)	Access normal words Repair faults and write/read SRAM Access normal Redundant and normal words. Write/read SRAM only

The BISR unit operates on test mode or Access mode operation. In the test mode operation, the memory undergoes a fault diagnosis for stuck at zero, or stuck at 1 fault testing. In access mode, SRAM users can decide whether the BISR is used base on their needs. If the BISR is needed, the Normal-Redundant words will be taken as redundancy to repair fault. If not, they can be accessed as normal words.

In the operation of fault condition, each fault address is stored only once into a fault address memory. In the detection process, faulty addresses are detected in more than one step of match operation.

The flows of storing fault addresses are shown in Fig1.The BIST detects whether the current address is faulty. If it is, the BISR checks whether the Fault-address-Memory overflows. If not, the current fault address should be compared with those already stored in Fault-address-Memory. Only if the faulty address isn't equal to any address in Fault-address-Memory, it can be stored. To simplify the comparison, write a redundant address into Fault-address-Memory as background. In this case, the fault address can be compared with all the data stored in Fault-address-Memory, no matter how many fault addresses have been stored, and the BISR strategies are high speed. As shown in Fig 1, once a fault address is stored in Fault-address-Memory, it points to all fault addresses. The fault addresses are retrieved by processing a one-to-one mapping. Using this method, the BISR operates on all the fault location in the address memory, which is an exhaustive search operation. The fault diagnosis in such case takes a larger time for reading and large address memory locations for storage. This overhead leads to low operation efficiency of memory application and result in high power consumption. To overcome these issues, a redundant fault addressing logic is suggested.

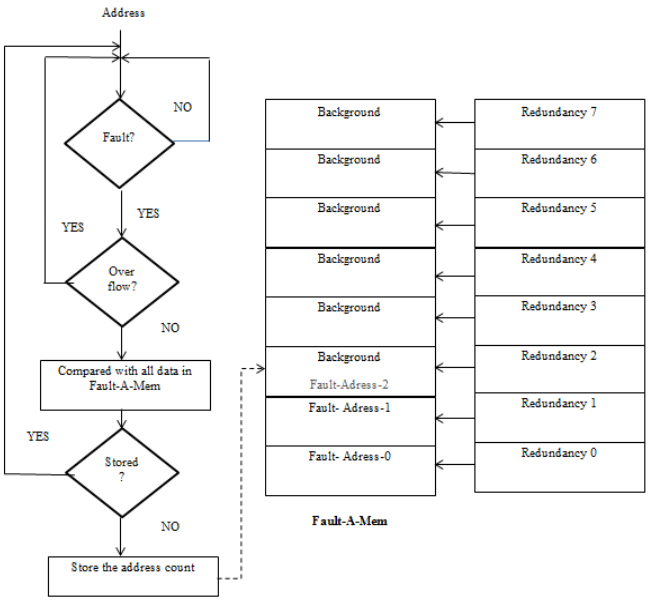


Figure 1. Faulty Addresses stored sequence

B. Redundant Fault Address Mapping

To develop the suggested redundant fault address mapping, a block repair fault tolerant architecture is proposed for main memories (SRAM). In this approach, the main memory cell array will be divided into blocks i.e., 4 bits as a block. This main memory is subjected to test to find the faults. Then the main memory will be repaired by allocating spare memory to the faulty blocks of main memory. The fault diagnosis for the suggested approach is carried out in fault testing and fault mapping operation.

1) Fault testing

In the fault testing operation, the test memory is filled with all 1's the output data out is compared with 1's in block wise so that if any mismatch is found the comparator will set the status signal to be 1 to indicate the fault in the memory location.

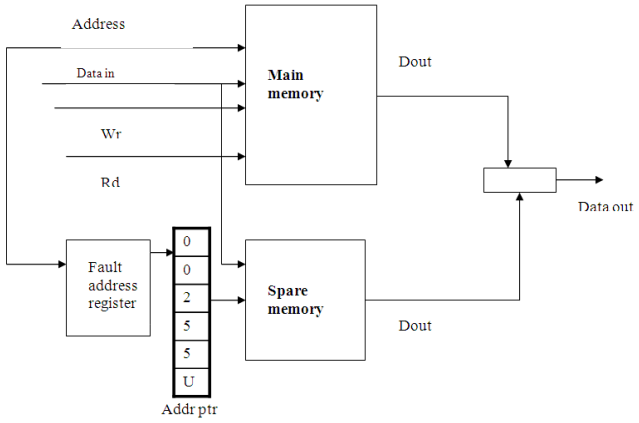


Figure 2: Memory Fault Testing

Then the address of the faulty block memory location is stored in a register. Similarly the main memory is tested for stuck_at_1 faults by filling all 0's into the main memory and comparing it with all 0's in block wise. The faulty addresses of the main memory are buffered and stored into the fault address memory.

