High Speed Test Architecture for SRAM using Modified March Algorithm

D. Viswabharathi*, K. Raghuram, G. Rajesh Kumar*****

*(Department ofElectronics & Communications Engineering, Pragati Engineering College)
** (Department ofElectronics & Communications Engineering, Pragati Engineering College)
*** (Department ofElectronics & Communications Engineering, Vishnu Institute of Technology)

ABSTRACT

Fault diagnosis and location in semiconductor Random Access Memories (RAM) are of prime importance in connection with the increasing density and dominating portion of embedded memories in system-on-chips (SOC). Manufacturing defects should be detected, diagnosed and located for further repair in order to improve the product quality, reliability and yield. It becomes highly important to test various kinds of defects rapidly and precisely to reduce the testing cost and to improve the memory quality, especially in a SoC (system-on-a-chip) design environment. Memory defects can be modeled as stuck-at, coupling, transition, address decoder, and pattern-sensitive faults. Industry-wide use of memory fault models and March test algorithms, with special emphasis on memory fault simulation and test algorithm generation are discussed here and an improved version of March test algorithm for high speed memory testing is implemented.

Keywords - Random Access Memory (RAM), System-on-Chip (SOC), Stuck-at Faults, Coupling Faults, Transition Faults, Address Decoder Faults, Pattern-Sensitive Faults, March Test Algorithm

I. INTRODUCTION

It becomes highly important to test various kinds of defects rapidly and precisely to reduce the testing cost and to improve the memory quality, especially in a SoC (system-on-a-chip) design environment. Memory defects can be modelled as stuck-at, coupling, transition, address decoder, and pattern-sensitive faults, and it is known that the 80 % of the failures are due to leakage defects [11]. Among the different testing algorithms ranging from O(p n) to O(nlog(n)), BIST(built-in self test) techniques with O(p n) to O(n) complexity algorithms have been widely adopted for embedded memories [2]. Memory test patterns can be generated deterministically or randomly [3] through either test equipment or BIST circuitry. Test patterns generated randomly can detect not only modelled defects but also nonmodelled and timing defects [4, 6] nevertheless, deterministic march patterns, for their simplicity, are widely adopted for BIST and chip testing. A few hardwired memory BIST techniques have been developed [7], and recently some microcoded memory BIST circuits were implemented for embedded Memories [8]. In general microcoded memory BIST techniques have great exibility in applying di_erent combinations of test patterns for static and dynamic defects. Memory retention faults, as well as conventional static faults, are major targets, with a register or SRAM storing microcodes [1, 9]. We introduce a different microcoded BIST technique which aims to capture address decoder open faults in addition to conventional static faults. Furthermore, a certain degree of neighborhood pattern-sensitive faults are detected by cellular-automata-based address and pattern generators.

II. BUILT IN SELF TEST (BIST)

Figure 1 shows the BIST hardware architecture in more detail. Basically, a design with embedded BIST architecture consists of a test controller, hardware pattern generator, input multiplexer, Circuit Under Test (CUT). Optionally, a design with BIST capability may include also the comparator and Read-Only-Memory (ROM). As shown in Figure 1, the test controller is used to control the test pattern and test generation during BIST mode. Hardware pattern generator functions to generate the input pattern to the CUT.

Normally, the pattern generator generates exhaustive input test patterns to the CUT to ensure the high fault coverage. For example, a CUT with 10 inputs will require 1024 test patterns. Primary Inputs are the input for CUT during the non-BIST mode or in other word, functional mode. Input multiplexer is used to select correct inputs for the CUT for different mode.

During BIST mode, it selects input from the hardware pattern generator while during functional mode, selects primary inputs. Output response compactor acts as compactor to reduce the number of circuit responses to manageable size that can be used as the signature and stored on the ROM. Implementation of the pattern generation as well as the response compactor will be discussed in more details in section below.

