IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 7, Issue 3, Ver. I (May. - June. 2017), PP 55-63
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Robust Fault Tolerance in Content Addressable Memory Interface

K.Suresh Kumar[1], Y.Rajasree Rao[2], K.Manjunathachari[3]
[1]*Asst.Prof., ECE Dept., SSJEC, Hyderabad, suresh233661@gmail.com ,*
[2]*Prof &Dean, St.Peter's Engg College, Hyderabad,.*[3]*Prof.&HOD, ECE Dept., GITAM University, Hyderabad, India.*

Abstract: *With the rapid improvement in data exchange, large memory devices have come out in recent past. The operational controlling for such large memory has became a tedious task due to faster, distributed nature of memory units. In the process of memory accessing it is observed that data written or fetched are often encounter with fault location and faulty data are written or fetched from the addressed locations. In real time applications, this error cannot be tolerated as it leads to variation in the operational condition dependent on the memory data. Hence, It is required to have an optimal controlling fault tolerance in content addressable memory. In this paper, we present an approach of fault tolerance approach by controlling the fault addressing overhead, by introducing a new addressing approach using redundant control modeling of fault address unit. The presented approach achieves the objective of fault controlling over multiple fault location in different dimensions with redundant coding.*

Index terms: *Fault controlling, content addressable memory, recurrent address location*

I. 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. As part of the development of VLSI technology, System-on chip (SoC) is developed which is capable to integrate a 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). 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 failduring its expected life-time, it is called off-line testing. However, this assumption does not hold for modern day ICs,based on deep sub-micron technology, because they may develop failures even during operation within expected lifetime. To cater to this problem sometimes redundant circuitry are kept on-chip which replace the faulty parts. Toenable replacement of faulty circuitry, the ICs are tested before each time they startup. If a fault is found, a part ofthe circuit (having the fault) is replaced with a corresponding redundant circuit part (by re-adjusting connections).Testing a circuit every time before they startup, is called Built-In-Self-Test (BIST). Once BIST finds a fault, thereadjustment in connections to replace the faulty part with a fault free one is a design problem.BIST reduces the time to test a chip. Inaddition, the aggressive design rules make the memory arrays prone todefects [3]. Therefore, the overall SoC yield is dominated by the memory yield, and optimizing the memory yield plays acrucial role in the SoC environment. To improve the yield, memory arrays are usually equipped with spare elements, and external testers have been used to test the memory arrays and configure the spare elements. However,in the SoC environment, the overall test time is prohibitivelyincreased if the test response data from the memory arrays aresent to the external testers. On the other hand, the SoC environment,combined with shrinking technology, allows us more areafor on-chip test infrastructure at lower cost than before, whichmakes feasible a variety of built-in self-test (BIST) and built-inself-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 outlines the past approaches. The conventional model of BISR for fault tolerance is outlined in section III. The proposed redundant BISR is shown in section IV. The experimental results are illustrated in section v and aconclusionis outlined in section VI.

II. Content Addressable Memory

CAM-intensive systems today rely on increasingly complex and computationally intensive applications. Many CAM systems must meet extremely rigorous speed goals, since they operate on lengthy segments of real-world signals in real-time. Another key characteristic of a CAM system is its sample rate: the rate at which samples are consumed, processed, or produced. Combined with the complexity of the algorithms, the sample rate determines the required speed of the implementation technology. The sample rates required for CAM systems vary over many orders of magnitude, depending upon the types of signals being manipulated. In many cases, sample rates are quite high relative to the basic clock cycle time of the available hardware technology. An additional challenge in CAM design verification is the need for realistic test data. The verification of CAM application often requires complex, realistic test signals. The development of CAM system begins with the establishment of requirements for the system. These requirements may include power, size, weight, bandwidth, and signal quality. During the design development, the designer is mostly concerned with exploring approaches to solve the problems posed by the specifications at an abstract level. At the algorithm development level, the designer must be able to specify and experiment with both the control behavior and the processing behavior of the application. System architecture use to describe the overall selection and organization of the main hardware and software components in a system is then a critical task. In selecting a system architecture, the designer's inputs typically include the algorithms and other functionality to be implemented. The design of system architecture is often the most important step in the design process in terms of its influence on the product's performance, cost and design time. Signal processing software is perhaps the most important part of CAM system software. Many CAM systems are implemented using multitasking, which requires a real-time operating system to coordinate the execution of multiple tasks. CAM system involve the development of new hardware-particularly those systems targeted at cost-sensitive, high-volume applications. System integration has to take place throughout the design process. As the designs are refined, the integration is also refined. Starting system integration early in the design process and refining the integration as design proceeds paves the way for relatively straightforward final system integration. The current CAM systems, unlike the general-purpose memory, use a non-uniform addressing model in which the primary components of the memory system is the DRAM and dual tag less SRAMs are referenced through completely separate segments of the address space. The recent trend of programming CAMs in high-level languages. In many of today's high-performance CAMs this non-uniform model is being replaced by a uniform model a transparent organization like that of most general-purpose systems, in which all memory structures share the same address space as the DRAM system. In such a memory organization, one must replace the CAM's tagless SRAMs with something resembling a general-purpose cache. The dual-channel SRAM design provides a large amount of fast storage while supporting guaranteed single-cycle access to both operands of a typical instruction (e.g. the product terms of a multiply accumulate). The SRAMs are tagless because they are not transparent to the programmer: CAMs offer segmented memory spaces where different physical memory structures (SRAM0, SRAM1, DRAM, ROM, etc.) are explicitly indicated by corresponding sets of data addresses.