2) Fault Mapping

When the test memory is accessed, the address of the memory location to write/read is compared with the address in the faulty address register. If a match is found then data will be sent to spare memory while write operation and the data will be retrieved from spare in read mode. The proposed approach of memory fault testing is shown in Fig 2.

The fault tolerance system operates on the detection of fault location. In the convention fault-address-memory with each fault detected in the test process buffer the fault location in the temporary fault address register of the address memory. In such coding, the fault locations are represented as shown in Fig 3.

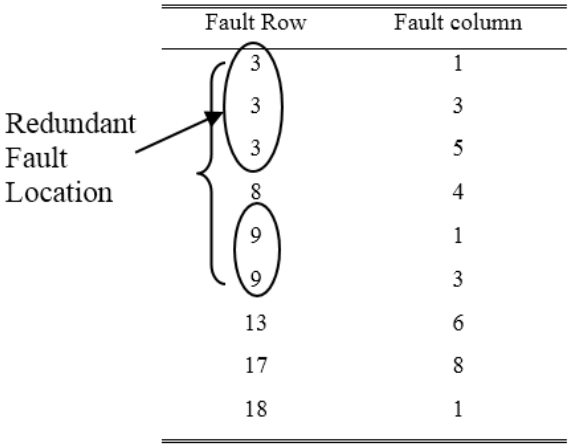


Figure 3: Ilustration of fault address memory in conventional BISR

To overcome the issue of redundancy, a fault redundant counter is set. The counter records the observed faults per location and stores the fault count in fault address memory. The suggested realigned memory unit is as shown in Table II. The operation of the suggested approach under data fetching with fault address logic is presented in Fig 4.

TABLE II
PROPOSED REDUNDANT FAULT ADDRESS MEMORY

Fault Row	Fault count	Fault column
3	3	1,3,5
8	1	4
9	2	1,3
13	1	6
17	1	8
18	1	1

TABLE IV
COMPARISON OF OBSERVATIONS ARE MADE FOR A 6/16 MEMORY

	Bit	Block	Word
Utilization of memory (No. of cells)	80	68	16
No. of spars used	10	7	3
Time taken to complete the simulation	20.1us	6670us	3192us
No of BELs	2926	1800	1127
Frequency(Hz)	83.289	93.868	99.086

At each of the location reading operation, a location address is requested, which is passed to the fault address memory to test

for faulty location. For a true match for fault location, the count value is observed and all the registered address columns are

fetched form the sparse memory to formulate the data byte.

In such case, the fault recovery is observed to be performed in 3 match cycles in comparison to 6 match cycles in conventional fault detection. The operation of fault detection is as illustrated below. A fault memory evaluation on the stated memory logic as shown in Fig 5 is performed.

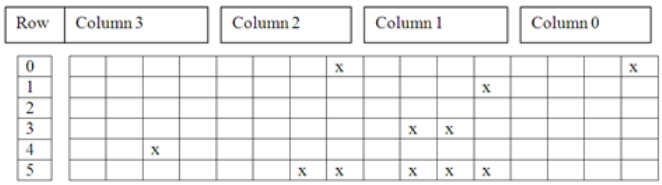


Figure 5: Memory unit with fault locations

In the operation the fault testing is carried out and the testing result is as shown below in Table III, number of spares are 3.

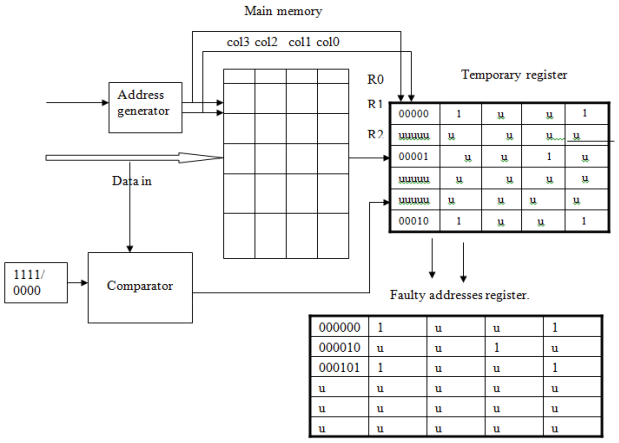


Figure 4: Operational flow for proposed redundant coding

RESULTS

To evaluate the operational functionality and implementation feasibility, a simulation for the proposed approach is made. The obtained results are as illustrated below in Fig 6.

