UACEE International Journal of Advancements in Electronics and Electrical Engineering – IJAEEE
Volume 2 : Issue 2 [ISSN 2319 – 7498]
Publication Date : 05 June 2013

An efficient BISD Scheme for Diagnosis of Coupling Faults

[Kirtisova Behera Manojit Panda Deepak Agarwal]

Abstract*—As the use and density of memories in electronic circuits is growing more and more, testing of memories and diagnosing various faults present in them are becoming more and more prominent now-a-days. In this paper, two new types of hardware BISD circuits are designed for bit oriented memories. The proposed BISD circuits can not only detect the coupling faults but also locate the address of the victim cell as well as that of the aggressor cell in the presence of a coupling fault. Above all this circuit is very simple, easy to design and is expected to reduce the testing time compared to the software based testing methodologies.

Keywords—Memory, *fault model, March Test, BISD*

I. Introduction

In recent years, memories have become the most universal component. Most of the electronic circuits contain embedded memory. As the VLSI technology moving into deep submicron level, the density of memory is growing day by day. Due to this dense integration, various types of faults are encountered in memories which in fact decrease the performance of the whole circuit. So testing of semiconductor memories and detecting various types of faults present in them is going to be of utmost importance. [1-4]Built-in self-test has been proven to be one of the most cost-effective and widely used solutions for memory testing, as it does not require external test equipment, consumes very less time and generates on-chip test pattern to provide higher controllability and observability. The memory testing algorithm plays a very important role in the diagnosis of memory. The algorithms implemented to test memories can also be classified into two types: Classical tests and March based tests. Some classical tests like Zero-One and Checkerboard are simple and fast but having poor fault coverage whereas tests like Walking zero-one and GALPAT have a better fault coverage but with a large testing time[5-7]. As compared to these tests, March based tests are simple having higher fault coverage for which they are being used widely in most modern memory BIST [8].

[9] Discusses a set of March tests together with methods to make composite tests for collections of fault types. In [10] With 17N Read/write operations, the algorithm for bit-oriented memories can distinguish between stuck-at fault, state coupling fault, idempotent coupling fault, and inversion coupling fault.

Kirtisova Behera, Manojit Panda, Deepak Agarwal
National Institute of Technology, Durgapur
India

Moreover, the aggressor cell in case of a CF can be located by applying an additional March-like algorithm with 3N complexity. In addition to testing the embedded memories using mach algorithm [9] and [10], diagnosis of fault sites and repair by the redundant bit-lines to increase the yield is necessary for large cores. Therefore, in [11], [12] built in self diagnosis (BISD), built-in-self repair (BISR) and built in self redundancy analysis (BIRA) technologies are becoming inevitable, so far as overall test cost is concerned.

In this paper, we have introduced two new schemes of hardware BISD circuits for embedded memories. The proposed BISD circuits are capable of detecting faults as well as the locations of aggressor and victim cells in case of any coupling fault. Here, the 1st BISD hardware approach is used to locate the address of victim cell for coupling fault. Upon receiving the address of victim cell, the second BISD circuit, based on 3N algorithm [10], can locate the aggressor cell. The discussed circuits in this paper are structurally very simple and easy to design. Moreover, these circuits reduce test time and test cost and thereby improving the yield of memory.

The paper is organized as follows. Section 2 defines fault models and notations. In section 3, the 17N diagnosis algorithm for bit oriented memories and algorithm for locating the coupling faults are described. Section 4 introduces the BISD hardware circuits for locating the victim and aggressor cell. Finally section 5 concludes the paper.

II. Fault Models

A fault model is an engineering model of something that could go wrong in the construction or operation of a piece of equipment. From the model, the designer or user can then predict the consequences of this particular fault. Fault models can be used in almost all branches of engineering. As the cell array dominates the silicon area of the memories, the faults within the cell array are considered. The fault models that are prominently found in semiconductor memories are considered here.

A. Stuck-At Fault

A stuck-at fault is a particular type of fault where the logic value of a line or cell sticks to one particular value (either 0 or 1).This usually happens when a cell or line gets shorted to either supply(1) or ground(0). Stuck-at fault can be of two types depending on the value the faulty cell or line permanently holds. If the cell gets shorted with the supply, it is

called stuck-at-1 fault (S-A-1) and if it is shorted to ground then it is called stuck-at-0 fault (S-A-0).

