International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 11 Issue 6 Series I || June 2022 || PP 23-31

VERILOG Implementation of ALU Based Address Generation for Memory

V.Seetharama Rao[1] B. Nitish Rao[2] C. Sai Bharath Rao[3]
K. Aditya Hrudai[4]
[234]UG Scholar Sreenidhi Institute of Science and Technology, Hyderabad.
[1]Assistant Professor Sreenidhi Institute of Science and Technology, Hyderabad.

Abstract: *Memory Built-In Self-Test has become a standard industrial practice. Its quality is mainly determined by its fault detection capability in combination with its required area overhead. Address Generators have a significant contribution to the area overhead. For generating the address for Radom Accesses Memory (RAM) it requires extra hardware. If an Arithmetic and Logic Unit (ALU) uses for generating the address for RAM it reduces the hardware overhead. ALU based address generation are used in many applications like Smartcard Chips, Digital Signal Processors and Microcontrollers etc. This technique is designed by using Verilog HDL. Simulation is done to verify the functionality and synthesis is done to get the Netlist. The code is simulated and synthesized using Xilinx ISE 13.2 and the designed is implemented in SPARTAN 3E FPGA board.*

Keywords: *Memory BIST, Address Generation, ALU-based implementation*

Date of Submission: 08-06-2022 Date of Acceptance: 16-06-2022

I. INTRODUCTION

Motivation In computing, an Arithmetic and Logic Unit (ALU) is a digital circuit that performs arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit of a computer, and ev enthe simplest microprocessors contain one for purposes such as maintaining timers. The processors found inside modern CPUs and Graphics Processing Units (GPUs) accommodate very powerful and very complex ALUs a single component may contain a number of ALUs. In some computer processors, the ALU is divided into two distinct parts, the Arithmetic Unit and the Logic Unit. The Arithmetic Unit performs the arithmetic operations and the Logic Unit performs the logical operations. The Arithmetic operations include the addition, subtraction, multiplication and division where as the logical operations include the logical AND, OR, XOR, NOT, NAND, NOR, these operations are bit-wise operations. In this thesis, the design of the address for memory device using ALU that is already in the processor is discussed. In doing so, the reduction of overhead area and complexity is expected to be achieved. Built-In Self-Test (BIST) is a Design-for-Testability (DFT) technique, because it makes the electrical testing of a chip easier, faster, more efficient and less costly. The concept of BIST is applicable to just about any kind of circuit, so its implementation can vary as widely as the product diversity that it caters to. As an example, a common BIST approach for Random Access Memory (RAM) includes the incorporation onto the chip of additional circuits for pattern generation, timing, mode selection, and go-/no-go diagnostic tests. The main drivers for the widespread development of BIST techniques are the fast-rising costs of Automatic Test Equipment ATE testing and the growing complexity of integrated circuits. It is now common to see complex devices that have functionally diverse blocks built on different technologies inside them. Such complex devices require high-end mixed-signal testers that possess special digital and analog testing capabilities. BIST can be used to perform these special tests with additional on-chip test circuits, eliminating the need to acquire such high-end testers. BIST is also the solution to the testing of critical circuits that have no direct connections to external pins, such as embedded memories used internally by the devices. In this thesis, for the memory testing MIPS processor is preferred. MIPS processor architecture is single cycle processor. This thesis describes the implementation of the MIPSs processor and in place of ALU and DATA memory module the memory BIST module is replaced.

II. Address Generation and Test Generation Methods:

Built-In Self-Test (BIST) has become a standard industrial practice for testing memories since memory cores constitute a major part of the die area. One of the key components of memory BIST is the address generator (AG). In order to detect speed-related faults, the address generator has to generate different address sequences to allow for appropriate address transitions. Its complexity is a major design issue, since it requires large area and limits the BIST speed. For memory testing, the address generator has to generate different

Counting Methods (CMs) since each counting method has its own detection. The main purpose of the address generator is to generate the test patterns. Such address generators are Linear Feedback Shift Registers (LFSRs), Pipeline LFSR, Binary Up down Counter and Gray Code up down counter. This chapter describes the different types test generation methods and testing using Automatic Test Pattern Generation (ATPG) and Built-In-Self-Test (BIST).