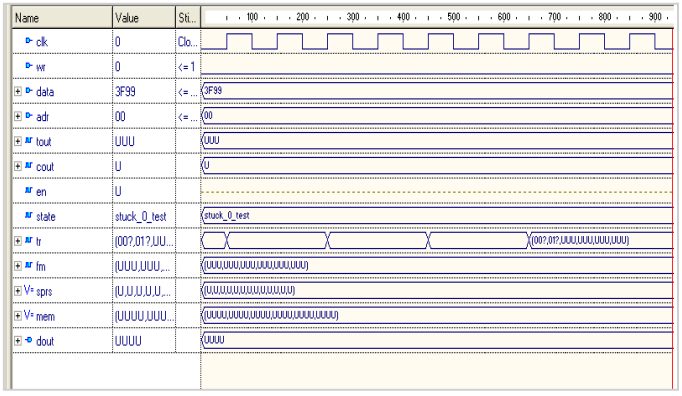
As show in the Table III, we have faults in 0th, 1st, 3rd, 4th and 5th rows. We have taken 3 spares to repair the memory. So in order to repair, the memory in word level, only 3 rows of memory are being repaired and the other 2 rows are getting faulty data. The main disadvantage is that even for a single cell fault in the memory a full row is being replaced where memory is getting waste for a single cell. If we consider the case of repair in bit level, each and every cell of memory is being repaired. But the time taken to repair the memory is high and the hardware used to repair is also large. This is recorded in the Table IV.

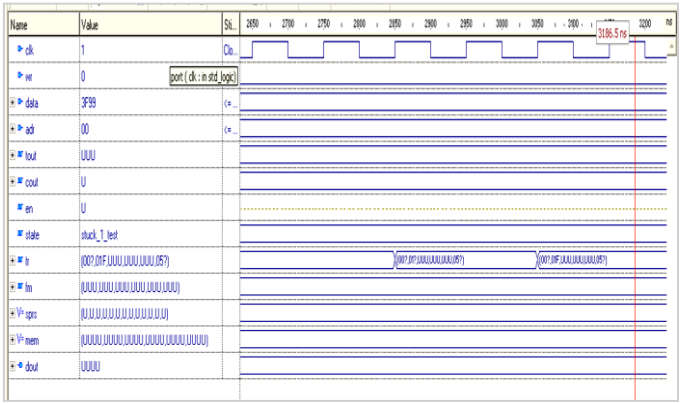
TABLE III
OBSERVATIONS ON THE SELECTION OVER BIT, BLOCK AND WORD LEVEL

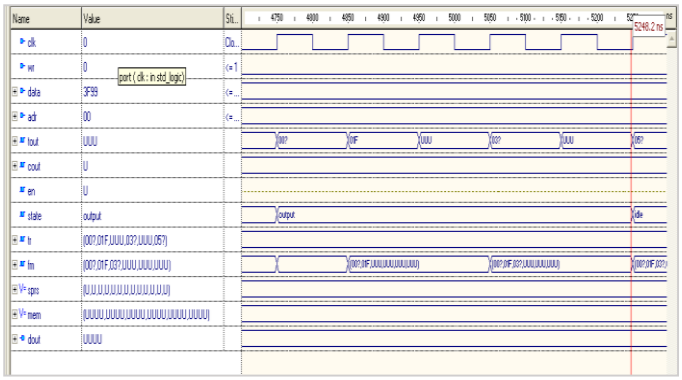
	Bit	Block	Word
0th row	2 cells	2 blocks	Full word
1th row	1 cell	1 block	Full word
2th row	----------No fault----------		
3th row	2 cells	1 block	Full word
4th row	1 cell	1 block	Not repeated
5th row	5 cells	2 blocks	Not repeated