B. *Transition Fault*

Transition fault (TF) can be viewed as a special type of SAF. This is the case when a cell fails to undergo a transition from one particular logical value to another, i.e. either 1 to 0 or 0 to 1. If a cell fails to undergo up transition (0→1), then the fault is called up transition fault denoted as <↑/0> and if a cell fails to undergo down transition (1→0), then the fault is called down transition fault which is denoted as <↑/1>.

C. *Coupling Fault*

It involves two cells, A-cell (Aggressor cell) and V-cell (Victim cell). If any operation performed to the A-cell forces or changes the state of the V-cell, this is said to be coupling fault. A-cell is coupling cell and V-cell is coupled. Coupling faults (CF) can be of three types.

$CF(A_p,A_s,V_s)$ represents the Coupling fault for bit-oriented memory, where $A_p \in \{H,L\}$ represents the relative position (higher or lower) of the aggressor with respect to the victim, $A_s \in \{0,1, \uparrow,\downarrow\}$ represents the state of the aggressor cell that activates the fault, and $V_s \in \{0,1, \updownarrow\}$ is the faulty state of the victim cell. The symbol ↕ stands for either ↑ or ↓. For example, CFin(H,↓, ↕) represents an inversion coupling fault where the possible aggressor is located at a higher address than the victim, and when the aggressor undergoes a down transition, the victim is forced to invert its value.

1) **Inversion CF**: A transition (↑ or ↓) write operation to the A-cell toggles or inverts the contents of V-cell. 0 to 1 (or 1 to 0) transition in one cell inverts the content of a second cell. A CFin can be thought of as a D flip-flop with an extra clock input and the Q' output tied to the D input as in Fig. 1(b).

2) **Idempotent CF**: A CF whereby the transition write operation (0 to 1 or 1 to 0) applied to the A-cell forces the state of the V-cell to a certain value '0' or '1'. An idempotent coupling fault can be thought of as an S/R-type flip-flop with an OR-gate in the Set or Reset line as shown in Fig. 1(a).

3) **State CF**: A CF whereby the state of A-cell forces the state of V-cell to a fixed value is said to be State coupling Fault. The CFst is of four subtypes: <1;0>, <1;1>, <0;0> and <0;1>. It can be thought of as a D-type flip-flop with an OR/AND-gate in the data line (D).

D. *Neighborhood Pattern Sensitive Fault*

NPSF can be defined as a fault model which is somewhat similar to coupling faults but in this case the no. of aggressor cells is more than one.

1) **Active NPSF**: The base cell changes its contents due to changes in the neighborhood pattern. A test that has to detect and locate ANPSFs should satisfy the following requirement: each base cell must be read in state 0 and in state1 for all possible transitions in the deleted neighborhood pattern. There are two different possible values for the base cell (0 and 1), k-1 ways of choosing the deleted neighborhood cell which must undergo one of two possible transitions (↑ or ↓), and 2^{k-2} possibilities for the remaining neighborhood cell contents. The total number of active neighborhood patterns (ANPs) is $2^* (k-1)^*2^*2^{k-2} = (k-1)^*2^k$.

TABLE I. FAULTS COVERED BY 17-N ALGORITHM

Stuck At Faults	Coupling Faults		
SAF	Static Coupling Fault(CFst)	Idempotent Coupling Fault(CFid)	Inversion Coupling Fault(CFin)
SAF(0)	CFst(L,0,0)	CFid(L,↑,1)	CFin(L,↑,↕)
SAF(1)	CFst(H,0,0)	CFid(L,↑,0)	CFin(L,↓,↕)
	CFst(L,0,1)	CFid(L,↓,1)	CFin(H,↑,↕)
	CFst(H,0,1)	CFid(L,↓,0)	CFin(H,↓,↕)
	CFst(L,1,0)	CFid(H,↑,1)	
	CFst(H,1,0)	CFid(H,↑,0)	
	CFst(L,1,1)	CFid(H,↓,1)	
	CFst(H,1,1)	CFid(H,↓,0)	