II. Fault Tolerant In Memory

In order to improve fabrication yield and in-field reliability,built-in self-repair (BISR) is considered a promising solution.There are many BISR/built-in redundancy analysis (BIRA) algorithms and architectures proposed for test and repair ofembedded memories in the past [3]–[12], [18]–[24]. In [3], aBISR scheme with 2-D redundancy structure is proposed. Theproposed BISR circuit has a low area overhead about 4.6% for an 8K × 64 static random-access memory (SRAM). In [4], acomprehensive exhaustive search BIRA scheme for embeddeddynamic random-access memories is proposed which can achieve optimal repair rates. However, the hardware overhead (HO) required to implement this algorithm is very high. In [5], Wey and Lombardi presented a branch-and-bound techniquewith early screening in the repair process. The least number of required spares can be modeled by a bipartite graph.In [6] and [7], BISR techniques for embedded memories containing random defects and cluster faults are proposed, respectively. Instead of the traditional spare row (SR)/sparecolumn (SC) redundancy mechanisms, block-based replacement techniques are used. Moreover, the proposed redundancymechanisms can be used locally or globally. However, for systemchips containing multiple heterogeneous memory cores,BISR solutions become more complicate and difficult forimplementation. It is inevitable to seek for efficient methodologiesfor repairing multiple memory cores. In [9]–[11], areconfigurable BISR scheme for repairing multiple RAMsis proposed. However, the traditional redundant mechanisms (SRs/SCs) are used.

III. Proposed Address Governing In Cam

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 1. 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.

Table 1. SRAM Operation Modes

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

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. Figure 1 shows the flows of storing fault addresses. 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-Memoryas background. In this case, the fault address can be compared with all the data stored in Fault-address-Memoryno matter how many fault addresses have been stored. At last, the BISR strategy is high-speed. As shown in Figure 1, once a fault address is stored in Fault-address-Memory, it points to allfault address. 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.