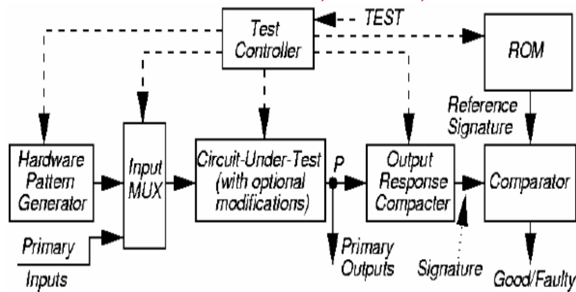

Fig. 1 BIST Architecture

As mentioned earlier, a BIST block can optionally consist of a ROM and a comparator. ROM is used to store the golden signature obtained from simulation at the pre-silicon phase. A comparator is used to compare the signature obtained during BIST mode with the golden signature. If the signature matched with the golden signature, then the chip is considered as fault free. On the other hand, if the signature is not matching with the golden signature, then the chip is considered as faulty.

From Figure 3.2, the wires from primary inputs to the input multiplexer and the wires from circuit output P to primary outputs cannot be tested by BIST. These wires require another testing method such as an external ATE or JTAG Boundary Scan hardware.

III. MARCH TEST ALGORITHM

Figure 2, taken from [19], depict the block diagram of a BISR scheme, including the BIST module, BIRA module, and test wrapper for the memory. The BIST circuit detects the faults in the main memory and spare memory and is programmable at the March element level [19]. The BIRA circuit performs redundancy allocation. The test wrapper switches the memory between test/repair mode and normal mode. In test/repair mode, the memory is accessed by the BIST module, whereas in normal mode the wrapper selects the data outputs either from the main memory or the spare memory (replacing the faulty memory cells) depending on the control signals from the BIRA module. This BISR is a soft repair scheme therefore, the BISR module will perform testing, analysis, and repair upon every power up. As figure 4.1 indicates, the BIST circuit is activated by the power-on reset (POR) signal. When we turn on the power, the BIST module starts to test the spare memory.

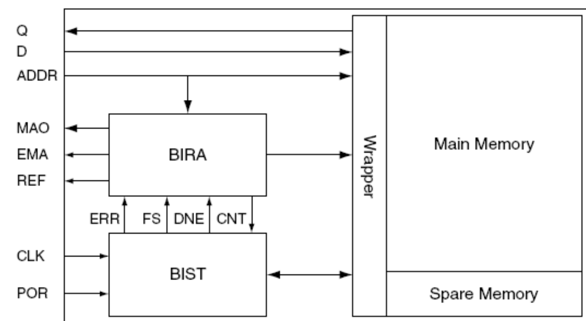

Fig: 2 Block diagram of a BISR scheme [5].

Once a fault is detected, the BIRA module is informed to mark the defective spare row or column as faulty through the error (ERR) and fault syndrome (FS) signals. After finishing the spare memory test, the BIST circuit tests the main memory. If a fault is detected (ERR outputs a pulse), the test process pauses and the BIST module exports FS to the BIRA module, which then performs the redundancy analysis procedure. When the procedure is completed and the memory testing is not yet finished, the BIRA module issues a continue (CNT) signal to resume the test process. During the redundancy analysis procedure, if a spare row is requested but there are no more spare rows, the BIRA module exports the faulty row address through the export mask address (EMA) and mask address output (MAO) signals.

The memory will then be operated in a downgraded mode (i.e., with smaller usable capacity) by software-based address remapping. If downgrade mode is not allowed, AO is removed and EMA indicates whether the memory is repairable. When the main memory test and redundancy analysis are finished, the repair end flag (REF) signal goes high and the BIRA module switches to the normal mode. The BIRA module then serves as the address remapped, and the memory can be accessed using the original address bus (ADDR). When the memory is accessed, ADDR is compared with the fault addresses stored in the BIRA module. If ADDR is the same as any of the fault addresses, the BIRA module controls the wrapper to remap the access to spare memory.

Although BIST schemes, such as the one in the previous example, are promising, a number of challenges in memory testing must be considered. For example, BIST cannot replace external memory testers entirely if the BIST schemes used are only for functional testing. Even BIST with diagnosis support is insufficient because of the large amount of diagnosis data that must be transferred to an external tester, typically through a channel with limited bandwidth. Furthermore, memory devices normally require burn-in to reduce field failure rate. For logic devices IDDQ is frequently used during

burn-in to detect the failing devices, but IDDQ for memories is difficult. What, then, should be done to achieve the same reliability requirement when we merge memory with logic? The combination of built-in current sensors and BIST is one possible approach, and the memory burn-in by BIST logic is another.

