Pipelined programmable MBIST design

Hamsa J[1], Dr Siva Yellampalli[2]

[1]PG student, VTU Extn. Centre, UTL Technologies Ltd, Bengaluru
[2]Principal, VTU Extn. Centre, UTL Technologies Ltd, Bengaluru

---***---

Abstract - *With rapid changes in technology ,it is becoming extremely difficult to predict defect types that could manifest during the manufacturing process. This paper proposes one such methodology to support at speed memory testing by implementing instruction pipelining in the design. The design is programmable to support any linear test algorithm and also the march 2 algorithm that includes varying data pattern.*

***Key Words*:** **PMBIST, march algorithms, instruction pipelining, SRAM**

1.INTRODUCTION

With the advent of deep sub micron technology the memory density on the chip are increasing and this development demands all the test algorithms to be included while testing. In memories at particular technology node, we may not get all the kinds of faults, but we may have to include all the algorithms to avoid any risk.

Programmable BIST solution would allow certain degree of flexibility to modify test programs at run time. With programmable MBIST, we can fine-tune the algorithms to something very specific for the technology node and memory design. This level of flexibility allows designers to adapt to the problem at hand. To achieve at speed testing pipelining is the most efficient approach. Hard wired methods of MBIST are not flexible, they are operated for the same algorithms and even requires larger instruction memory. Pipelined PMBIST is capable of supporting wide range of linear test algorithm and the user defined algorithm that can detect wide variety of memory faults.

Basically there are three different MBIST designs and are developed further with improved methodologies namely microcode based MBIST, FSM based MBIST and Counter based MBIST[1]. In each of these designs the only difference is the way they can be utilized to store the march elements. This paper presents pipelined PMBIST design developed from microcode based MBIST design and the proposed design is programmable at run time instead of being hardwired. The design is built by efficient use of three sage instruction pipelining.

2. MICROCODE BASED MBIST

The microcode based MBIST uses microcode to represent March test algorithms. In microcode based controller test patterns are written as set of instructions and loaded into memory BIST controller. Fig 1 shows a block diagram of the Microcode based MBIST controller.

The microcode is a binary code that consists of a fixed number of bits, each bit specifying a particular data or operation value. The microcode instructions are designed by the user, depending on the test pattern algorithm to be used and no standard rule is followed. According to the microcode, address generation logic and data generation logic will work. Test data is applied to memory under test. The output from memory under test is given to comparator. It is used to compare the test inputs to memory and outputs from memory. This result is used to determine whether the memory is faulty or not. This design is usually hardwired and all the required algorithms are stored in the in the microcode memory this would rather increase the size of microcode memory when number of algorithms increases.

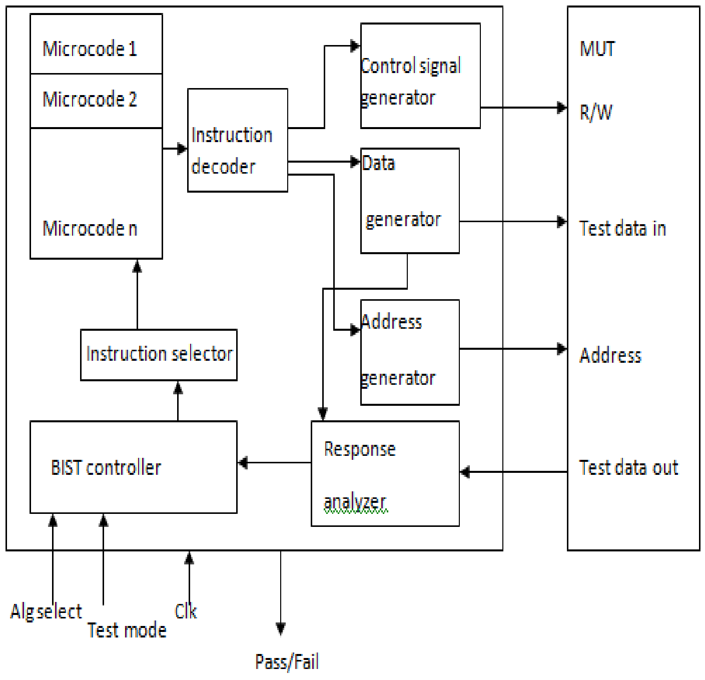

Fig -1: Block diagram of microcode based MBIST

3. PROPOSED PMBIST DESIGN

MUT. Output response analyzer (ORA) is a comparator circuitry that is enabled only when read operations are performed on MUT , this circuit compares the data written to the memory and data read from the memory to say that weather both are same if not indicates the particular memory location is faulty.

The design is intend to work on the principle of instruction pipeline. Instruction pipeline is a technique used in modern processors and controllers to increase the instruction throughput (the number of instructions that can be executed in unit time). By breaking the logic into smaller pieces and inserting flip flops between pieces of logic, the time required by the logic (to decode values till generating valid outputs depending on these values) is reduced. In this way the clock period can be reduced. The inclusion of pipeline registers in MBIST is done at the RTL level.

