*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 1, Ver. I (Jan - Feb 2019), PP 08-15*
*www.iosrjournals.org*

# Built-In Self-Repair Techniques of Embedded Memories with BIST for Improving Reliability

P.S.N. Bhaskar[1], B. Sarada [2], Kandregula Shailaja[3].
*M.Tech Assistant Professor Sanketika Vidya Parishad College of Engineering, Visakhapatnam*
*M.Tech Assistant Professor Sanketika Vidya Parishad College of Engineering, Visakhapatnam*
*M.Tech Student Sanketika Vidya Parishad College of Engineering, Visakhapatnam*
*Corresponding Author: P.S.N. Bhaskar*

***Abstract:*** *Error correction code (ECC) and built-in selfrepair(BISR) techniques by using redundancies have beenwidely used for improving the yield and reliability of embeddedmemories. The target faults of these two schemes are soft errorsand permanent (hard) faults, respectively. In recent works, thereare also some techniques integrating ECC and BISR to deal withsoft errors and hard defects simultaneously. However, this willcompromise reliability, since some of the ECC protection capabilityis used for repairing single hard faults. To cure this dilemma,we propose an ECC-enhanced BISR (EBISR) technique, whichuses ECC to repair single permanent faults first and sparesfor the remaining faults in the production/power-ON test andrepair stage. However, techniques are proposed to maintain theoriginal reliability during the online test and repair stage. We alsopropose the corresponding hardware architecture for the EBISRscheme. A simulator is implemented to evaluate the hardwareoverhead (HO), repair rate, reliability, and performance penalty.Experimental results show that the proposed EBISR scheme canimprove yield and reliability significantly with negligible HO andperformance penalty.*

***Index Terms—****Built-in self-repair (BISR), error correctioncode (ECC), hard repair, reliability, yield.*

---

Date of Submission: 26-12-2018 Date of acceptance: 11-01-2019

---

## I. Introduction

Nowadays, the area occupied by embedded memories in System-on-Chip (SoC) is over 90%, and expected to rise up to 94% by 2014. Thus, the performance and yield of embedded memories will dominate that of SoCs. However, memory fabrication yield is limited largely by random defects, random oxide pinholes, random leakage defects, gross processing and assembly faults, specific processing faults, misalignments, gross photo defects and other faults and defects. To increase the reliability and yield of embedded memories, many redundancy mechanisms have been proposed [3-6]. In both redundant rows and columns are incorporated into the memory array. In spare words, rows, and columns are added into the word-oriented memory cores as redundancy. All these redundancy mechanisms bring penalty of area and complexity to embedded memories design. Considered that compiler is used to configure SRAM for different needs, the BISR had better bring no change to other modules in SRAM. To solve the problem, a new redundancy scheme is proposed in this paper. Some normal words in embedded memories can be selected as redundancy instead of adding spare words, spare rows, spare columns or spare blocks.

Memory test is necessary before using redundancy to repair. Design for test (DFT) techniques proposed in 1970 improves the testability by including additional circuitry. The DFT circuitry controlled through a BIST circuitry is more time-saving and efficient compared to that controlled by the external tester (ATE). However, memory BIST does not address the loss of parts due to manufacturing defects but only the screening aspects of the manufactured parts. BISR techniques aim at testing embedded memories, saving the fault addresses and replacing them with redundancy. In, the authors proposed anew memory BISR strategy applying two serial redundancy analysis (RA) stages. Presents an efficient repair algorithm for embedded memory with multiple redundancies and a BISR circuit using the proposed algorithm. All the previous BISR techniques can repair memories, but they didn't tell us how to avoid storing fault address more than once.

## II. Related Work

### A. Built In Self Test (BIST)

Built-In Self Test (BIST) constitutes an attractive and practical solution to the problem of testing VLSI devices and systems. Advantages of BIST include the capability of performing at-speed testing, very high fault coverage, elimination of test generation effort and less reliance on expensive external testing equipment for

applying and monitoring test patterns. Therefore BIST drives down the cost of testing. BIST techniques are classified into off-line and on-line.

An off-line BIST architecture operates in one out of two modes, normal and test. During normal mode the BIST circuitry is idle. During test mode the inputs generated by the Test Generator (TG) are applied to the inputs of the Circuit Under Test (CUT) and the responses are captured in the Response Verifier (RV). Upon test completion, the contents of the response verifier are examined and the CUT status is decided. Off-line testing schemes cannot detect temporary faults, since the detection of such faults requires continuous monitoring of the CUT outputs. In off-line BIST, the normal operation of the CUT is stalled in order to perform the test.