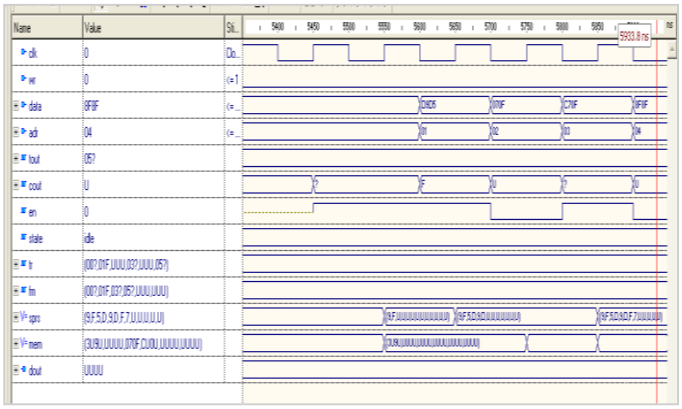
In block level of repairing memory, we can compensate the disadvantages of both word and bit level, instead of full row being replaced for a single fault in the cell. A block is replaced, and if there are more than 2 cells faulty in a block. It is very advantageous to replace that full block into a spare block. The hardware and time taken to repair are also less than that of bit wise repairing.

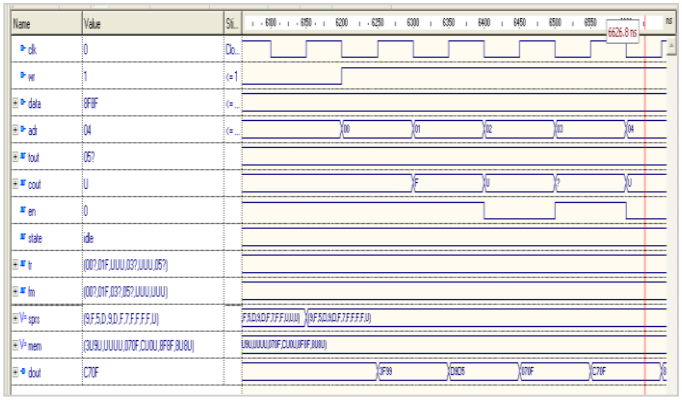
Case 1: Redundancy Added At in Block Level



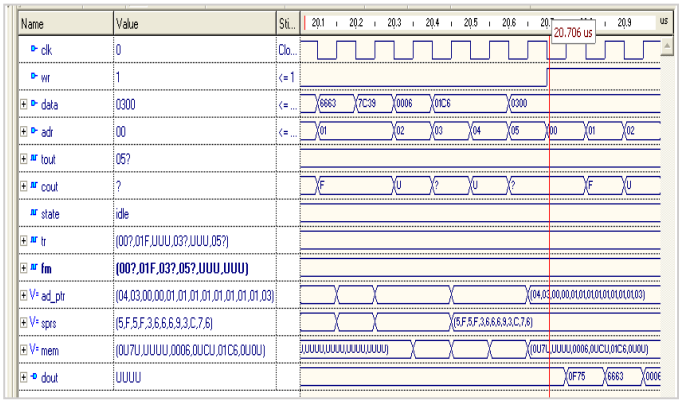




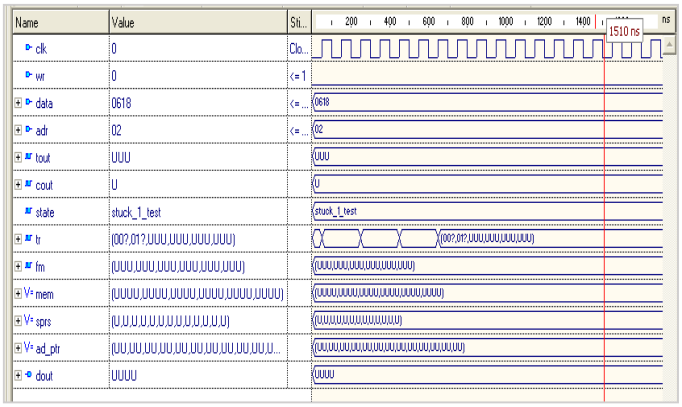




Case 2: Redundancy Added At in Bit Level



Case 3: Redundancy Added At in Word Level



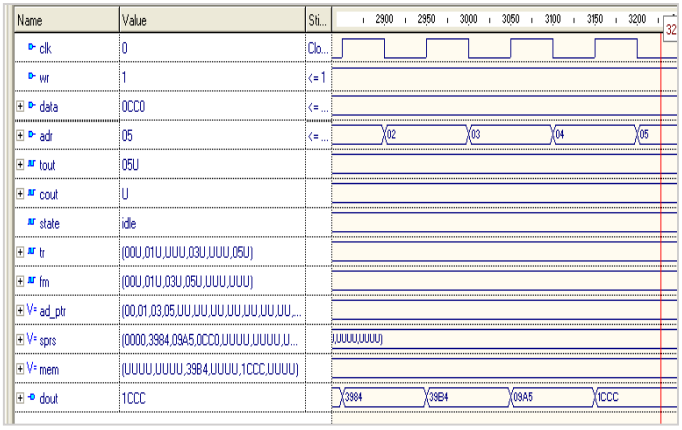


Figure 6: Simulation Results for three cases of Block level, Bit level and Word level.