One of the most efficient RAM test algorithms currently in use, in terms of test time and fault detection capability, is the March algorithm [16]. This algorithm has a test time on the order of 22N, where N is the number of address locations. In addition to classical stuck-at faults, this algorithm is capable of detecting pattern sensitivity faults, intra word coupling faults, and bridging faults in the RAM. For word-oriented memories, a background data sequence (BDS) must be added to detect these faults within each word of the memory. March Algorithm which is generally used for designing testable SRAMs are described below.

{↕(w0)↓(r0,w1,r1,w1,r1)↓(r1,w0,r0,w0,r0)↑(r0,w1,r1,w1,r1) ↑(r1,w0,r0,w0,r0) ↕(r0)}

The basic notations used in algorithm are as follows:
↑: address n-1 to 0
↓: address 0 to n-1
↕: either way
w0: Write 0 to the word
w1: Write 1 to the word
r0: Read a cell whose value should be 0
r1: Read a cell whose value should be 1

IV. MODIFIED MARCH ALGORITHM

Assume that the memory array under test has R rows and C columns. A typical march algorithm (such as Modified March algorithm) could consist of four steps as shown below.

↕ (W0); ↑(R0,W1,R1); ↓(R1,W0,R0); ↕ (R0)

The basic notations used in algorithm are as follows:
↑: address 0 to n-1
↓: address n-1 to 0
↕: either way
W0: Write 0 to the word
W1: Write 1 to the word
R0: Read a cell whose value should be 0
R1: Read a cell whose value should be 1

This algorithm is described using Verilog HDL. Figure 3 shows the Verilog hierarchical diagram of the system.

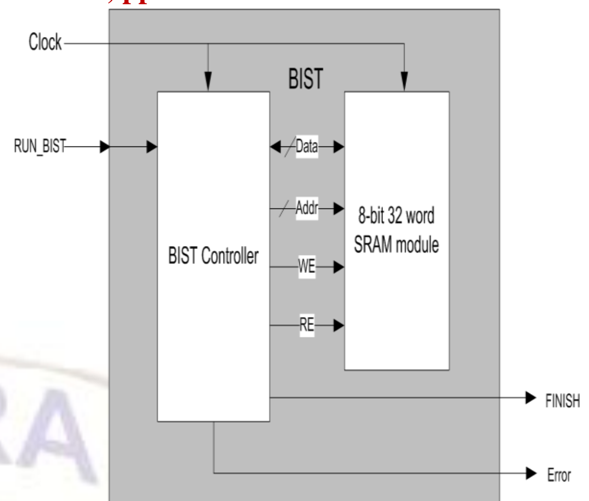

Fig: 3 Verilog hierarchy.

Here it is presented graphically how the FSM works and how the testing is done with March Algorithm. Take SRAM of Size 32 Bytes. For this SRAM testability is included. In March Algorithm, first of all Write all 0's from top to bottom or from bottom to top. Then read data from last address and compare with "00000000". If it is matched replace "00000000" with "11111111" then read data from the same location and compare with "11111111". If it is matched with it decrement the address. If not error signal goes high. Decrement the address till it reaches first location. If it reached first location then do the read data from first location, compare with "11111111'. If it matched write "00000000" in the first location. Again read and compare with "00000000". Do this process from top to bottom. After that again Read and compare with "00000000" from top to bottom or bottom to top.

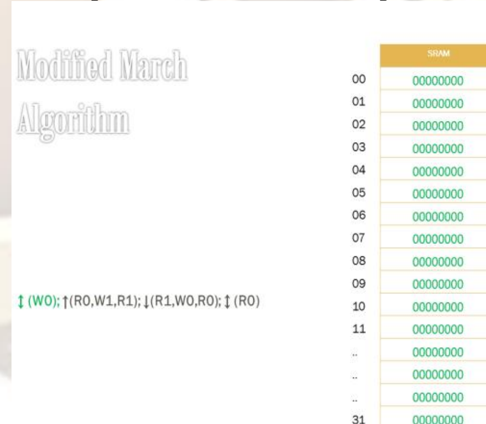

Figure 4: SRAM chip at ↕ (W0) stage

Figure 4 shows the graphical representation of SRAM chip which is controlled by BIST FSM. When BIST is running and it is at ↕ (W0) stage it loads "00000000" at each location in the SRAM chip. That stage we can observe in above figure.