Thus, if the CUT is of critical importance for the function of the circuit, the total circuit performance is degraded. To avoid this performance degradation, input vector monitoringconcurrent BIST techniques have been proposed, that exploit input vectors arriving at the inputs of the CUT during normal operation.

**B. Logic Built In Self Test (LBIST)**

The traditional LBIST typically employs a Linear FSR (LFSR) to generate pseudo-random test patterns that are applied to the circuit under test and a Multiple Input Signature Register (MISR) for obtaining the compacted response of the circuit to these test patterns. An incorrect MISR output indicates a fault in the circuit. Various techniques can be used to complement pseudo-random test patterns.

A problem with the traditional LBIST is that many pseudorandom patterns (several thousands or more) need to be applied to reach a satisfactory fault coverage. This implies that test execution time can be too long for some applications.

The conventional way of test is to put the IC in an ATE, to generate test stimuli and check the IC responses. In this case all PINS are used, therefore this procedure is not applicable when the IC is soldered in final product board.

An alternative way to support testing IC when it is already soldered on final product board is to reduce the number of needed pins to perform test. It could be done by inserting inside the IC some extra-logic in charge of testing the circuit itself. This approach, illustrated in Figure uses the main concept of Logic BIST (LBIST) which consists of a self test mechanism for digital circuits.

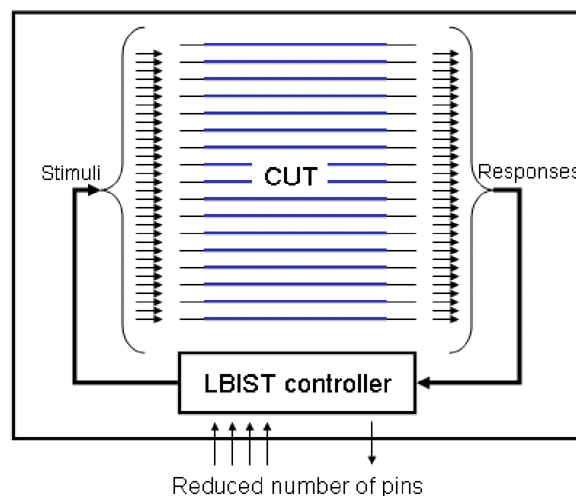


**Fig1. General LBIST scheme**

The LBIST architecture most commonly used is the STUMPS architecture, see Figure which uses a pseudo-random pattern generator/linear feedback shift register (PRPG/LFSR) to generate the inputs to the device's internal scan chain, initiate a functional cycle to capture the response of the device, and then compress the captured response into a MISR. The output of the MISR is called test signature and any corruption in output test signature indicates structural defect in the device.

