RESEARCH ARTICLE OPEN ACCESS

Design and Implementation of Improved BISR Strategy for Systems-on-a-Chip (SoC)

Mr. D. Sri Harsha[1] , Mr. D. Surendra Rao[2]
[1]Assistant Professor, Dept. of ECE, GNITC, Hyderabad
[2]Assistant Professor, Dept. of ECE, GNITC, Hyderabad sriharsha2284@gmail.com[1], sdustakar@gmail.com[2]

Abstract:
This paper focuses on improved strategy which is used to solve memory failure problems in systems-on-a-chip (SOCs). Built-In Self-Repair (BISR) with Redundancy is proposed. It is an effective yield-enhancement strategy for embedded memories. We designed and implemented an efficient BISR strategy for embedded memories. The BISR design is flexible that it can provide four operation modes to SRAM users. Each fault address can be saved only once in the feature of the proposed BISR strategy, and hence high speed. So the redundancy of the SRAM is designed to be selectable. In another words, some normal words in SRAM can be selected as redundancy if, the SRAM needs to repair itself such that it detects SRAM failures with a comparator that compares actual memory data with expected data.

Index terms: SoCs, SRAM, ATE, CMOS, Built-In Self-Repair (BISR), Built-In Self-Test (BIST), Built-In Address-Analysis (BIAA),

I. INTRODUCTION

Embedded memories plays vital role and consuming an increasing portion of the die area in deep submicron systems-on-a-chip (SOCs) [2, 3]. Manufacturing test of embedded memories is an essential step in the SOC production that screens out the defective chips and accelerates the transition from the yield learning phase to the volume production phase of a new manufacturing technology. Built-in self-test (BIST) is establishing itself as an enabling technology that can effectively tackle the SOC test problem.

Many redundancy mechanismshave been proposed to increase the reliability and yield of embedded memories. Both redundant rows and columns are incorporated into the memory array. The 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 [1, 5, and 6]. 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.

This paper proposes embedded memory BISR strategy for Systems-on-a-chip. First we will discuss theory on memory testing, by comparing ATE verses BIST we come to know that importance of BIST. Then we concentrate to understand the importance of fault tolerant design of digital systems and various redundancy approaches [3, 4, and 5]. By understanding functional testing and MARCH test algorithms, we propose BISR strategy for embedded memory testing. Simulation wave forms will conclude proposed BISR strategy is well suited for embedded memories for System on Chips.

II. TESTING APPROACHES

In very large scale integrated (VLSI) technology, an increasing number of transistors can be fabricated onto a single silicon die. For example, a state-of-the-art 130 nm complementary metal-oxide semiconductor (CMOS) process technology can have up to eight metal layers, poly gate lengths as small as 80 nm and silicon densities of 200K-300K gates/mm2. However, although million gates integrated circuits (ICs) can be manufactured, the increased chip complexity requires robust and sophisticated test methods [7]. Hence, manufacturing test is becoming an enabling technology that can improve the declining manufacturing yield, as well as control the production cost, which is on the rise due to the escalating volume of test data and testing times. Therefore reducing the cost of manufacturing test, while improving the test quality required to achieve higher product reliability and manufacturing yield, has already been established as a key task in VLSI design

A. Digital Test Methodologies ATE vs. BIST:

The basic principle of manufacturing testing is illustrated in Figure 1 Circuit under test (CUT) can be the entire chip or only a part of the chip (e.g., a memory core or a logic block). Input test vectors are binary patterns applied to the inputs of the CUT and the associated output responses are the values observed on the outputs of the CUT. Using a comparator output responses are checked against the expected correct response data, which is obtained through simulation prior to design tape-out. If all the output responses match the correct response data, the CUT has passed the test and it is labeled as fault-free. Based on the techniques how the test vectors are applied tothe CUT and how the output responses are compared, there are two main directions to test electronic circuits: external testing using automatic test equipment (ATE) and internal testing using built-in self-test (BIST). When external testing is employed, the input test vectors and correct response data are stored in the ATE memory. Input test vectors are generated using ATPG tools, while correct response data is obtained through circuit simulation. For external testing, the comparison is carried out on the tester. Although the ATE-based test methodology has been dominant in the past, as transistor to pin ratio and circuit operating frequencies continue to increase, there is a growing gap between the ATE capabilities and circuit test requirements (especially in terms of speed and volume of test data).