A fault tolerant mechanism for memory operation is developed. The simulation observations over a memory location with stuck_at_0 and stuck_at_1 are developed and tested. A spare logic is developed with addressing logic to overcome the faults in memory. The data read are found to be exactly same as input data with fault locations. The observations were developed for bit, block & word selections.

FINAL RESULTS

RTL Top Level Output File Name	: scrub.ngr
Top Level Output File Name	: scrub
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: NO
Design Statistics	
# IOs	: 24
Cell Usage:	

# BELS	: 643
# GND	: 1
# INV	: 8
# LUT1	: 66
# LUT2	: 32
# LUT2_D	: 3
# LUT3	: 92
# LUT3_D	: 4
# LUT3_L	: 1
# LUT4	: 169
# LUT4_D	: 35
# LUT4_L	: 19
# MUXCY	: 92
# MUXF5	: 43
# MUXF6	: 15
# VCC	: 1
# XORCY	: 62
# FlipFlops/Latches	: 262
# FD	: 7
# FDE	: 183
# FDRE	: 64
# LDC_1	: 1
# LDE_1	: 7
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 23
# IBUF	: 16
# OBUF	: 7

Minimum period: 38.340ns

(Maximum Frequency: 26.082MHz)

Minimum period: 38.340ns

(Maximum Frequency: 26.082MHz)

Before Clock minimum input arriving time: 9.167ns

After clock minimum output required time: 6.229ns

The synthesis approach is similar to work of the reference of [1][2][14]. The implemented design is targeting to the FPGA device Xc2s50e-ft256-7 of Virtex family. The logic routing, Placement, Floorplan and Bottom packaging all have done on the same device.

The logical routing obtained as shown in Fig 7. This routing is conceded to the Xilinx FPGA editor. The output indicates, the internally connections of different logic blocks and input outputs of FPGA device. This routing technique for the given FPGA designates about the amount of area utilization by the interconnections and the logic implemented in the device. Due to the complexity of the design it is not possible to perform Place and Route manually, only it can be done through the VLSI tool.

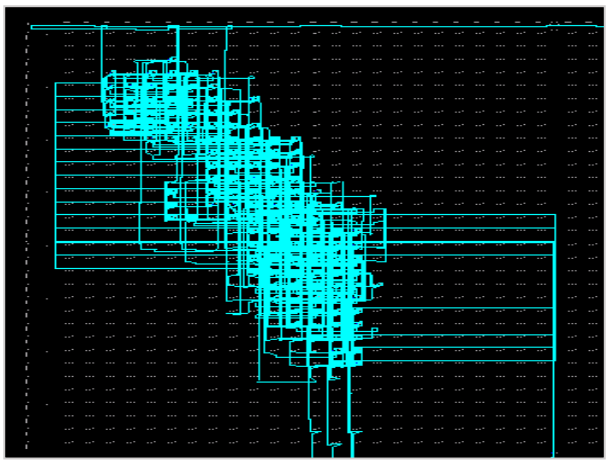


Figure 7: Logical Routing

Logical placement of proposed design targeting to the same device is shown in Fig 8. The logical placement obtained is described about the CLB and its design requirements. It describes about how the components, Logic elements and circuitry are efficiently used in FPGA in order to save the space. This step is performed by Xilinx tool with routing.