## III. Proposed Model

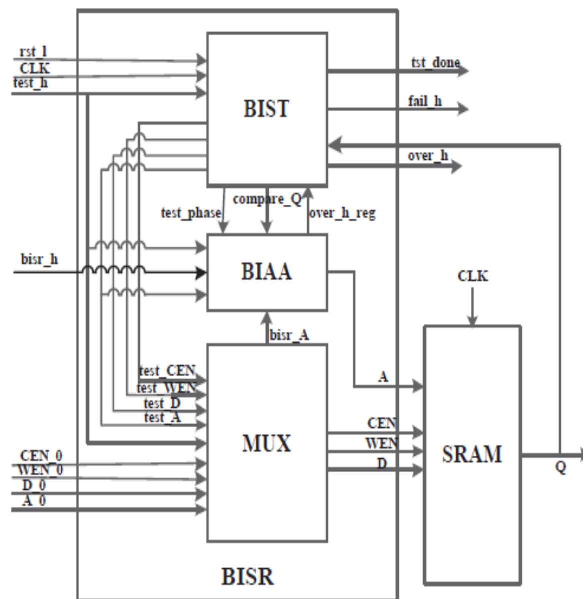


**Fig2. BISR Architecture**

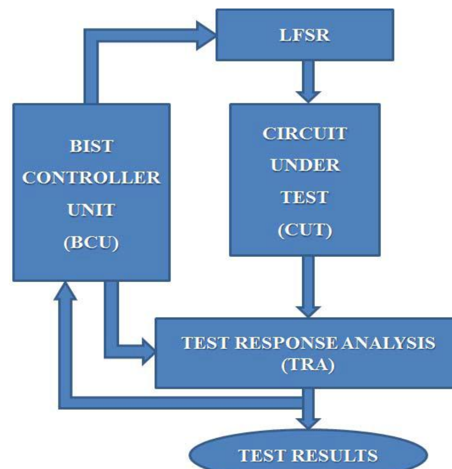


**Fig3. Proposed Model with CUT**

### a) Test Pattern Generator (TPG)

This module generates the test patterns required to sensitize the faults and propagate the effect to the outputs (of the CUT). As the test pattern generator is a circuit (not equipment) its area is limited. So storing and then generating test patterns obtained by ATPG algorithms on the CUT using the hardware test pattern generator is not feasible. In other words, the test pattern generator cannot be a memory where all test patters obtained by running ATPG algorithms (or random pattern generation algorithms) on the CUT are stored and applied during execution of the BIST. Instead, the test pattern generator is basically a type of register which generates random patterns which act as test patterns. The main emphasis of the register design is to have low area yet generate as many different patterns (from 0 to 2n, if there are $n$ flip-flops in the register) as possible.

### b) LFSR (Linear Feedback Shift Register)

The proposed low power LFSR technique uses bit swapping technique to reduce the peak power. By connecting multiplexers on the LFSR register. The number of transitions are decreased for that cell which are under bit swapping. The number of transitions in each register in LFSR without applying bit swapping technique here two cells in an $n$bit LFSR are considered to be adjacent if the output of one cell feeds the input of the second directly (i.e., without an intervening XOR gate). Each cell in a maximal-length $n$-stage LFSR (internal or external) will produce a number of transitions equal to $2n$-1 after going through a sequence of $2n$ clock cycles.

The sequence of 1s and 0s that is followed by one bit position of a maximal-length LFSR is commonly referred to as an *m* sequence. Each bit within the LFSR will follow the same *m* sequence with a one-time-step delay. The *m*-sequence generated by an LFSR of length *n* has a periodicity of $2n$-1. It is a well-known standard property of an *m*-sequence of length *n* that the total number of runs of consecutive occurrences of the same binary digit is $2n$-1.The beginning of each run is marked by a transition between 0 and 1.Therefore, the total number of transitions for each stage of the LFSR is $2n$-1.

**c) Test Response Analysis (TRA)**

It analyses the value sequence on PO and compares it with the expected output.

**d) BIST Controller Unit (BCU)**

It controls the test execution; it manages the TPG, TRA and reconfigures the CUT and the multiplexer.

Basic BIST architecture includes functions which are necessary to execute the self-testing feature so that testing is accomplished without the aid of external hardware. It has two major components named Hardware or Test Pattern Generator (TPG) and Output Response Compacter or Analyzer (ORA).

The TPG produces a sequence of patterns for testing the Circuit Under Test (CUT) while the ORA compacts the output responses of the CUT into some type of pass/fail indication which decides good or faulty result. The other two functions needed for system-level use of BIST include the test controller and input MUX. Besides the normal input/output (I/O) pins, the incorporation of BIST may also require additional I/O pins for activating the BIST sequence and to give valid results. The input test patterns can be stored in a Read Only Memory (ROM). Expected responses are read from ORA ROM and are compared to the actual output response of the CUT for each test vector. Any mismatch detected by the comparator is latched to indicate a failure has occurred during the BIST sequencing.

**e) Circuit Under Test (CUT)**

It is the portion of the circuit tested in BIST mode. It can be sequential, combinational or a memory. It is delimited by their Primary Input (PI) and Primary Output (PO).

In CUT we are using UART(Universal Asynchronous receiver/Transmitter). Serial data is transmitted via its serial port. A serial port is one of the most universal parts of a computer. It is a connector where serial line is attached and connected to peripheral devices such as mouse, modem, printer and even to another computer. In contrast to parallel communication, these peripheral devices communicate using a serial bit stream.

Design with BIST Capability protocol (where data is sent one bit at a time). The serial port is usually connected to UART, an integrated circuit which handles the conversion between serial and parallel data.

There are many reasons to use an SRAM or a DRAM in a system design. Design tradeoffs include density, speed, volatility, cost, and features. All of these factors should be considered before you select a RAM for your system design.