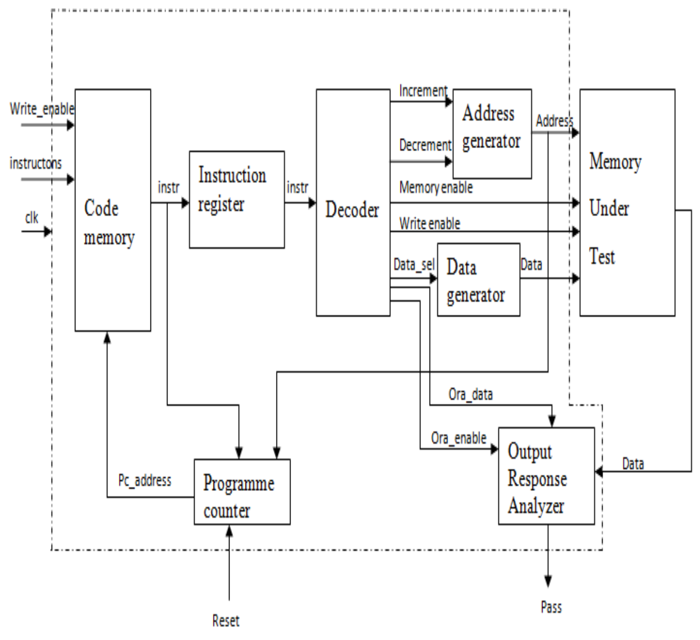

Fig -2: Block diagram of proposed PMBIST

The code memory is designed to store the instructions of 6 bit each. Design includes three types of instructions , algorithmic instruction to generate control signals ,address instructions to monitor the addressing order and jump instructions to access code memory

Table -1: Description of bits in algorithmic instructions

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Instruction type	Instruction type	Data	Data	Write/read	Enable

Bit 1 is used to enable the memory chip, bit2 represents read operation if it is 0 else write operation ,bits 3 and 4 represents the type of data as described in table 2 and the bits 5 and 6 are used to represent type of instruction as described in table 3.

Table -2: Description of data bits

Bit 4	Bit 3	Type of data
0	O	16'b0000000000000000
0	1	16'b0101010101010101
1	0	16'b1010101010101010
1	1	16'b1111111111111111

Table -3: Description of bits 5 and 6

Bit 4	Bit 3	Type of instruction
0	O	Algorithmic
0	1	Address
1	0	Jump
1	1	Invalid

Description of address instructions are shown in table 4 , these instructions are used to instruct address generator to count the address in either upward direction or in downward direction.

Table -4: Description of bits in address instructions

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Instruction type	Instruction type	Not used	Not used	Up/down	Enable

In the address instructions bit 2 instructs address generator to act as up counter if the bit is 1 else as down counter.

Jump instructions are described in table 5 these instructions are used to instruct the programme counter to load the next instruction to instruction register from the code memory.

Table -5: Description of bits in jump instructions

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Instruction type	Instruction type	Not used	Jump value	Jump value	Enable

Bits 3 and 2 are used to represent the jump back value using this information pc loads the next instruction. Each march element is encoded using these instruction types. Encoded instructions are stored in code memory in the order they need to be executed. As a first step algorithmic instructions are executed to generate control signals and then address instruction is executed to generate the address value upon which data operation has to be done. When a specified march element is applied to a memory location jump instructions are used to jump back to same algorithmic instruction in the code memory to apply it on next memory location. This routine continues for all the memory location.

3.2 Instruction pipelining

In the proposed design three stage pipelining is used as shown in fig 3. The first stage is to load the data from code memory to instruction register , the second stage decode is to generate control signals to generate required address, data , read/write signal and jump value. The third stage executes the specified memory operation and results are obtained.

To read the instructions from code memory it takes two clock cycles hence a instruction register is placed in between the code memory and decoder and second clock cycle is used to decode the instruction without any timing violations and the results are obtained in third clock cycle. Further these stages are overlapped to effectively utilize the available time.

Load Instr 1	Decode Instr 1	Execute Instr 1		
	Load Instr 2	Decode Instr 2	Execute Instr 2	
		Load Instr 3	Decode Instr 3	Execute Instr 3

Fig -3: Three stage instruction pipelining

3. IMPLEMETATION AND RESULTS

The proposed Pipelined PMBIST architecture was designed using verilog for MATS , MARCH C and MARCH 2 algorithms and performed the functional verification using cadence ncsim simulator. The type of MUT used was SRAM of size 256X16 bits. The MUT was made faulty by making some of the cells permanently 0 and some cells permanently 1. This is equivalent to stuck at o fault and stuck at 1 fault respectively. Then the march algorithms are applied to the MUT. This produces an output indicating that a fault was introduced in the memory cell array. A signal fail is used to denote the test results after testing. If the fail 0 this means MUT is faulty.