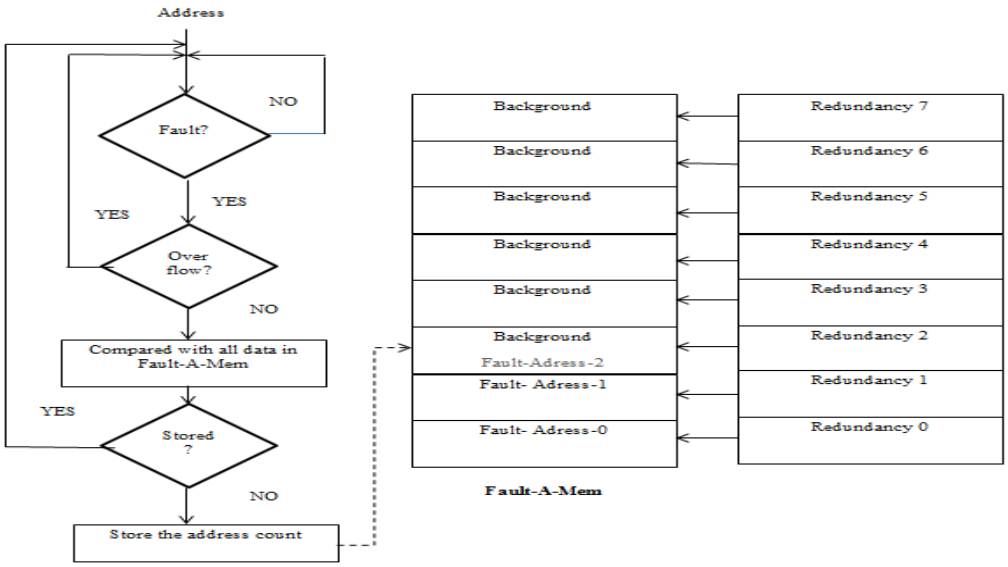


Figure.1: Flows of Storing Fault Addresses

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.

IV. 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 *dout* 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. 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 intothe 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 read operation and the data will be retrieved from spare in read mode. The proposed approach of memory fault testing is shown in figure 2.

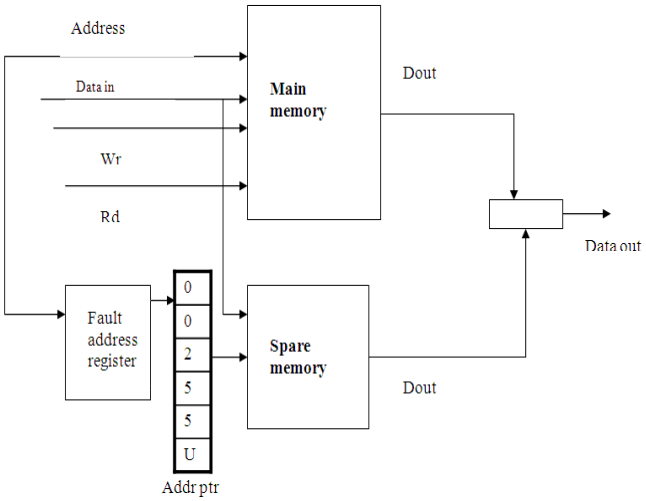


Figure 2: Proposed redundant address map for BISR

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 location are represented as shown in figure 3,

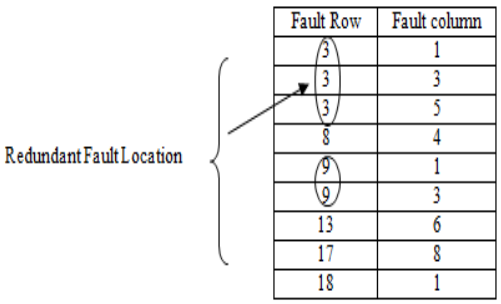


Figure 3: illustration of fault address memory in conventional BISR

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.2.

Table.2 Proposed redundant fault address memory

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

The operation of the suggested approach under data fetching with fault address logic is presented in figure 4.