- **Speed.** The primary advantage of an SRAM over a DRAM is its speed. The fastest DRAMs on the market still require five to ten processor clock cycles to access the first bit of data. Although features such as EDO and Fast Page Mode have improved the speed with which subsequent bits of data can be accessed, bus performance and other limitations mean the processor must wait for data coming from DRAM. Fast, synchronous SRAMs can operate at processor speeds of 250 MHz and beyond, with access and cycle times equal to the clock cycle used by the microprocessor. With a well designed cache using ultra-fast SRAMs, conditions in which the processor has to wait for a DRAM access become rare.
- **Density.** Because of the way DRAM and SRAM memory cells are designed, readily available DRAMs have significantly higher densities than the largest SRAMs. Thus, when 64 Mb DRAMs are rolling off the production lines, the largest SRAMs are expected to be only 16 Mb.
- **Volatility.** While SRAM memory cells require more space on the silicon chip, they have other advantages that translate directly into improved performance. Unlike DRAMs, SRAM cells do not need to be refreshed. This means they are available for reading and writing data 100% of the time.
- **Cost.** If cost is the primary factor in a memory design, then DRAMs win hands down. If, on the other hand, performance is a critical factor, then a well-designed SRAM is an effective cost performance solution.
- **Custom features.** Most DRAMs come in only one or two flavors. This keeps the cost down, but doesn't help when you need a particular kind of addressing sequence, or some other custom feature. IBM's SRAMs are tailored, via metal and substrate, for the processor or application that will be using them. Features are connected or disconnected according to the requirements of the user. Likewise, interface levels are selected to match the processor levels. IBM provides processor specific solutions by producing a chip with a standard core design, plus metal mask options to define feature sets.

**III.BIAA**

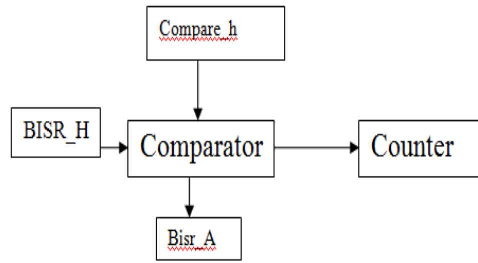


**Fig .4: BIAA Architecture**

If The BISR_H=0 then initially BIAA stores 0 address when the compare_h=0 then comparator can take address from the mux and directly it will given to the counter. If The BISR_H=1 then counter increments the values from 0 t0 8 why because in the bist comparator and test_v_ram having 0 t0 8 addresses only.

The proposed SRAM BISR strategy is flexible. The SRAM users can decide whether to use it by setting a signal. So the redundancy of the SRAM is designed to be selectable. In another word, some normal words in SRAM can be selected as redundancy if the SRAM needs to repair itself. We call these words Normal-Redundant words to distinguish them from the real normal ones. We take a 64 × 4 SRAM for example, as shown in Figure 1. There are 60 normal words and 4 Normal-Redundant words. When the BISR is used, the Normal-Redundant words are accessed as normal ones. Otherwise, the Normal-Redundant words can only be accessed when there are faults in normal words. In this case, the SRAM can only offer capacity of 60 words to users. This should be referred in SRAM manual in details.

**BISR Procedure and working**

Figure 3.3 shows the proposed BISR block diagram. The BISR starts by resetting the system (rst_l = 0). After that if the system work in test mode, it goes into TEST phase. During this phase, the BIST module and BIAA module work in parallel.

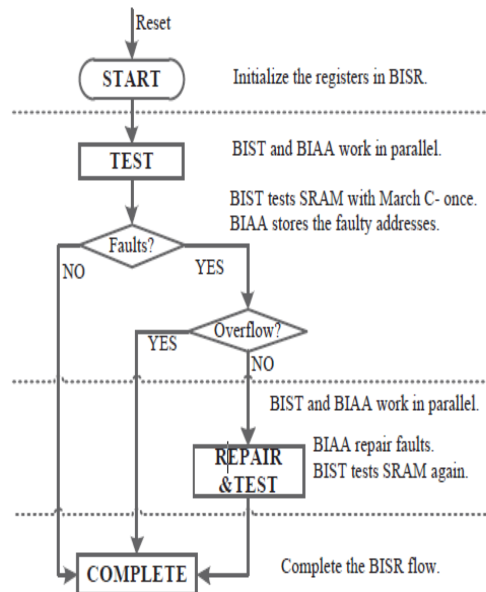


**Fig.5. Block Diagram of BISR**

The BIST use March C- to test the normal addresses of SRAM. As long as any fault is detected by the BIST module, the faulty address will be sent to the BIAA module. Then the BIAA module checks whether the faulty address has been already stored in Fault-A-Mem. If the faulty address has not been stored, the BIAA stores it and the faulty address counter adds 1. Otherwise, the faulty address can be ignored. When the test is completed, there will be two conditions. If there is no fault or there are too many faults that overflow the redundancy capacity, BISR goes into COMPLETE phase. If there are faults in SRAM but without overflows, the system goes into REPAIR&TEST phase. The same as during TEST phase, the BIST module and BIAA module work at the same time in REPAIR&TEST phase. The BIAA module replaces the faulty addresses stored

in Fault-A-Mem with redundant ones and the BIST module tests the SRAM again. There will be two results: repair fail or repair pass. By using the BISR, the users can pick out the SRAMs that can be repaired with redundancy or the ones with no fault.
we have analysis for RTL somatic diagrams for proposed methodology.