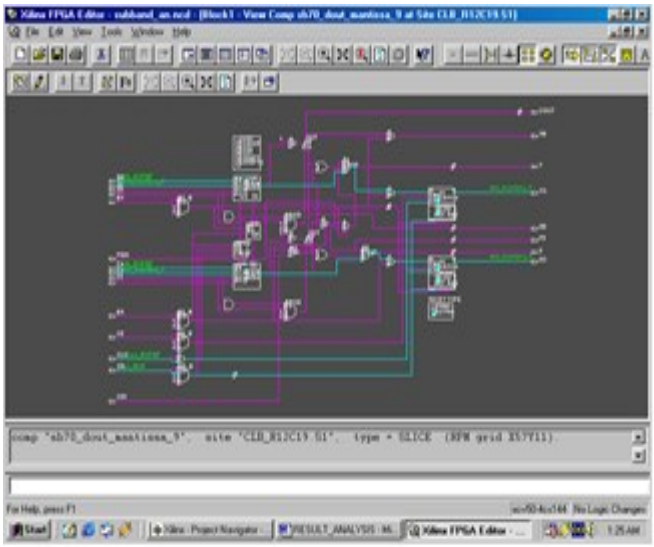


Figure 8: Logical Placement

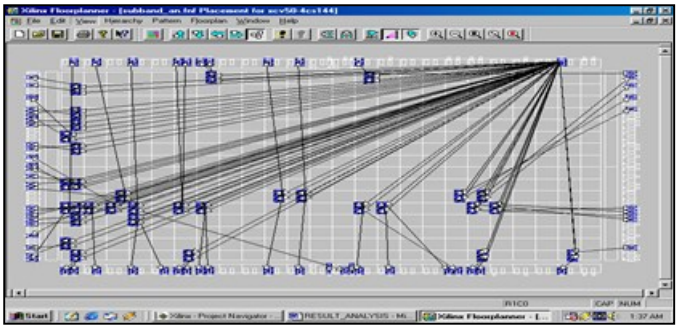


Figure 9: Floor Planning

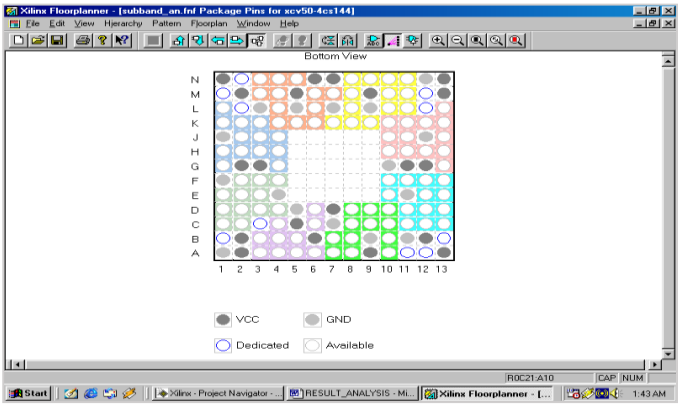


Figure 10: Bottom Package View

These are the steps to move towards the tape out the chip. In this sequence the next step is the floorplan. The Floorplan output is shown in Fig 9. It is actually indicates the logical blocks and interconnecting nets are so arranged to form connections between configurable logic block and the IO buffer. This interconnection indicates the logical mapping for the path of the resources of data flow operation.

The Package view is shown as in Fig 10.This is the bottommost view of the targeted FPGA device. It contains the power supply pin (VCC), the ground (GND) terminal. The dedicated lines used for data flow operation and the remaining pins for rest of use.

DISCUSSION

The fabrication yield and in-field stability are two important entities of any product. In this scenario the built-in self-repair (BISR) is assumed to be better solution. It helps is improving the total fabrication yield of the chip. Therefore to test the memories against the faults and introducing the self-repair concepts with redundant analysis algorithms have been proposed by the researchers [4],[3],[6]. The Self-Repair Redundancy structured scheme is proposed [3]. The proposed BISR circuit methodology has a low area overhead penalty is about 2% for an 8K × 64 static random-access memory (SRAM). A wide search of built in redundant analysis BIRA scheme for embedded dynamic RAM is proposed [4] that achieves ideal repairable rates for embedded memories. On the other hand, the hardware overhead to implement this algorithm is slightly larger. The branch and bound technique (BBT) with primary screening in the repair process is proposed [12]. The minimum number of necessary/required spares then may be modeled by a graph called a bipartite graph. The BISR techniques for on chip memories, containing casual defects and cluster faults respectively are proposed in [3] and [8][9]. It is always better to use the block based replacement methods in replacement of the old spare row (SR)/(SC) spare column redundancy approaches. Furthermore, the proposed redundancy mechanisms can be a better solution for the use it in internally or globally. Though, for system on chips which are containing multiple assorted memory cores, BISR approaches become more hard and complicated for implementation. Thus, it becomes mandatory to seek for effective procedures for repairing multiple memory cores. The configurable BISR scheme for repairing multiple RAMs is proposed [11][5][10]. However, the traditional redundant mechanisms (SRs/SCs) are used. In this research the attempt has been taken to limit these issues at certain extend.