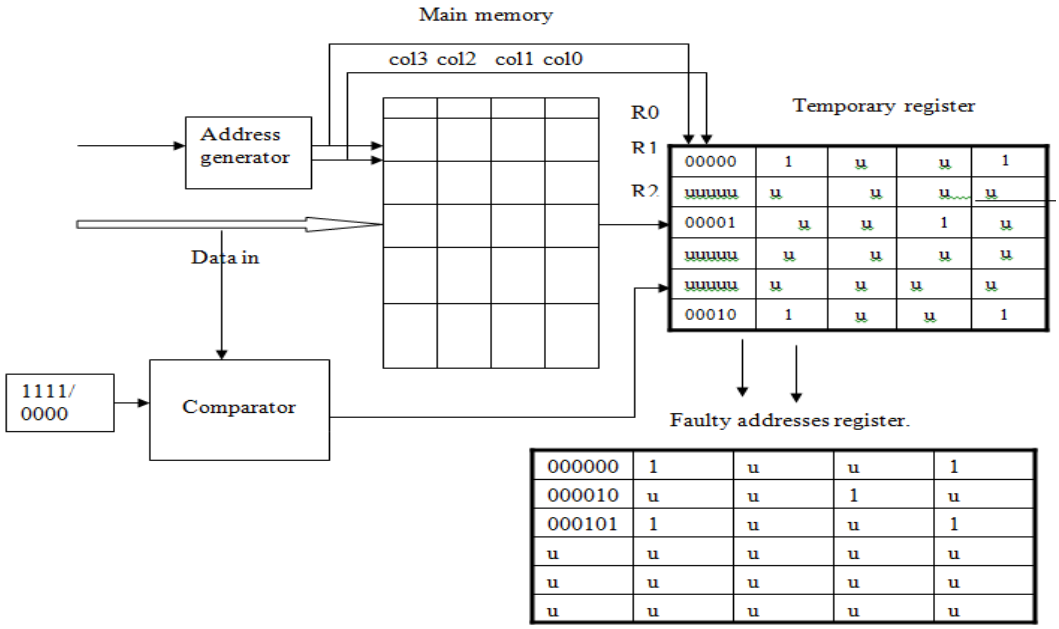


Figure 4: Operational flow for proposed redundant coding

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 figure 6 is performed.

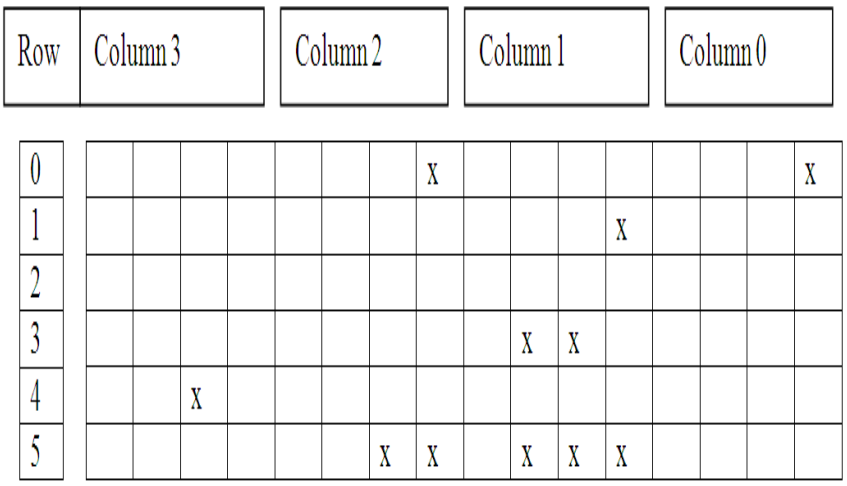


Figure 5: Memory unit with fault locations

In the operation the fault testing is carried out and the testing result is as shown below, No of spares are 3.

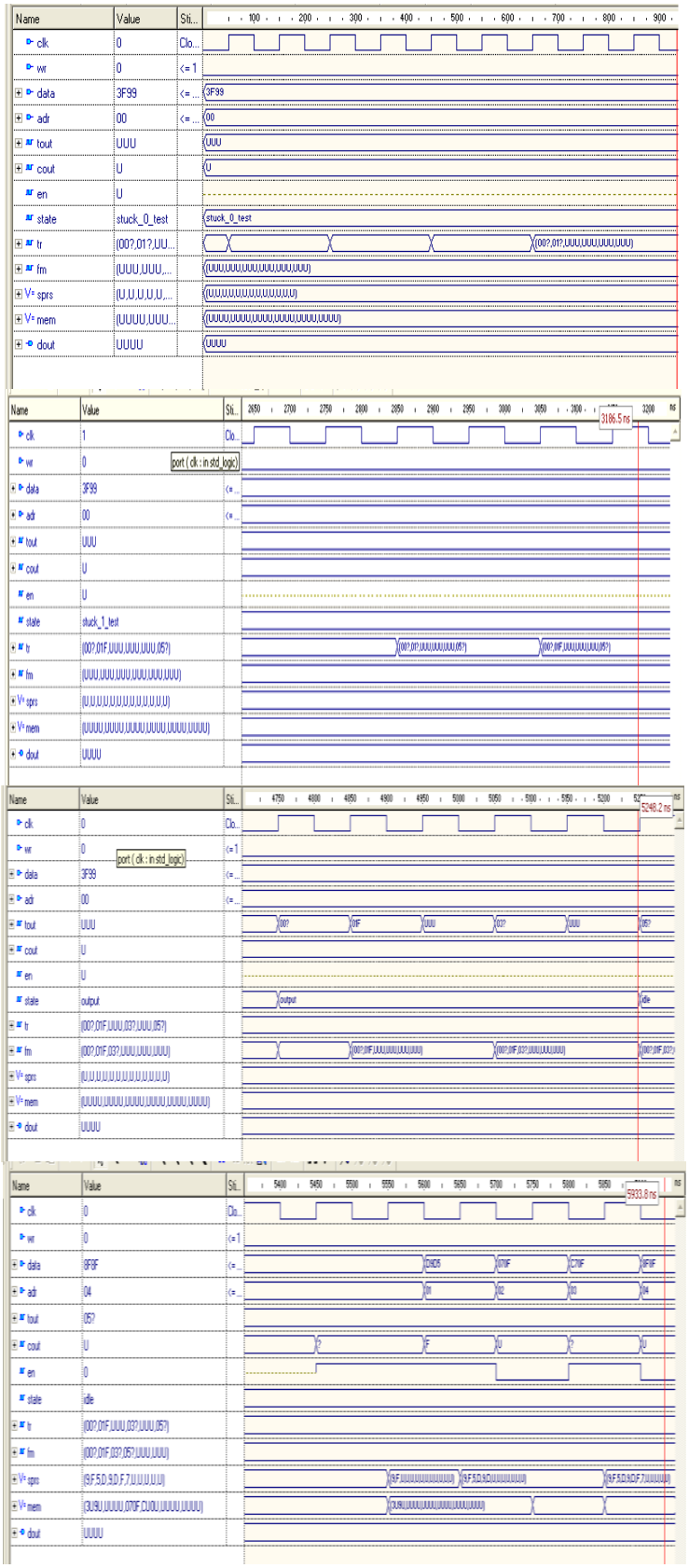
Table.1, observations on the selection over bit, block and word level