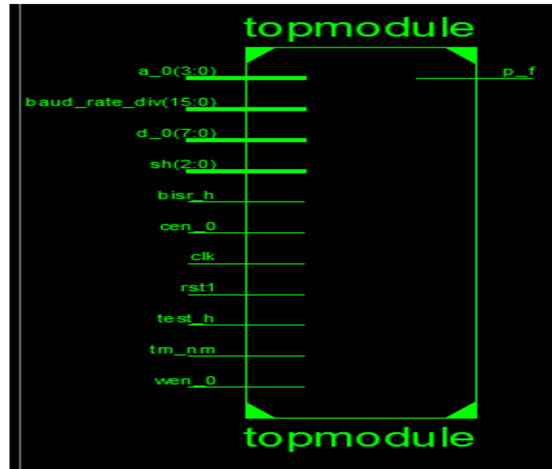


**Fig: 6. Pin Diagram of proposed system**

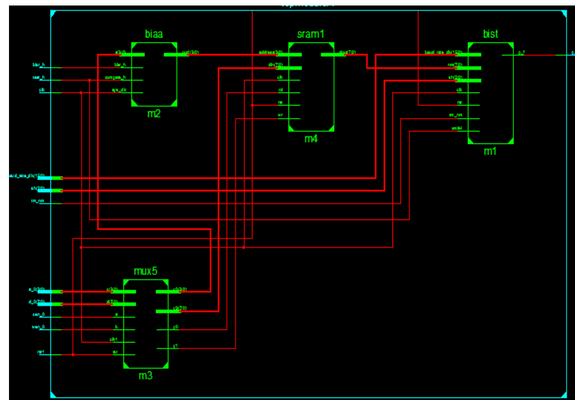


**Fig7. RTL diagram of Proposed System**

**BISR Features**

Firstly, the BISR strategy is flexible. TABLE I lists the operation modes of SRAM. 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.

**Comparison Table for Existing Methodology and Proposed Methodology**

| Component | Existing System | Proposed System |
|---|---|---|
| Area | 36% | 38% |
| Delay | 13.186ns | 10.516ns |
| Frequency | 200.562MHz | 173.040MHz |
| Memory | 214164 kilobytes | 213716 kilobytes |

**Table1: Comparison Table for Existing Methodology and Proposed Methodology**

The above table compared to existing methodology analysis proposed system is very efficient results will be produced for area, delay and memory also. the are will be minimized in the proposed analysis automatically reduce the power consumption for total circuit.