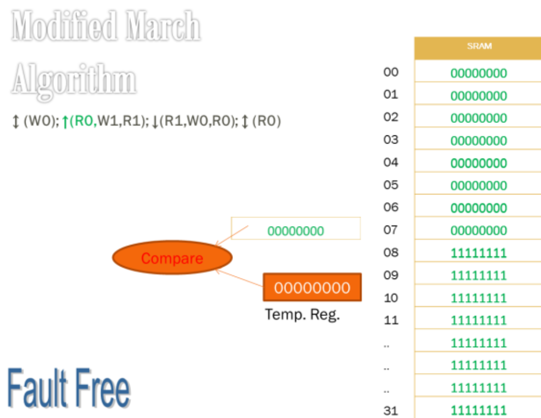

Figure 5: SRAM chip at↑(R0,W1,R1)stage

Figure 5 shows the graphical representation of SRAM chip which is controlled by BIST FSM. Here BIST is running and it is at ↑(R0,W1,R1)stage. After writing “00000000” in all the locations of this SRAM chip, Now the FSM is in ↑(R0,W1,R1)stage. Figure 5 indicates up to this stage SRAM chip working properly.

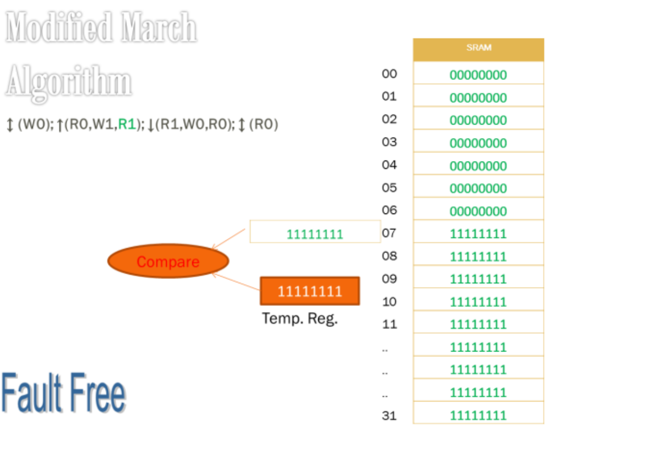

Figure 6: SRAM chip at↑(R0,W1,R1)stage

Figure 6 shows the graphical representation of SRAM chip which is controlled by BIST FSM. Here BIST is running and it is at ↑(R0,W1,R1) stage. After writing “11111111”, now the FSM read a data byte from the same location and compares it with “11111111”. Figure 6 indicates up to this stage SRAM chip working properly.

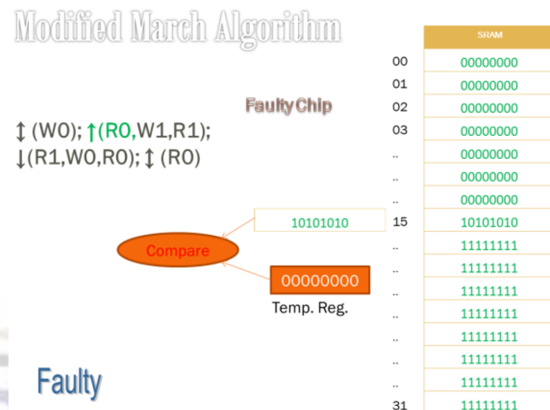

Figure 7: Faulty SRAM chip at↑(R0,W1,R1)stage

Figure 7 shows the graphical representation of SRAM chip which is controlled by BIST FSM. Here BIST is running and it is at ↑(R0,W1,R1)stage. After writing “00000000” in all the locations of this SRAM chip, Now the FSM is in ↑(R0,W1,R1)stage. Here at 15th address during write operation there is a fault. Instead of writing “00000000” it has written “10101010”. So while read operation FSM read the data byte and compares it with “00000000”. There is a mismatch says that the chip is faulty one, which is indicated as Faulty

V. SIMULATION RESULTS

Verilog HDL Design of Testable SRAM is done with Xilinx ISE Simulator. The design is simulated with the same tool ISE simulator. Simulation results are shown below. For different levels of the algorithm the simulated results we can observe here. Figure 7.1 shows the simulation results when the FSM is in (R0, W1, R1) stage and figure 7.2 shows the simulation results when the FSM is in (R0) stage. After this stage “finish” goes high indicating testing process completed. After this stage error signal is “low” indicating it is error free chip.