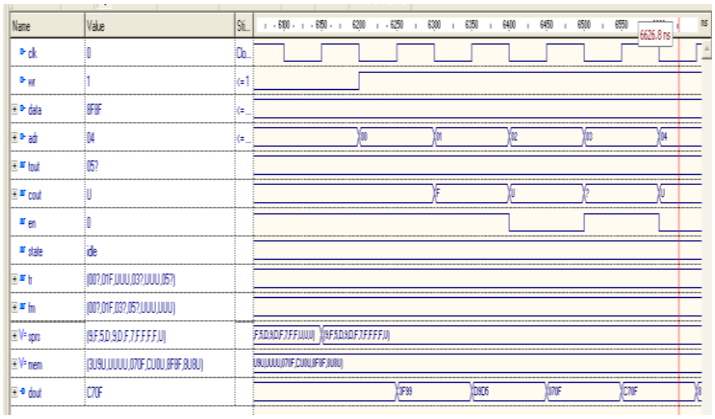
	Bit	**Block**	**Word**
0th row	2 cells	2 blocks	Full word
1th row	1 cell	1 block	Full word
2th row	----------No fault----------		
3throw	2 cells	1 block	Full word
4th row	1 cell	1 block	Not repeated
5th row	5 cells	2 blocks	Not repeated

V. Experimental Results

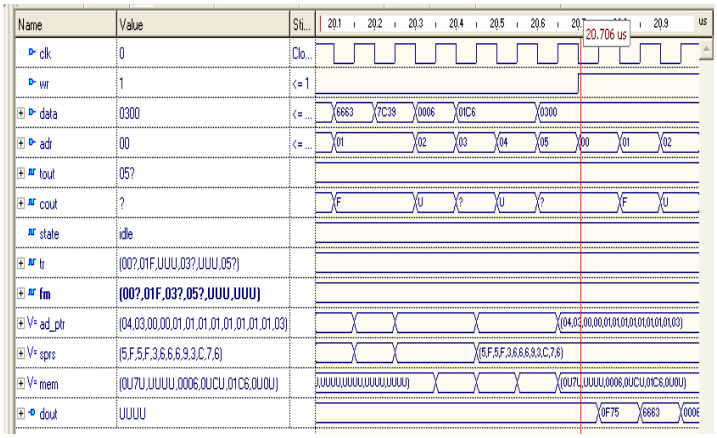
To evaluate the operational functionality and implementation feasibility, a simulation for the proposed approach is made. The obtained results are as illustrated below.