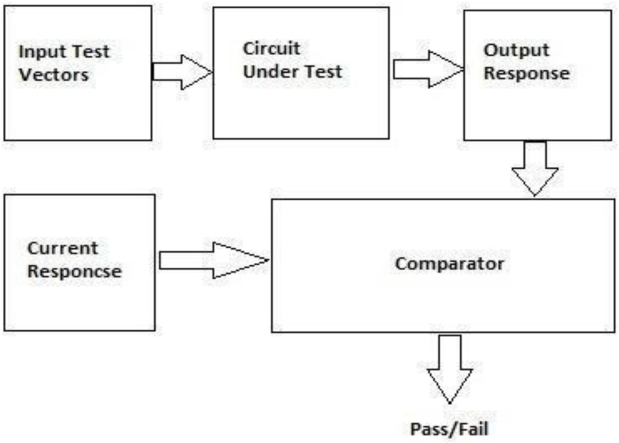

Fig.1: Basic Principle of Digital Testing

ATE limitations make BIST technology an attractive alternative to external test for complex chips. BIST is a design-for-test (DFT) method where part of the circuit is used to test the circuit itself (i.e., test vectors are generated and test responses are analyzed on-chip). BIST needs only an inexpensive tester to initialize BIST circuitry and inspect the final results (pass/fail and status bits) [5]. However, BIST introduces extra logic, which may induce excessive power in the test mode, in addition to potential performance penalty and area overhead. BIST circuitry can further be divided into logic BIST for random logic blocks (e.g., control circuitry or data path components) and memory BIST for on-chip memory cores.

B. System-on-a-Chip Test Challenges:

As process technologies continue to shrink, designers are able to integrate all or mostof the functional components found in a traditional system-on-a-board (SOB) onto asingle silicon die, called system-on-a-chip (SOC). This is achieved by incorporatingpre-designed components, known as intellectual property (IP) cores (e.g., processors,memories), into a single chip [6]. While SOCs benefit designers in many aspects, theirheterogeneous nature presents unique technical challenges to achieve high qualitytest, i.e., acceptable fault coverages for the targeted fault models. In the following,several SOC test challenges like Controllability and observability, Volume of test data, tester channel capacity and testing time, Heterogeneous IP cores, At-speed test, Power dissipation, are enumerated along with the motivation for a shift from ATE-based SOC testing to BIST. All the above SOC test challenges need to be overcome in order to reduce the ever-growing cost of manufacturing test while enabling high manufacturing yield and reliability through satisfactory test quality.

C. Embedded Memory Testing:

Memory cells are designed using transistors and/or capacitors, and therefore they cannot be modeled by logic gates. Structural test based on gate level netlist cannot be applied to memory testing. However, as mentioned in the previous section, memory cores have a rather regular structure caused by identical memory cells and very simple functional operations (only read and write) which are very suitable for functional test. Unlike the case of random logic testing, which needs a large set of deterministic test patterns to reach the desired fault coverage, functional test programs for embedded memory cores can be generated by compact and scalable on-chip test pattern generators. Furthermore, since written data is unaltered in a fault-free memory, the expected responses can easily be re-generated on-chip and low overhead comparison circuitry can check the correctness of output responses.

Therefore, the complexity of memory BIST circuit is lower than that of logic BIST. Due to the deterministic nature and high test quality of memory test algorithms, memory BIST has emerged as the state-of-the-art practice in industry. Being parts of an SOC, embedded memories face the same test challenges as SOCs. However, the cost of testing embedded memories has unique characteristics and it is influenced by three major components: cost of ATEs, manufacturing testing time, and DFT and BIST area/performance overhead [7 and 8]. When considering the challenges faced by SOC testing, reduced testability, high volume of test data, heterogeneous IP cores and at-speed test, can all be solved by implementing programmable embedded memory

III. IMPORTANCE OF FAULT TOLERANT DESIGN

There are two fundamentally different approaches that can be taken to increase the reliability of computing systems. The first approach is called fault prevention (also known as fault intolerance) and the second fault tolerance. In the traditional fault prevention approach the objective is to increase the reliability by a priori elimination of faults. Since this is almost impossible to achieve in practice, the goal of fault prevention is reduce the probability of the system failure to an acceptable low value. In the fault tolerance approach, faults are expected to occur during computation, but their effects are automatically counteracted by incorporating redundancy.