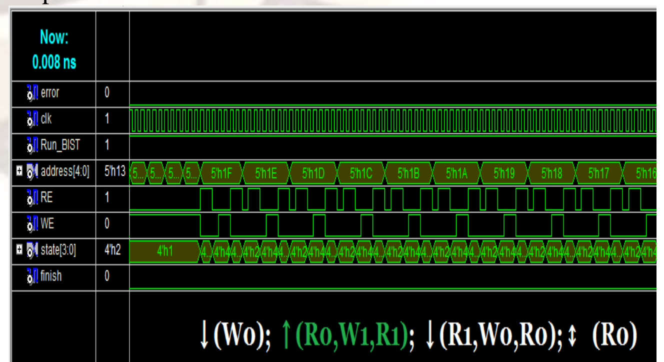

Figure 8: Simulation results when the FSM is in (R0, W1, R1) stage.

Above is the simulation result showing that the FSM is in (R0, W1, R1) stage. After writing “00000000” in all the locations of this SRAM chip,

Now the FSM is in (R0, W1, R1) stage. In this stage 'error' output is '0' indicating up to this stage chip working properly and 'finish' output signal '0' indicates the FSM still checking further.

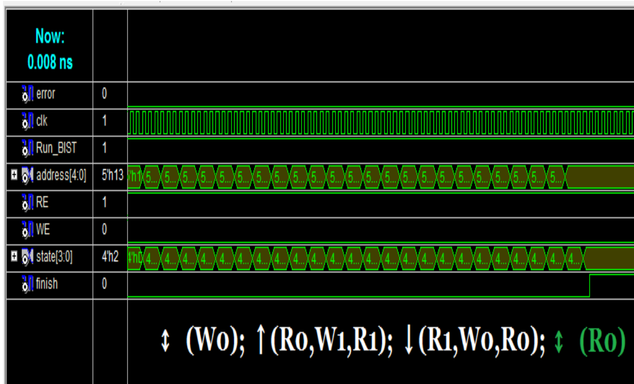

Figure 9: Simulation results when the FSM is in (R0) stage.

Above is the simulation result showing that the FSM is in (R0) stage. After (R1, W0, R0) stage, Now the FSM is in (R0) stage. In this stage 'error' output is '0' indicating up to this stage chip working properly. After completion of this stage 'finish' output signal '1' indicates the FSM finished checking.

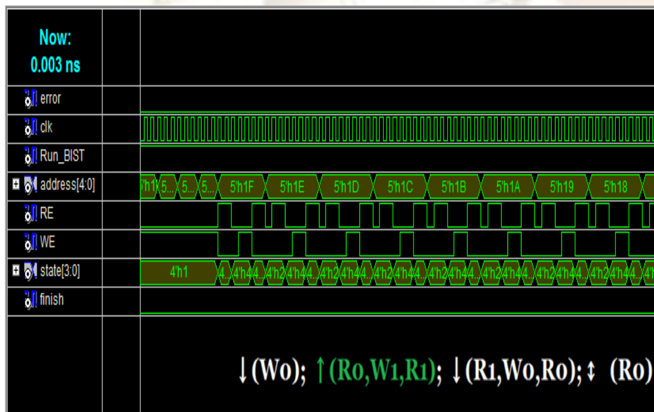

Figure 10: Simulation results when the FSM is in (R0, W1, R1) stage for a faulty chip

Above is the simulation result showing that the FSM is in (R0, W1, R1) stage. After writing "00000000" in all the locations of this SRAM chip, Now the FSM is in (R0, W1, R1) stage. In this stage 'error' output is '0' indicating up to this stage chip working properly and 'finish' output signal '0' indicates the FSM still checking further.

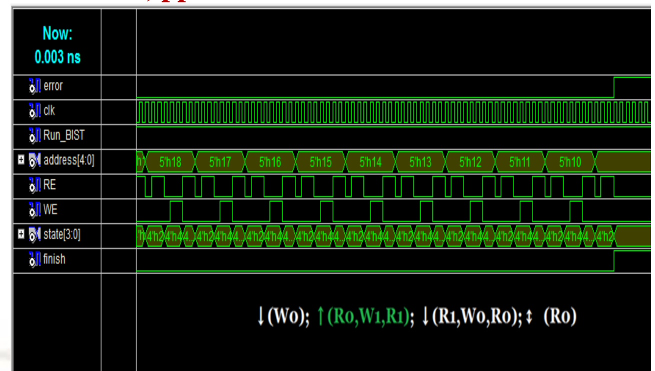

Figure 11: Simulation results when the FSM is in (R0, W1, R1) stage for a faulty chip.