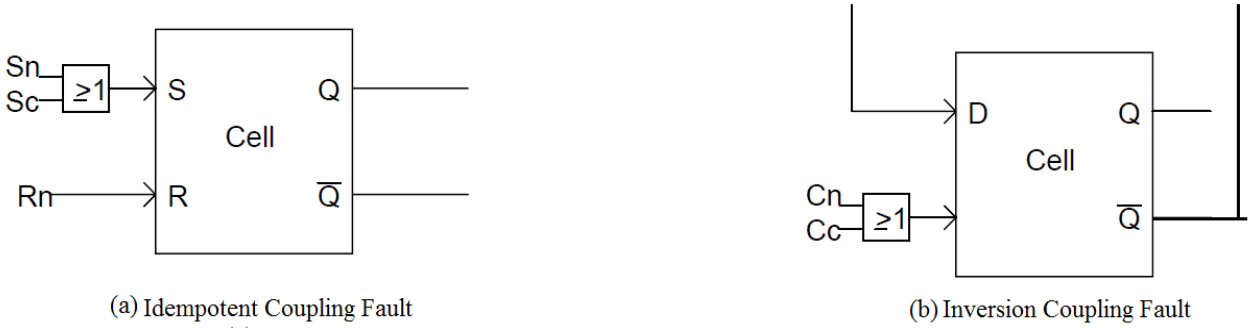
2) **Passive NPSF:** The contents of the base cell cannot be changed due to a certain neighborhood pattern. The necessary condition to detect and locate PNPSF: for each of the2^{k-1} deleted neighborhood patterns, the two possible transitions up and down of base cell must be verified. Therefore, the total number of PNPSFs is $2^*2^{k-1} = k^*2^k$. The total pattern count for active and passive neighborhood pattern sensitive fault APNPSFs is therefore, $(k-1)^*2^k + 2^k = k^*2^k$.

3) **Static NPSF**: The contents of a base cell is forced to a certain or particular state due to a certain neighborhood pattern. The necessary condition to detect and locate SNPSF is that we must apply the 2^k combinations of 0s and 1s to the k-cell neighborhood, and verify by reading each cell that each pattern can be stored or not. It differs from Active and Passive NPSF such that it need not have a transition to sensitize the SNPSF.

The faults that are covered under the algorithm discussed in this paper are given in Table-I.

III. March Algorithms

March tests are widely used to test and diagnose the semiconductor memories if any fault found in it. The linear complexity of these March tests computes with respect to the number of memory cells which are to be tested. While comparing them with the traditional testing methods, March tests are found to be less time consuming as well as covering more faults effectively.



(a) Idempotent Coupling Fault (b) Inversion Coupling Fault

Fig. 1- Logical explanation for Various types of Coupling Faults

A March test usually consists of a number of March elements represented as Ms where's' specifies the March sequence number [9]. Each March element consist a certain number of Read and/or Write operations to all cells according to a predefined address order which may be ascending (⇑), descending (⇓), or either (⇳). March algorithm is designed to detect the fault in semiconductor memories. But the same March algorithm cannot be used for diagnosis. Diagnosis in this context means finding the fault type with the help of syndrome and then locating the faulty cells. After applying any March test, the generated syndromes are then compared to the fault dictionary and the type of fault can then easily be determined. Our current interest is to design hardware for locating the faulty cells to cover all the coupling faults.

A. *Algorithm for Syndrome Generation of Coupling Faults*

In this paper we have considered March-17N test due its coverage of all Stuck-at as well as coupling faults [10]. The March-17N RAM diagnosis algorithm is given below. Also among all the March algorithms, March 17N diagnosis algorithm has the lowest time complexity.

{ ⇳(w0); ⇑(r0,w1,r1); ⇳(r1); ⇑(r1,w0,r0);⇳(r0); ⇓(r0,w1,r1); ⇳(r1); ⇓(r1,w0,r0); ⇳(r0)}

For a given test algorithm, the corresponding dictionary of fault syndromes is constructed each row of which corresponds to a certain fault class. From the Table I, CFst(L,0,1) means that state coupling fault, when the value of aggressor cell is 0, with the address lower than the victim cell (indicated by an L), then the victim cell is forced to 1; CFin(H,↑,↕) means inversion coupling fault, i.e. if there is a transition arising in the aggressor cell with the address higher than the victim cell (indicated by an H), the content of the victim cell will be inverted; and so on. Fig.2 describes the fault free content of a 4-bit memory. The read values of second March element from each cell give the logic 0, since previous (i.e. first March element) March operation has written logic 0. Fig. 3 describes that if the second cell (i.e. address 01) gives the value 1 that indicates that cell is faulty and called Victim cell.