Redundancy:

Incorporating additional facilities, into a system, so that valid computation can continue even in th presence of faults. These facilities consists of more hardware, more software or more time, or a combination of all these; they are redundant in the sense that they could be omitted from a fault-free system without affecting its operation. Fault tolerant is not a replacement but rather a supplement to the most important principles.

There different approaches for redundancy. Static redundancy, also known as “masking redundancy”, uses extra components such that the effect of a faulty component is masked instantaneously. Two major techniques employed to obtain fault masking are the triple modular redundancy and the use of error correcting codes. Dynamic redundancy consists of several modules but only operating at a time. If a fault is detected in the operating module it is switched out and replaced by a spare [6 and 5]. Thus dynamic redundancy requires consecutive actions of fault detection and fault recovery. Hybrid redundancy combines the static and dynamic redundancy approaches. The main advantage of the self-purging system over the standard approach is the simplicity of the switching mechanism.

IV. FUNCTIONAL TESTING AND MARCH TEST ALGORITHMS

Based on the used memory fault models, memory test algorithms can be divided into four categories as described below:

- Traditional tests including Zero-One, Check board, GALPAT and Walking 1/0, Sliding Diagonal, and Butterfly. They are not based on any particular functional fault models and over time have been replaced by improved test algorithms, which result in higher fault coverage and equal or shorter test time.
- Tests for stuck-at, transition, and coupling faults that are based on the reduced functional fault model and are called March test algorithms.
- Tests for neighborhood pattern sensitive faults.
- Other memory tests: any tests which are not based on the functional fault model are grouped in this category.

March test algorithms can efficiently test embedded memories and, therefore, the rest of this section provides more details about them [4, 6, 7].

1.) March Test Notation:

A March test consists of a finite sequence of March elements. A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell. For example, ↓ (r1,w0) is a March element and r0 is a March primitive. The address order in a March element can be increasing (↑), decreasing (↓), or either increasing or decreasing (↨). An operation can be either writing a 0 or 1 into a cell (w0 or w1), or reading a 0 or 1 from a cell (r0 or r1). In summary, the notation of March test is described as follows:

↨ *Addressing order can be either increasing or decreasing;*
↑ *Increasing memory addressing order;* ↓ *Decreasing memory addressing order; r0 Read 0 from a memory location;*
r1 Read 1 from a memory location;
w0 Write 0 to a memory location;
w1 Write 1 to a memory location;

2.) March Test Algorithms:

March algorithms are very easy to implement in either software or hardware. A piece of pseudo-code for the MATS+ algorithm is given to demonstrate the basic test procedures. In the code shown below, n is the total number of bits of the memory (bit-oriented memory) and Addr[i] points to the i_{th} memory address for read or write. Line 1 runs the first March element of MATS+ algorithm ↨ (w0). Since the address sequence can be either up or down, here we use an up address sequence. Line 2 to 5 run the second March element ↑ (r0,w1) with the up address sequence. Line 6 to 9 implement the third March element ↓ (r1,w0) with the down address sequence. If any data mismatch happened during the test (line 3 and 7) the program will stop and return fail. Otherwise, it will return success after all March elements are finished [8].

MATS+ Test

```
for (i = 0; i < n-1; i++) Addr[i] = 0;
for (i = 0; i < n-1; i++) {
    if (Addr[i] != 0) return (fail);
    Addr[i] = 1;
}
for (i = n-1; i >= 0; i– –) {
    if (Addr[i] != 1) return (fail);
    Addr[i] = 0;
}
return (success);
```

V. PROPOSED BISR

The present BIST concept prefers a redundancy logic that is placed in parallel to a memory without spare rows and spare columns. There will be no additional delay for the word redundancy logic on top of a memory in the data path of the memory. The memory is repaired during testing by storing faulty addresses in registers.Built-In Self-Repair (BISR) with Redundancy is an effective yield-enhancement strategy for embedded memories. The proposed architecture mainly consists of three modules: BIST module, BIAA module, MUX module.