III. Address Generators:

Memory BIST alleviates long test times by incorporating a self-testing logic within the memory itself. Memory Built-In-Self-Test (MBIST) tends to be of much simpler construction, where Address Generator read /write controller, Address counter and, Data generator are the main components of the memory BIST. The main purpose of the address generator is to generate the test patterns.

Linear Feedback Shift Register

Linear Feedback Shift Register (LFSR) is kind of data register in which a clock is used to control the data patterns. The LFSR is characterized by a feedback loop to determine the contents of the LFSR during a state transition. LFSR's behavior is in the form of particular the sequence of symbols that is stored after each clock cycle and modeled mathematically. LFSR module is configured according to characteristic polynomial for generating an output stream of test patterns according to an input stream. Each LFSR respectively receives a sub-input stream and at least one feedback stream and respectively generates a sub-output stream and a feedback stream according to the received sub-input stream, wherein the sub-input stream is generated according to the input stream. At least one of the received feedback streams is generated by another LFSR. The output generator generates the output stream according to inputs polynomials. The standard form of LFSR is shown in Figure 2.1.

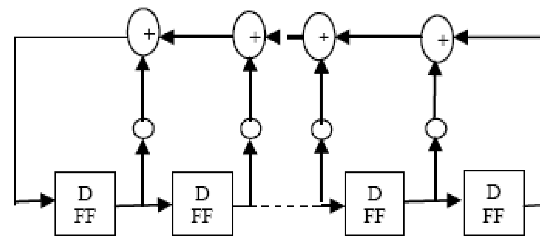


Figure: Simple LFSR

Pipeline LFSRLFSR allows counting at maximum device speed for practically unlimited bit width of the counter. The cycle length for the LFSR with *n* bits is 2*n* – 1. Sometimes, other cycle length may be desired, for that desired purpose, LFSR connections can be easily modified to achieve different cycle length, but at the price higher register-to-register delay and the counter speed goes down. As LFSR speed is the one of the important factor in VLSI testing, so pipelining technique can be used to reduce the register-to- register delay in the LFSR. In case of LFSR, all output signals are created by the shift register. If this output signal is used as feedback signal, it is equivalent to use of the output value with the index by one lower and with register added into its path as shown in the Figure 2.2.

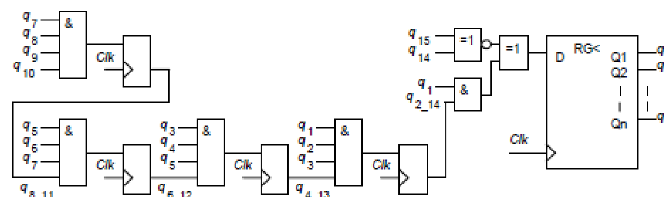


Figure: Pipeline LFSR

Binary Up or Down Counter:

Binary up down counter is designed in VLSI. It consists of the jk flip-flop. Binary up counter is designed to count the memory location in the memory. It counts up by one only wt the positive level edge of the clock or when any event occurs on the cloak. But when the down signal is high and up signal goes low it count by one to downward. Finite state machine is used to design this counter as shown in Figure 2.3.

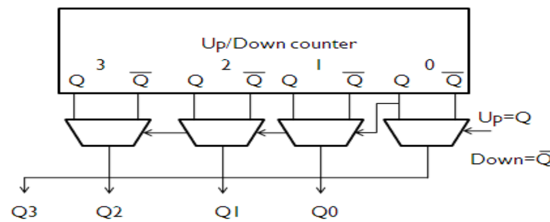


Figure: Up or Down counter.