B. *Algorithm for locating the aggressor cell of coupling faults*

March-like algorithm for locating the aggressor cell was reported in [10]. Assume that the position (address) of the victim of a CF is represented by v and V denotes the fault-free state of the victim. [10]The algorithm for locating the aggressor is shown as follows:

$$\{\Uparrow (w\bar{A}); w_v V; \Uparrow(wA, r_v V)\}$$

Or

$$\{\Downarrow (w\bar{A}); w_v V; \Downarrow(wA, r_v V)\}$$

The symbols ⇑ means that the operations are performed from 0 to (v – 1) and ⇓ means the operations performed from (N – 1) to (v + 1). w_v and r_v, represent the Write and Read operations that are performed only to victim cell address v. Also, the value A is determined by the state of the aggressor of the CF after the above diagnosis test. For example, if we want to locate the aggressor of a CFid (L,↑,0), then we have to take A=l.

The worst case complexity of above algorithm is 3N since the position of victim cell is 0 or N - 1. Depending on whether the possible aggressor cell is located lower or higher than the victim cell, one may select only the first or second part of the algorithm. E.g., to locate the aggressor of CFin (L,↑,↕), the 1st March algorithm is selected. For example, CFid (L,↑,0) has been diagnosed by the March-17N algorithm and the faulty cell is the second bit in an 8-bit memory array. Here A=1 is selected, so algorithm given below is applied.

$$\{\Uparrow (w0); w_1 1; \Uparrow (w1, r_1 1)\}$$

The first two Write operations are used to initialize the memory. The last March element writes 1 from 7th down to the 3rd bit and at same time reads 1 (expected) in the 2nd bit.

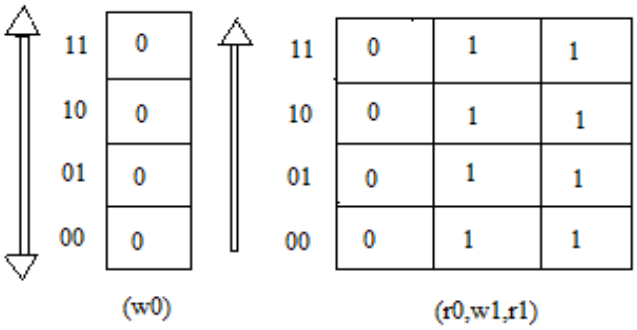


Figure 2. Fault free Content of a Memory

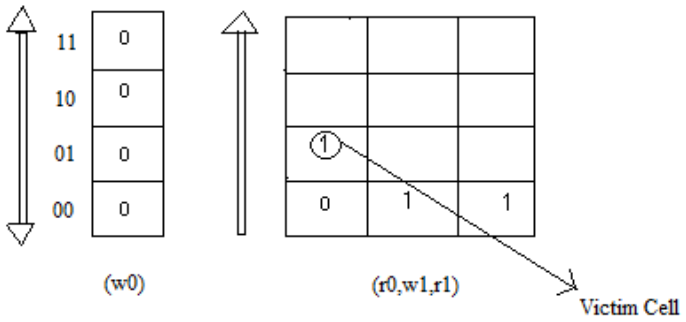


Figure 3. Faulty content of a Memory

IV. BIST implementation

In this section, first we have described our hardware approach to locate the victim cell of a coupling fault. Next, the 3N algorithm I used to design the hardware scheme to locate the aggressor cell, depending on the victim cell. In this case we have considered the location of aggressor cell is lower than the victim cell.

A. *Circuit to locate the Victim cell*

The fig. 3 shows the hardware implementation of BIST to locate the address of victim cell. Here the test controller is used to controls the application of the different (March) test phases, the address generation by, e.g., a linear-feedback shift register (LFSR), and application of the data and control signals such as R/W.

A comparator is used to compare the read and write bit. If no error has been found then the output will be 0, otherwise it will be 1. This output is fed to a demultiplexer as select line as well as to the two m-bit latches as enable line where m is the number of address lines of memory. For 1st latch, error line is fed directly whereas the 2nd latch is fed by inverting it. Demultiplexer input is coming from address generating LFSR.

During test application, when memory faults are detected, the comparator output goes to 1 and the demultiplexer passes the faulty address through O1 (victim cell address). If there is no error present comparator output will be 0 and demultiplexer will pass the input (fault free address) through the O0. We can get the same address at the output of the 2nd latch.

B. *Circuit to locate the Aggressor cell*

In this paper, a special type of circuit is proposed which can identify the location of the aggressor cell in the presence of a coupling fault. The circuit in Fig.4 can get the address of the victim cell and the type of coupling fault present in it. Using this information, circuit presented in fig.5 can easily find out the location of the aggressor cell.