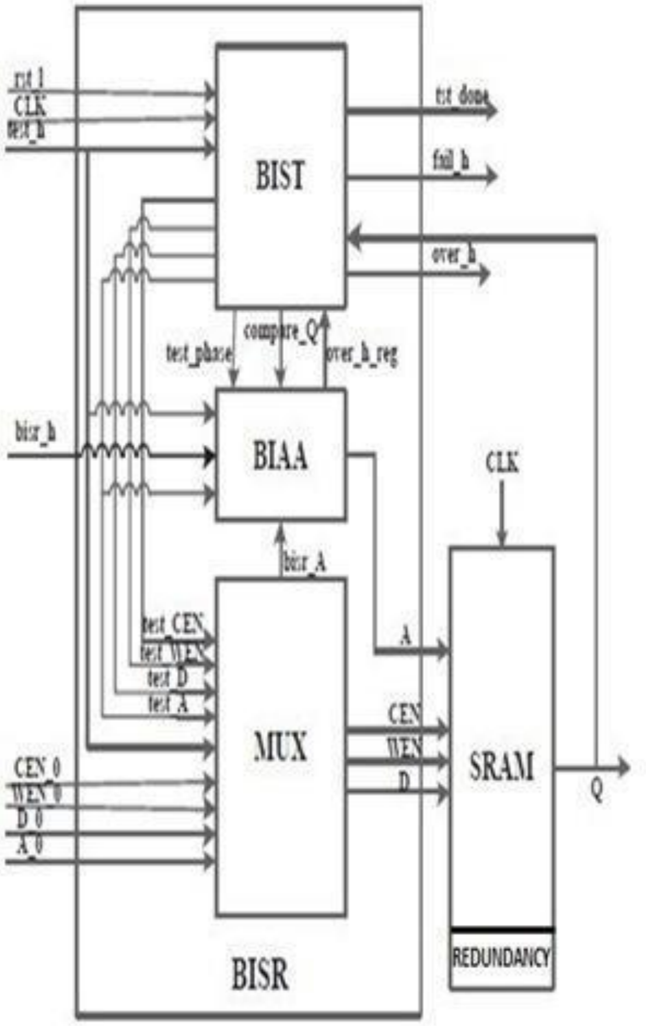

Fig.2: Proposed BISR Strategies

i. BIST module:

It uses March C- to test the addresses of the normal words in SRAM. It detects SRAM failures with a comparator that compares actual memory data with expected data. If there is a failure (compare_Q = 1), the current address is considered as a faulty address.

ii. BIAA module:

It can store faulty addresses in a memory named Fault_A_Mem. counter in BIAA that counts the number of faulty addresses. When BISR is used (bisr_h = 1), the faulty addresses can be replaced with redundant addresses to repair the SRAM.

iii. MUX module:

The inputs of SRAM in different operation modes are controlled by the MUX module.

iv. SRAM module:

In test mode (bist_h = 1), the inputs of SRAM are generated in BISR while they are equal to system inputs in access mode (bist_h = 0). At the system level, BIST functionality can be used in the design phase to characterize digital interface timing between the digital processors and the data converters.

v. Flow chart:

Each fault address can be stored only once into Fault-A-Mem. Assaid before, March C- has 6 steps. In another word, the addresses will be read 5 times in one test. Some faulty addresses can be detected in more than one step. Take Stuck-at-0 fault for example, it can be detected in both 3rd and 5th steps. But the fault address shouldn‟t be stored twice. So we propose an efficient method to solve the problem in BIAA module. Below Fig. 3 shows the flows of storing fault addresses. BIST detects whether the current address is faulty. If it is, BIAA checks whether the Fault-A-Mem overflows. If not, the current fault address should be compared with those already stored in Fault-A-Mem. Only if the faulty address isn‟t equal to any address in Fault-A-Mem, it can be stored. To simplify the comparison, write a redundant address into Fault-A-Mem as background. In this case, the fault address can be compared with all the data stored in Fault-A-Mem no matter how many fault addresses have been stored.