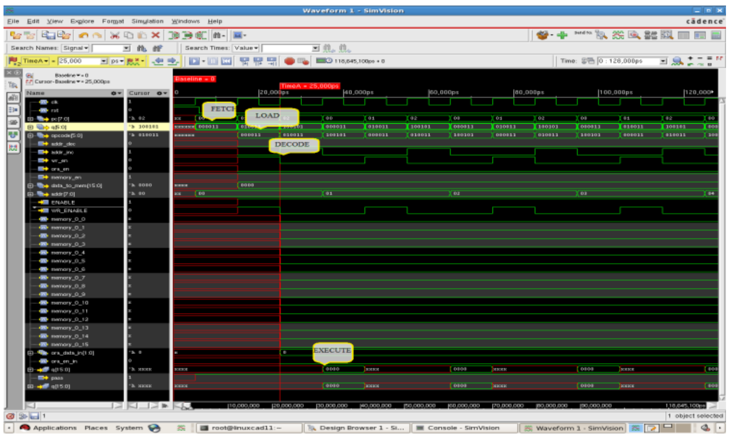

Fig -3: Simulation showing three stages of pipeline

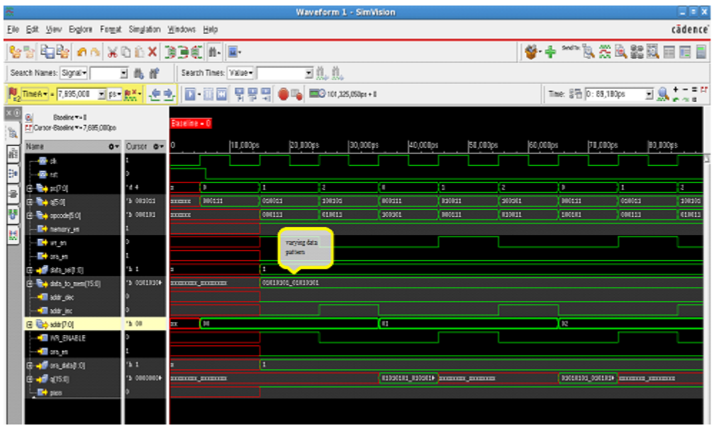

Fig -4: Simulation result of fault free test for MRCH 2 with varying data background

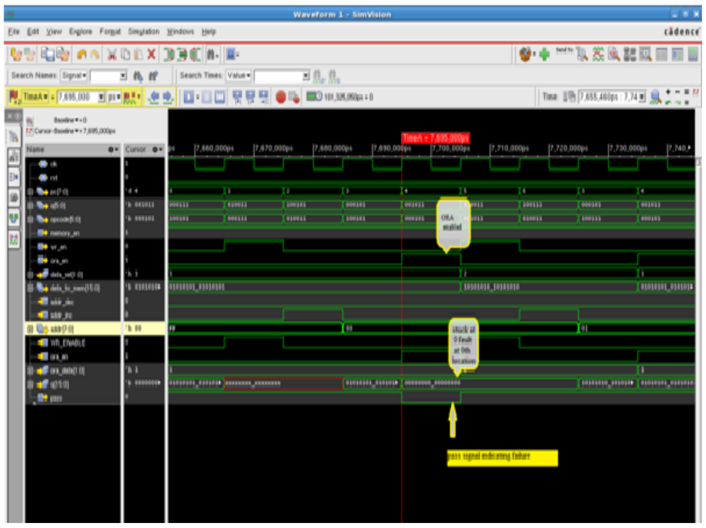

Fig.5 simulation result for march 2 with varying data after fault insertion

5. CONCLUSIONS

The proposed Pipelined PMBIST architecture was designed and verified with MUT that has a access time of two clock cycles and it design has been verified to meet all the timing requirements to generate the results within two clock cycles. The design can be programmed to any linear test algorithms to support high fault coverage with different fault models. The design was verified for single port SRAM , a future work can be on developing the design for other type of memories like DRAM with its specific memory faults.

6. REFRENCES

[1] Elsa Joseph P , Rony Antony P, "VLSI design and Comparative Analysis of Memory BIST controllers". International Conference on Computational Systems and Communications(ICCSC), December 2014.

[2] Nor Azura Zakaria, "Architecture of Pipelined MBISTs and Its

Configuration in Complex SoC", IEEE Symposium on Industrial Electronics & Applications (ISIEA2013).

[3]Xiaogang Du , "Full-speed Field-Programmable Memory BIST Architecture",IEEE ITC, 2005.

[4] AD J.van de goor , "Testing Semiconductor memories : theory and practice" ,1999, ISBN 90-80 4276-1-6 . [5] Muddapu Parvathi , N. Vasantha, K Sarthya Prasad ,"Modified Macrch C –Algorithm for Embedded Memory Testing", IJECE ,Vol.2, No.5, october 2012,pp.571~576.]