Gray Code Up Or Down Counter:

Gray counters are based on the gray codes that are the unweighted code that do not have any specific weight assigned to them. A typical use of Gray code counters is building a FIFO (First-In First-Out) data buffer that has read and write ports that exist in different clock domains. The input and output counters inside such a dual-port FIFO are often stored using Gray code to prevent invalid transient states from being captured when the count crosses clock domains The thing which forces the implementation of gray counter is the power utilization. Because in case of the simple binary up down counter when there is step of counting from “01111111” then next level is “10000000”, which conveys that there is the 4-bit change which consumes more power whereas gray counter is of the only single bit change. The gray counter is beneficial for address up down counting in low power applications due to reduced toggling from one stage to next stage. Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems. The advantage of Gray codes in these applications is that differences in the propagation delays of the many wires that represent the bits of the code cannot cause the received value to go through states that are out of the Gray code sequence. Gray code up or down counter is shown in Figure 2.4.

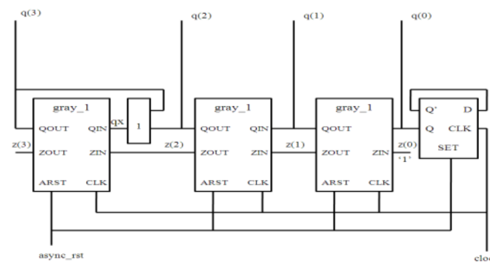


Figure: Gray code up or down counter

IV. Test Generation Methods

Automatic Test Pattern Generation (ATPG) pattern-data to systematically exercise as many logic-gates, and other components, as possible. Built-in Self-Test, or BIST installs self-contained test-controllers to automatically test a logic structure in the design.

Automatic Test Pattern Generation (ATPG)

ATPG stands for Automatic Test Pattern Generation. As the name suggests, ATPG procedure involves generation of input patterns that can ascertain presence or absence of fault(s) at some location(s) in a circuit. There are several ways of ATPG. In the last module two ways have been discussed. (i) fault simulation and (ii) sensitization propagation justification. These techniques are based on Boolean logic manipulations and are most widely used. However, it must not be felt that only logic based schemes can be used for ATPG. ATPG (acronym for both Automatic Test Pattern Generation and Automatic Test Pattern Generator) is an electronic design automation method/technology used to find an input (or test) sequence that, when applied to a digital circuit, enables testers to distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects. The generated patterns are used to test semiconductor devices after manufacture, and in some cases to assist with determining the cause of failure. The effectiveness of ATPG is measured by the amount of modeled defects, or fault models, that are detected and the number of generated patterns. These metrics generally indicate test quality (higher with more fault detections) and test application time (higher with more patterns). ATPG efficiency is another important consideration. It is influenced by the fault model under consideration, the type of circuit under test (full scan, synchronous sequential, or asynchronous sequential), the level of abstraction used to represent the circuit under test (gate, register-transistor, switch), and the required test quality.

Built In Self Test : The main drivers for the widespread development of BIST techniques are the fast-rising costs of ATE testing and the growing complexity of integrated circuits. It is now common to see complex devices that have functionally diverse blocks built on different technologies inside them. Such complex devices

require high-end mixed-signal testers that possess special digital and analog testing capabilities. BIST can be used to perform these special tests with additional on-chip test circuits, eliminating the need to acquire such high-end testers. Built-in Self Test is the technique of designing additional hardware and software features into integrated circuits to allow them to perform self-testing, i.e., testing of their own operation (functionally, parametrically, or both) using their own circuits, thereby reducing dependence on an external automated test equipment. BIST is a Design-For-Testability (DFT) technique, because it makes the electrical testing of a chip easier, faster, more efficient, and less costly. The concept of BIST is applicable to just about any kind of circuit, so its implementation can vary as widely as the product diversity that it caters to. As an example, a common BIST approach for Dynamic Random Access Memory (DRAM) includes the incorporation onto the chip of additional circuits for pattern generation, timing, mode selection, and go-/no-go diagnostic tests. The main drivers for the widespread development of BIST techniques are the fast-rising costs of Automatic Test Equipment (ATE) testing and the growing complexity of integrated circuits. It is now common to see complex devices that have functionally diverse blocks built on different technologies inside them. Such complex devices require high-end mixed-signal testers that possess special digital and analog testing capabilities. BIST can be used to perform these special tests with additional on-chip test circuits, eliminating the need to acquire such high-end testers. BIST is also the solution to the testing of critical circuits that have no direct connections to external pins, such as embedded memories used internally by the devices. In the near future, even the most advanced tester may no longer be adequate for the fastest chip, a situation wherein self-testing may be the best solution