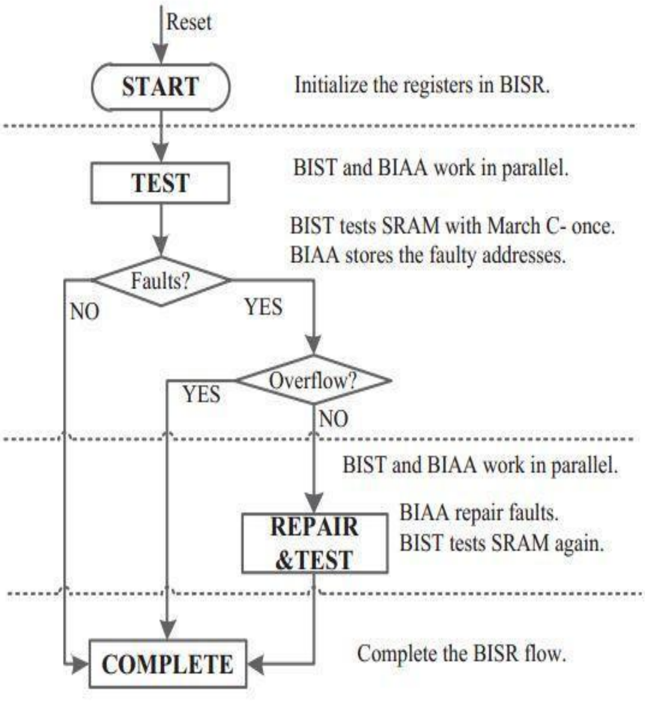
Reset
START
Initialize the registers in BISR.
TEST
BIST and BIAA work in parallel.
BIST tests SRAM with March C- once.
BIAA stores the faulty addresses.
Faults?
NO
YES
Overflow?
YES
NO
BIST and BIAA work in parallel.
REPAIR &TEST
BIAA repair faults.
BIST tests SRAM again.
COMPLETE
Complete the BISR flow.

Fig.3: BISR flow chart

vi. Analysis:

The following flow chart in Fig.3 is used to make the analysis for BISR Strategy. 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. 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 [1, 2]. 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.[8]

VI. SIMULATION RESULTS

The proposed BISR strategy will be simulated to verify output through wave forms. The following simulation waveforms will explain function of BISR strategy.

A. SRAM Module:

The signals in the simulation results are clock, reset, chip enable, and write enable, address are impartment signals. There are 4 data bits and 6 address bit are applied to SRAM. The simulation results are obtained according test bench assigned to the SRAM. The below Fig.4 shows SRAM module simulation results.

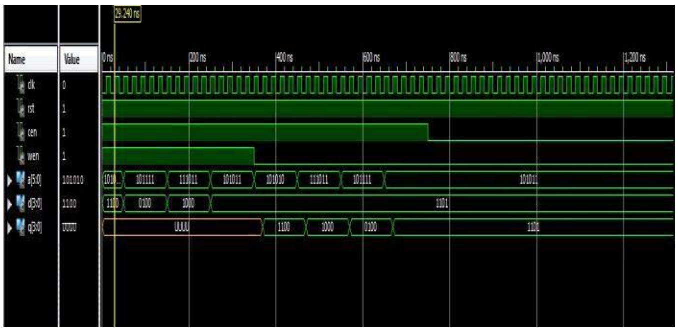

Fig.4: Simulation Wave Forms of SRAM Module

B. BISR Module:

The below Fig.5 shows simulation results for

BISR module. The signals in the simulation results are clock, reset, chip enable, write enable, address are impartment signals. Their are 4 data bits and 6 address bit The signals in the simulation results are clock, reset, test_h, bisr_h, test_done, normal_cen, normal_wen, memory and q.Simulation results are run frame 1600 ns onwards bisr_h is enable. Than repairing operation is perfume and repair the fault address in RAM memory. Output address location can observe frame q[3:0] signal.