## IV. Simulatation Results

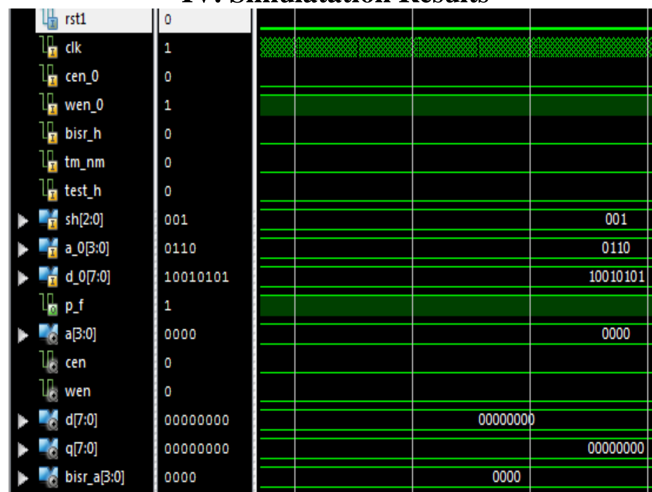


**Fig.8: Simulation Results for BISR when Rst=0**

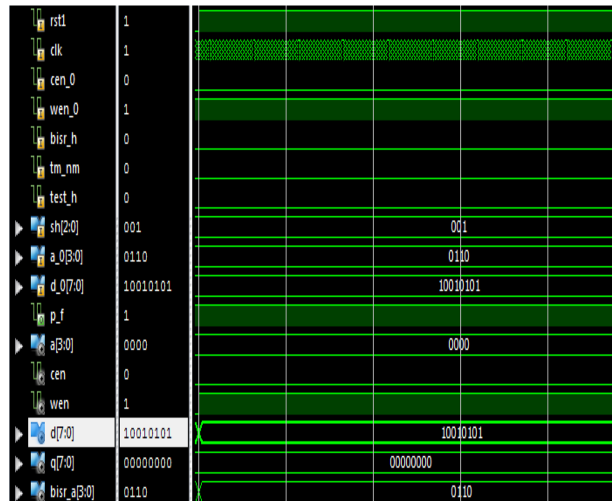


**Fig.9: Simulation Results for BISR when Rst=1**

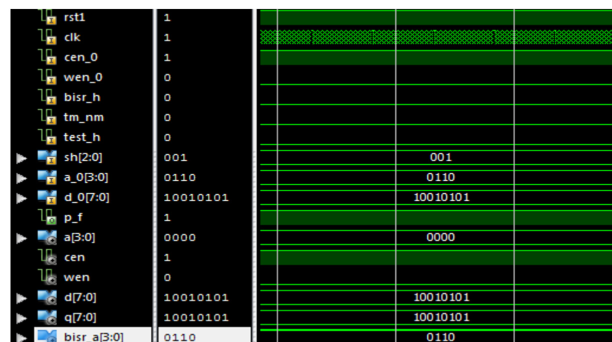


**Fig.10: Simulation Results for BISR when Rst=1(Rd=1&wr=0)**

## V. Conclusion

An efficient BISR strategy for SRAM IP with selectable redundancy has been presented in this paper. It is designed flexible that users can select operation modes of SRAM. The BIAA module can avoid storing fault addresses more than once and can repair fault address quickly. The function of BISR has been verified by the post simulation. The BISR can work at up to 150MHz at the expense of 20% greater area.

## References

[1]. Semiconductor Industry Association, “International technology roadmap for semiconductors (ITRS), 2003 edition,” Hsinchu, Taiwan, Dec.2003.

[2]. C. Stapper, A. Mclaren, and M. Dreckman, “Yield model for Productivity Optimization of VLSI Memory Chips with redundancy and Partially good Product,” IBM Journal of Research and Development, Vol. 24, No. 3, pp. 398-409, May 1980.

[3]. W. K. Huang, Y. H. shen, and F. lombrardi, “New approaches for repairs of memories with redundancy by row/column deletion for yield enhancement,” IEEE Transactions on Computer-Aided Design, vol. 9, No. 3, pp. 323-328, Mar. 1990.

[4]. P. Mazumder and Y. S. Jih, “A new built-in self-repair approach to VLSI memory yield enhancement by using neuraltype circuits,” IEEE transactions on Computer Aided Design, vol. 12, No. 1, Jan, 1993.

[5]. H. C. Kim, D. S. Yi, J. Y. Park, and C. H. Cho, “A BISR (built-in self repair) circuit for embedded memory with multiple redundancies,” VLSI and CAD 6th International Conference, pp. 602-605, Oct. 1999.

[6]. Shyue-Kung Lu, Chun-Lin Yang, and Han-Wen Lin, “Efficient BISR Techniques for Word-Oriented Embedded Memories with Hierarchical Redundancy,” IEEE ICIS-COMSAR, pp. 355-360, 2006.

[7]. C. Stroud, A Designer’s Guide to Built-In Self-Test, Kluwer Academic Publishers, 2002.

[8]. Karunaratne. M and Oomann. B, “Yield gain with memory BISR-a case study,” IEEE MWSCAS, pp. 699-702, 2009.

[9]. I. Kang, W. Jeong, and S. Kang, “ High-efficiency memory BISR with two serial RA stages using spare memories,” IET Electron. Lett., vol. 44, no. 8, pp. 515-517, Apr. 2008.

[10]. Heon-cheol Kim, Dong-soon Yi, Jin-young Park, and Chang-hyun Cho, “A BISR (Built-In Self-Repair) circuit for embedded memory with multiple redundancies,” in Proc. Int. Conf. VLSI CAD, Oct. 1999, pp. 602-605.

P.S.N. Bhaskar M. "Built-In Self-Repair Techniques of Embedded Memories with BIST for Improving Reliability." IOSR Journal of Computer Engineering (IOSR-JCE) 21.1 (2019): 08-15.