Case 1:Redundancy Added At in Block Level

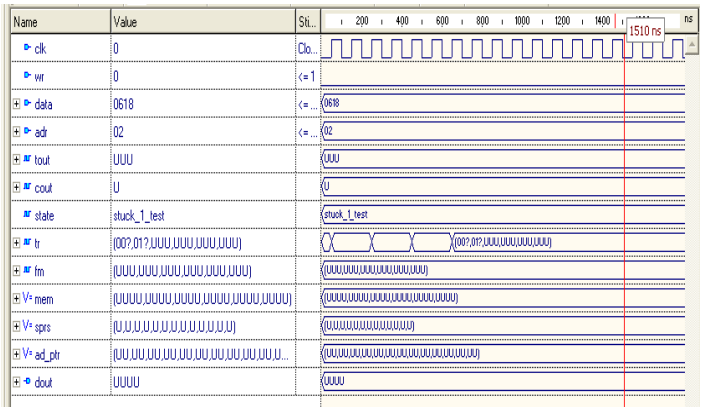




Case 2: Redundancy Added At in Bit Level



Case 3:Redundancy Added At in Word Level



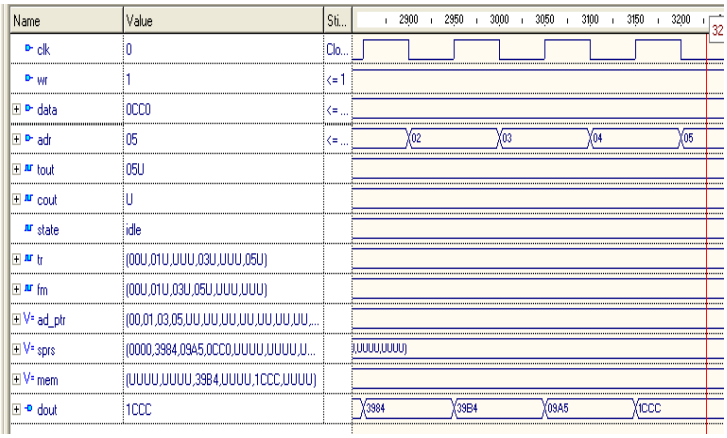


Table.3 Comparison of Observations are made for a 6/16 memory

	Bit	Block	Word
Utilization of memory (No. of cells)	80	68	16
No. of spares 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

As show in the table, we have faults in 0^{th},1^{st} ,3^{rd} ,4^{th} and 5^{th} 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 here the time taken to repair the memory is high and the hardware used to repair is more, this we can see in the table given below. 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 only a block is replaced, and if there are more than 2 cells faulty in a block, then that 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. A fault tolerant mechanism for memory operation is developed.The simulation observations over a memory location with stuck_at_0 & stuck_at_1 is 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.

VI. 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 input arrival time before clock: 9.167ns
Maximum output required time after clock: 6.229ns

VII. Conclusion

Dynamic BISR strategy for SRAM 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 once and can repair fault address and bits quickly. Dynamic redundancy architecture repair more number of faults by replacing even single bit fault with single redundancy bit.