In this circuit, we have placed a special controlling circuit inside the test controller named as 'March Operation Controller' which is used to control the March operation. From Table-2, we get the output expression for March operation controller. In this table, X and Y represent March element number whereas Z represents operation r/w for each March element. E.g. the test algorithm in [10] to locate the address of aggressor cell is

$$\{\Uparrow (w\bar{A});\ w_v V;\ \Uparrow(wA, r_v V)\}$$

Or

$$\{\Downarrow (w\bar{A});\ w_v V;\ \Downarrow(wA, r_v V)\}$$

Suppose we have taken the address of aggressor cell lower than that of the address of the victim cell. Therefore we will consider the 1st algorithm. In that algorithm, the 1st March element (m0 i.e. xy=00) is $\Uparrow(w\bar{A})$, 2nd March element (m1 i.e. xy=01) is w_vV and the 3rd March element (m2 i.e xy=10) is $\Uparrow(wA, r_v V)$. Operation number for wA is 0 and for r_vV is 1 [Table I]. For each March element, z=0 represents write operation and z=1 represents read operation [table III]. Table IV explains the March operation controller value.

We have considered the address of aggressor cell to be lower than the victim cell. The victim cell address is stored in a buffer with the help of the circuit in Fig. 4. There is an address generator which generates address from 0 to v-1. The AV controller controls the address of the victim as well as the aggressor cell through an MOC signal, during the cycle of March sequence. This signal comes from the output of the March element counter present inside Test controller. When the MOC signal is 0, it passes the addresses from 0 to v-1 i.e. aggressor cell address and when the signal is 1, it passes the victim cell address to memory. Test controller is used to control the application of different (March) test phases, MOC signal, and application of data and controls the signals such as r/w.

During the test application, output data is available for the 3rd March element in the algorithm for read operation which is fed to the comparator as one input with the 2nd input being the true value of the victim cell. If the comparator output becomes 1 then the address of aggressor cell is latched.