CONCLUSION

Dynamic built in self repair scheme for static RAM with selectable redundancy has been presented in this paper. It is designed flexible that users can select operation modes of SRAM dynamically. More number of faults are detected using March –SS algorithm. The BIRA module can avoid storing fault addresses more than one and can repair fault address and bits more quickly. The Dynamic redundancy architecture repairs more number of faults by replacing even single bit fault with single redundancy bit. The test has conducted and smooth results obtained for all three cases of bit, block & word selections.

REFERENCES

[1] Ahmed MA, Rani D Elizabeth and Sattar Syed Abdul, (2015), FPGA Based High Speed Memory Bist Controller for Embedded Applications, Indian Journal of Science and Technology, Vol 8(33), pp 1-8.

[2] Aljumah A and Ahmed MA,(2016), AMBA Based Advanced DMA Controller for SoC. International Journal of Advanced Computer Science and Applications.Vol.7, 3,p-188. 2016.

[3] Chang, Da-Ming; Li, Jin-Fu; Huang, Yu-Jen,(2008), A Built-In Redundancy-Analysis Scheme for Random Access Memories with Two-Level Redundancy" Journal of Electronic Testing, 06/2008, Volume 24, Issue 1, pp-181–192. 2008.

[4] Huamin Cao, Ming Liu, Hong Chen, Xiang Zheng, Cong Wang and Zhihua Wang, (2012),Efficient Built-in Self-Repair Strategy for Embedded SRAM with Selectable Redundancy. IEEE, pp-2565-68,2012.

[5] Huang Chao-Da, Tseng Tsu-Wei, Li Jin-Fu,(2007),An infrastructure IP for repairing RAMs In SOCs. IEEE Trans. Very Large Scale Integr.Syst., vol. 15, no. 10, pp. 1135–1143,2007.

[6] Huang C T, Wu C F, Li J F, Wu C W,(2003), Built-in redundancy analysis for memory yield improvement. IEEE Trans Reliability 52(4): Dec. pp 386–399, 2003.

[7] ITRS.,(2015), International Technology Roadmap for Semiconductors. Available at:http://www.semiconductors.org/main/2015_international_ technology_roadmap_for_semiconductors_itrs/. Pp-1-64,2015.

[8] Lu S. K., H su C. H., Tsai Y. C., Wang K. H., and Wu C. W.,(2006), Efficient built-in Redundancy analysis for embedded memories with 2-D redundancy," IEEE Trans. Very Large Scale Integr. Syst., vol. 14, no.1, pp. 31–42,2006.

[9] Lu S. K., Yang C. L., Hsiao Y. C., and Wu C. W.,(2010), Efficient BISR techniques for embedded memories considering cluster faults. IEEE Trans. Very Large Scale Integr. Syst., vol. 18, no. 2, pp. 184–193,2010.

[10] Tseng T. W., Li J. F., and Hsu C. C.,(2010),A reconfigurable built-in self-repair Scheme for random access memories in SOCs," IEEE Trans. Very Large Scale Integr. Syst., vol.18, no. 6, pp. 921–932,2010.

[11] Tseng T. W., Li J. F., Hsu C. C., Pao A., Chiu K., and Chen E.,(2006), A reconfigurable built-in Self-repair scheme for multiple repairable RAMs in SOCs. in Proceeding ITC, pp.1–8, 2006.

[12] Wey C. L.& Lombardi F,(1987),On the repair of redundant RAM's, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. CAD-6, no. 2,pp. 222–231, 1987.

[13] Zorian Y & Shoukourian S,(2003), Embedded-memory test and repair: Infrastructure IP for SOC Yield, IEEE Des. Test Computer. vol. 20, no. 3,pp. 58–66. 2003.

[14] Aljumah A and Ahmed MA,(2015), . Design of High Speed Data Transfer Direct Memory Access Controller for System on Chip Based Embedded Products. *Journal of Applied Sciences, 15: 576-581.*

[15] M. Altaf Ahmed, D. Elizabath Rani, S. A. Sattar, Embedded Memory Test Strategies and Repair, International Journal of Engineering (IJE), TRANSACTIONS C: Aspects Vol. 30, No. 6, (June 2017) 839-845