References

[1]. *International Technology Roadmap for Semiconductors*. (2007) [Online].Available: http://www.itrs.net/links/2007itrs/home2007.html
[2]. Y. Zorian and S. Shoukourian, "Embedded-memory test and repair:Infrastructure IP for SOC yield," *IEEE Des. Test Comput.,* vol. 20, no. 3,pp. 58–66, May–Jun. 2003.
[3]. J. F. Li, J. C. Yeh, R. F. Huang, and C. W. Wu, "A built-in self-repairscheme for semiconductor memories with 2-D redundancy," in *Proc. Int.Test Conf.,* Oct. 2003, pp. 393–402.
[4]. T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada, and H. Hidaka,"A built-in self-repair analyzer (CRESTA) for embedded DRAMs," in*Proc. Int. Test Conf.,* Oct. 2000, pp. 567–574.
[5]. C. L. Wey and F. Lombardi, "On the repair of redundant RAM's," *IEEETrans. Comput.-Aided Des.Integr. Circuits Syst.,* vol. CAD-6, no. 2,pp. 222–231, Mar. 1987.
[6]. S. K. Lu, C. H. Hsu, Y. C. Tsai, K. H. Wang, and C. W. Wu,"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, Jan. 2006.
[7]. S. K. Lu, C. L. Yang, Y. C. Hsiao, and C. W. Wu, "Efficient BISRtechniques for embedded memories considering cluster faults," *IEEETrans. Very Large Scale Integr.Syst.,* vol. 18, no. 2, pp. 184–193, Feb.2010.
[8]. I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L.Lewandowski, "Built-in self-repair for embedded high density SRAM,"in*Proc. Int. Test Conf.,* 1998, pp. 1112–1119.
[9]. T. W. Tseng, J. F. Li, C. C. Hsu, A. Pao, K. Chiu, and E. Chen, "Areconfigurable built-in self-repair scheme for multiple repairable RAMsin SOCs," in *Proc. ITC,* Oct. 2006, pp. 1–8.
[10]. C. D. Huang, J. F. Li, and T. W. Tseng, "ProTaR: An infrastructure IPfor repairing RAMs in SOCs," *IEEE Trans. Very Large Scale Integr.Syst.,* vol. 15, no. 10, pp. 1135–1143, Oct. 2007.
[11]. T. W. Tseng, J. F. Li, and C. C. Hsu, "ReBISR: A reconfigurable built-inself-repair scheme for random access memories in SOCs," *IEEE Trans.Very Large Scale Integr. Syst.,* vol. 18, no. 6, pp. 921–932, Jun. 2010.
[12]. M. Miyazaki, T. Yoneda, and H. Fujiwara, "A memory grouping methodfor sharing memory BIST logic," in *Proc. Asian South Pacific Conf. Des.Auto.,* 2006, pp. 24–27.
[13]. M. Yoshimoto, K. Anami, H. Shinohara, T. Yoshihara, H. Takagi,S. Nagao, S. Kayano, and T. Nakano, "A divided word-line structurein the static RAM and its application to a 64K full CMOS RAM,"*IEEE J. Solid-State Circuits,* vol. SC-18, no. 5, pp. 479–485, Oct. 1983.
[14]. A. Karandidar and K. K. Parhi, "Low power SRAM design usinghierarchical divided bit-line approach," in *Proc. Int. Conf. Comput.Des.,*Oct. 1998, pp. 82–88.
[15]. C. T. Huang, C. F. Wu, J. F. Li, and C. W. Wu, "Built-in redundancyanalysis for memory yield improvement," *IEEE Trans. Reliab.,* vol. 52,no. 4, pp. 386–399, Dec. 2003.
[16]. R. F. Huang, J. F. Li, J. C. Yeh, and C. W. Wu, "A simulatorfor evaluating redundancy analysis algorithms of repairable embeddedmemories," in *Proc. IEEE Int. Workshop MTDT,* Jul. 2002, pp. 68–73.[17] C. H. Stapper, "Simulation of spatial fault distributions for integratedcircuit yield estimations," *IEEE Trans. Comput.-Aided Des.,* vol. 8,no. 12, pp. 1314–1318, Dec. 1989.
[17]. D. K. Bhavsar, "An algorithm for row-column self-repair of RAM's andits implementation in the Alpha 21264," in *Proc. Int. Test Conf.,* Sep.1999, pp. 311–318.
[18]. W. K. Huang, Y. H. Shen, and F. Lombardi, "New approaches for therepairs of memories with redundancy by row/column deletion for yieldenhancement," *IEEE Trans. Comput.-Aided Des.Integr. Circuits Syst.,*vol. 9, no. 3, pp. 323–328, Mar. 1990.
[19]. P. Mazumder and Y. S. Jih, "A new built-in self-repair approach to VLSImemory yield enhancement by using neural-type circuits," *IEEE Trans.Comput.-Aided Des.Integr.Circuits Syst.,* vol. 12, no. 1, pp. 24–36, Jan.1993.
[20]. S. Y. Kuo and W. K. Fuchs, "Efficient spare allocation in reconfigurablearrays," *IEEE Des. Test Comput.,* vol. 4, no. 1, pp. 24–31, Feb. 1987.
[21]. M. Nicolaidis, N. Achouri, and S. Boutobza, "Dynamic data-bit memorybuilt-in self-repair," in *Proc. Int. Conf. Comput.-Aided Des.,* Nov. 2003,pp. 588–594.
[22]. M. Nicolaidis, N. Achouri, and L. Anghel, "Memory built-in self-repairfor nanotechnologies," in *Proc. Int. On-Line Test. Symp.,* Jul. 2003, pp.94–98.
[23]. L. M. Deng, T. C. Wang, and C. W. Wu, "An enhanced SRAM BISRdesign with reduced timing penalty," in *Proc. 15th ATS,* Nov. 2006,pp. 25–30.