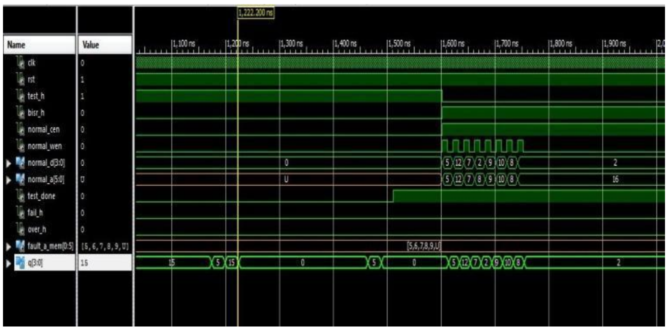

Fig.5: Simulation Wave Forms of BISR Module

C. Multiplexer Simulation Results:

The below Fig.6 shows simulation wave forms of multiplexer module. Test_h is equal to „1‟ up to 1600 ns means testing operation(BISR_h="0‟ means no repairing operation). At 1510 ns algorithm execution is done. So test_done is equal to high. In testing operations fault addresses in memory is stored in fault_a_mem. Fault_a_mem all locations are filled with fault addresses. So over_h value is equal to one and fail_h is equal to one. < 6 faults fail_h is equal to zero. =6 and >6 faults fail_h and over_h is equal to high. =6 there is no effect on operation. We can perform repairing operation. >6 means there are no redundancies for repairing the circuit.BISR_h is equal to „1‟ after 1600 ns means repairing operation.

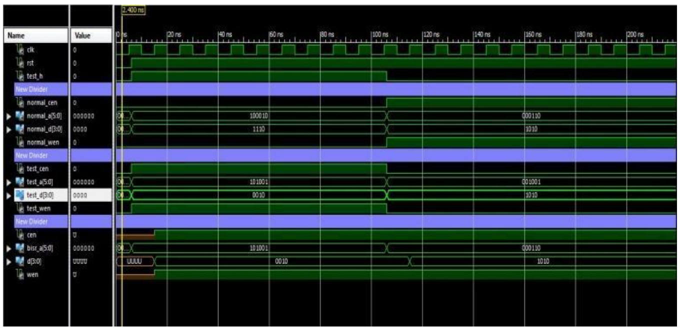

Fig.6: Simulation Wave Forms of Multiplexer Module

VII. CONCLUSIONS

An improved efficient BISR strategy for SRAM IP with selectable redundancy design and implementation presented in this paper. The design provides flexibility to the users so that, they 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 simulation. Embedded memories represent more and more area of system-on-chip (SOC) designs the yield of memory cores dominates the yield of chips. Repair rate simulation under different redundancy analysis algorithms and Spare-element configurations– Helps evaluate BIRA algorithms and develop BISR schemes.

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] 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'l Test Conf. (ITC)*, 2000, pp. 567.574.

[6] V. Schober, S. Paul, and O. Picot, "Memory built-in self-repair using redundant words," in *Proc. Int'l Test Conf. (ITC)*, Baltimore, Oct. 2001, pp. 995-1001.

[7] Y. Zorian, "Embedded memory test & repair: Infrastructure IP for SOC yield," in *Proc. Int'l Test Conf.(ITC)*, Baltmore, Oct. 2002, pp. 340.349.

[8] Parag K. Lala "Fault Tolerant and Fault Testable Hardware Design" BS publications, 1990, pp. 69. 86.

[9] M. Abramovici, M. Breuer, and A. Friedman.Digital Systems Testing andTestable Design. IEEE Press, 1995.

[10] P. H. Bardell, W. H. McAnney, and J. Savir. Built-In Test for VLSI: PseudorandomTechniques, John Willey, NY, 1987

ABOUT THE AUTHORS

Mr. D. Sri Harsha received his B.Tech in Electronics & Communication Engineering (E.C.E) from Indur Institute of Engineering & Technology, JNTUH in 2005 and M.Tech in VLSI SYSTEM DESIGN from Sree Datta Institute of Engineering & Science, JNTUH 2011. He has more than 8+ years of Teaching Experience at various engineering colleges. At present he is working as an Assistant Professor, Department of ECE at Guru Nanak Institutions Technical Campus. He contributed several research articles in International Journals & Conferences. His areas of interest are VLSI & Embedded Systems.

Mr. Dustakar Surendra Rao working as a Assistant Professor in Guru Nanak Institutions Technical Campus, affiliated to JNTU Hyderabad. He has M.Tech degree in VLSI System Design with a proven teaching experience of 8+Years. Mr Surendra have extensive experience in handling labs, conducting workshops and conferences. He taught effectively several subjects. In addition, Mr Surendra also assisted and played a key role in shaping up Electronics Department at his current work place. He delivered numerous guest lectures at several colleges and has nearly 50+ undergraduate projects to his credit. He attended one national conference, two national workshops and three Refresher courses. His area of interest is Low Power VLSI and his Scientific Interest is Electronic Design Automation and Digital IC Design and Verification.