Above is the simulation result showing that the FSM is in (R0, W1, R1) stage. After writing "00000000" in all the locations of this SRAM chip, Now the FSM is in (R0, W1, R1) stage. In this stage 'error' output is '1' indicating up to this stage chip working properly. Now after reading from the address "00001111" that data byte is compared with "00000000" and it does not matched with "00000000". So 'error' signal goes logic '1' indicating it a faulty chip. The 'finish' output signal also goes logic '1' level indicating test completed.

VI. CONCLUSION

This algorithm is quite fast and can provide excellent solutions in short run-times. My experimental results indicate that this flow can save the designer many days of work by offering good BIST architectures which are complete in terms of logical and physical attributes. BIST designed here using March algorithm tests faster than the previously proposed ones and nearly 100% fault coverage .Compared with currently known microcode-based BIST techniques, my design requires only one third of those microcode storages and the testing time is reduced fifty percent. It is strongly believed that this BIST can be widely used for the embedded memory testing especially under the SoC design environment due to the superior flexibility and extendibility in applying different combination of memory test algorithms. Future work will consider optimization algorithms other than the March algorithm used in this work to obtain better solutions.

REFERENCES

[1] S. W. Golomb, Shift Register Sequence, Aegean Park Press, Laguna Hills, CA, 1982.

[2] F. Brglez and H. Fujiwara, A neutral netlistof 10 combinational benchmark circuits and a target translator in Fortran, in Proc. Int. Symp. on Circuits and Systems, pp. 663–698, June 1985.

[3] P. H. Bardell, W. H. McAnney, and J. Savir, Built-In Test for VLSI:

Pseudorandom Techniques, John Wiley & Sons, Somerset, NJ, 1987.

[4] R.-M. Chou, K. K. Saluja, and V. D. Agrawal, Power constraint scheduling of tests, in Proc. Int. Conf. on VLSI Design, pp. 271–274, January 1994.

[5] L.-T. Wang, C.-W. Wu, and X. Wen, editors, VLSI Test Principles and Architectures: Design for Testability, Morgan Kaufmann, San Francisco, 2006.

[6] F. Corno, M. Rebaudengo, M. SonzaReorda, G. Squillero, and M. Violente, Low power BIST via non-linear hybrid cellular automata, in Proc. VLSI Test Symp., pp. 29–34, May 2000.

[7] P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, A test vector inhibiting technique for low energy BIST design, in Proc. VLSI Test Symp., pp. 407–412, April 1999.

[8] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, J. Figueras, S. Manich, P. Teixeira, and M. Santos, Low energy BIST design: Impact of the LFSR TPG parameters on the weighted switching activity, in Proc. Int. Symp.on Circuits and Systems, CD-ROM Proceedings, June 1999.

[9] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, and H. J. Wunderlich, A modified clock scheme for a low power BIST test pattern generator, in Proc. VLSI Test Symp., pp. 306–311, May 2001.

[10] D. Gizopoulos, N. Kranitis, A. Paschalis, M. Psarakis, and Y. Zorian, “Low Power/Energy BIST Scheme for Datapaths,” in Proc. VLSI Test Symp., pp. 23–28, May 2000.

[11] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design, IEEE Press, Revised Printing, Piscataway, NJ, 1994.

[12] E. J. McCluskey, Logic Design Principles: With Emphasis on Testable Semicustom Circuits, Prentice-Hall, Englewood Cliffs, NJ, 1986.

AUTHORS

D. Vishwa Bharathi: Asst. Professor in Pragati Engineering College. Has Two years of teaching experience. Major working areas are Digital electronics, Image processing and VLSI.

K. Raghuram: Associate Professor in Pragati Engineering College. Has Five years of teaching experience. Major working areas are Embedded systems, Image processing and VLSI.

G. Rajesh Kumar: Asst. Professor in Vishnu institute of technology. Has five years of teaching experience. Major working areas are Digital electronics, Embedded Systems and VLSI. Presented research papers in four international conferences and two national conferences. Published Research papers in Four international journals.