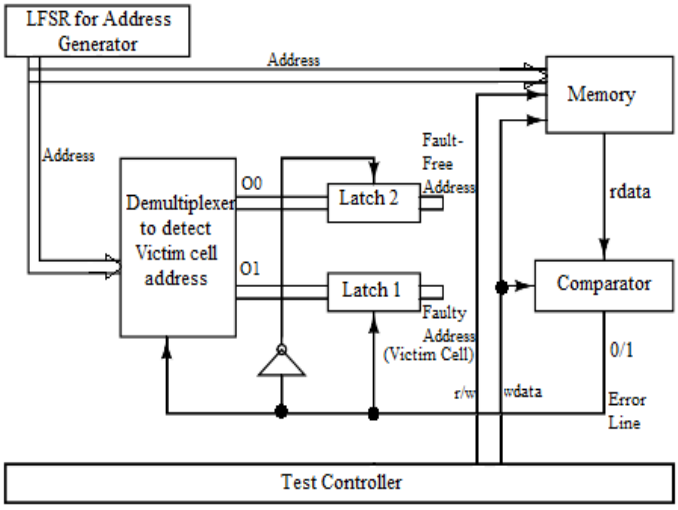


Figure 4. Hardware BIST design to locate the address of Victim Cell

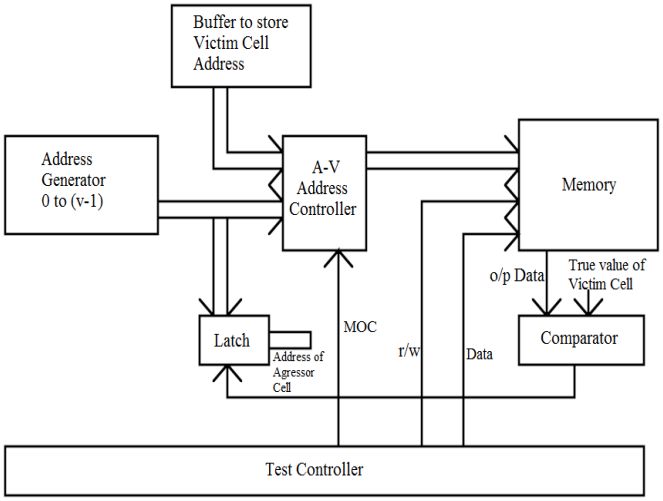


Figure 5. Hardware BIST Design to find the address of the Aggressor Cell

TABLE II. MARCH ELEMENT

X	Y	Element
0	0	M0
0	1	M1
1	0	M2

TABLE III. MARCH OPERATION

Z	Operation
0	Write
1	read

TABLE IV. March Element Controller

X Y Z	F = march operation controller value
0 0 0	0
0 1 0	1
1 0 0	0
1 0 1	1

Conclusion

In this paper two hardware based BISD designs are proposed for Bit-oriented memory. The main purpose of this approach is to find the location of the victim cell and aggressor cell of the coupling fault in the memory. As a result, we are able to locate a large number of coupling faults at a lower time complexity. Hence, the yield of the memory is improved.

Acknowledgment

We would like to convey our utmost respect and hearty thanks to our project guide Ms. Mousumi Saha, Asst. Prof, Dept. of Computer Applications, NIT Durgapur for giving her invaluable time from her busy schedule, sharing her knowledge at each and every step and guiding us throughout our work on this project.

References

[1] B. E Cockburn, "Tutorial on semiconductor memory testing",J. Electronic Testing: Theory and Application, vol. 5,pp. 321-336, 1994.

[2] P. Camurati, P. Prinetto, M. S. Reorda, S. Barbagallo,A. Burri, and D. Medina, "Industrial BIST of embedded RAMs", IEEE Design & Test of Computers, vol. 12, no. 3,pp. 86-95, Fall 1995.

[3] C.-W. Wu, "Testing embedded memories: Is BIST the ultimate solution?", in Proc. Seventh IEEE Asian Test Svmp.*(ATS)*, Singapore, Dec. 1998, pp. 516-517.

[4] C.-T. Huang, J.-R. Huang, C.-E Wu, C.-W. Wu, and T.-Y. Chang, "A programmable BIST core for embedded DRAM",IEEE Design & Test of Computers, vol. 16, no. I , pp. 59-70,Jan.-Mar. 1999.

[5] Breuer .A;Friedman A.d Diagnosis and reliable design of digital systems.Computer science Press,Woodland Hills,california 1976.

[6] Knaizuk J;Hartman C.An optimalalgorithm for tesing stuck- at faults in random access memories.IEEE TC. Vol. C-26,pp.11411144,1977.

[7] A.J van de Goor A.J Testing semiconductor memories theory and practice.John wiley & sons,Chihester,UK,1991.

[8] Allen C.cheng , "COMPREHENSIVE STUDY ON DESIGNING MEMORY BIST :ALGORITHM,IMPLEMENTATION AND TRADE OFFS"Digital system testing,project report.

[9] A. J. van de Goor, "Using March tests to test SRAMs**",**IEEE Design & Test of Computers, vol. 10, no. 1, pp. 8-

14, Mar. 1993.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[10] J.-F.Li,K-L.cheng,C.-T.Huang,and C.-W.Wu,"March based RAM diagnosis algorithm for stuck-at and coupling fauls",Proc.IEEE ITC,2001,pp.51-55.

[11] I.Kim,Y.Zorian,G.komoriya,H.pham,F.P.Higgins,and J.L.Lweandowski,""Built in self repair for embedded high density SRAM",in proc.IEEE VLSI systems(DFT),Albuqueque,Nov.1999,pp 1112-1119.

[12] R.P reuer and V.K. Agarwal,"Built-in self-diagnosis for repairable ambeded RAMs",IEEE Design & Test of computers,vol.10,no.2,pp.24-33,une1993.

.

Kirtisova Behera received her B.Tech degree in Electronics and communication Engineering from Biju Pattnaik University of technology, Rourkela, Odisha in the year 2007.She is now pursuing her M.Tech degree in Microelectronics and VLSI from National Institute of Technology, Durgapur. Her area of interest includes VLSI design, memory testing and Diagnosis.

Manojit Panda got his B.Tech. degree in Electronics and Communication Engineering from Biju Pattnaik University of Technology, Rourkela, Odisha in the year 2010. He is now pursuing his M.Tech. degree in Microelectronics and VLSI from National Institute of Technology, Durgapur. His area of interest includes digital system design, circuit and memory testing.

Deepak Agarwal has completed his B.Tech Degree from Gautam buddha Technical University, Lucknow India in Electronics & communication Engg. He is now Pursuing his M.Tech degree in Microelectronics & VLSI from National Institute of Technology, Durgapur, India. His area of interest includes Digital Design, microprocessor and memory testing.