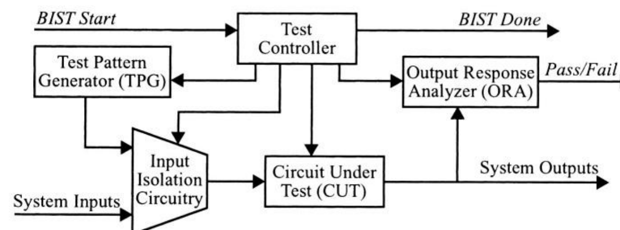


Figure: BIST Block Diagram

BIST Controller Unit (BCU) It controls the test execution, it manages the TPG,Output Response Analyser (ORA) and the multiplexer.

Test Pattern Generator (TPG) It generates the test pattern for the Circuit Under Test (CUT). The patterns may be generated in pseudorandom. 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 Automatic Test Pattern Generation (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 2^n, if there are n flip-flops in the register) as possible.

Input Mux This multiplexer is to allow normal inputs to the circuit when it is operational and test inputs from the pattern generator when BIST is executed. The control input of the multiplexer is fed by a central test controller.

Test Controller Circuit to control the BIST. Whenever an IC is powered up (signal start BIST is made active) the test controller starts the BIST procedure. Once the test is over, the status line is made high if fault is found. Following that, the controller connects normal inputs to the CUT via the multiplexer, thus making it ready for operation. Circuit Under Test (CUT)It is the portion of the circuit tested in BIST mode. It can be sequential, combinational or a memory. Output Response Analysis (ORA) It analyses the CUT output with the expected output if both are same error signal is low if not error signal is high. Output response compacter performs lossy compression of the outputs of the CUT. As in the case of off-line testing, in BIST the output of the CUT is to be compared with the expected response (called golden signature). If CUT output does not match the expected response, fault is detected. Similar to the situation for test pattern generator, expected output responses cannot be stored explicitly in a memory and compared with the responses of the CUT. So CUT response needs to be compacted such that comparisons with expected responses (golden signatures) become simpler in terms of area of the memory that stores the golden signatures.

Two common classification of BIST are the Logic BIST (LBIST) and the Memory BIST (MBIST). LBIST, which is designed for testing random logic, typically employs a Pseudo-Random Pattern Generator

(PRPG) to generate input patterns that are applied to the device's internal scan chain, and a Multiple Input Signature Register (MISR) for obtaining the response of the device to these test input patterns. An incorrect MISR output indicates a defect in the device.
ALU Based Memory Test Block Diagram

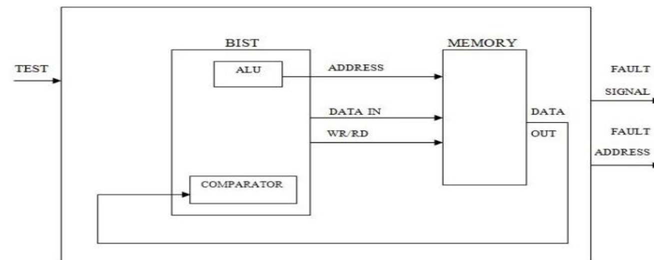


Figure: ALU Based Memory Test Block Diagram

The above figure explains the complete operation of the project. The BIST contain Test pattern generator as well as the Output response analyzer. As described in earlier chapters, the ALU from the processor is required to generate the linear addresses for the data to be stored in the memory hence the TPG is shown as the ALU module. The working of the ALU has been explained in the previous chapter. The output response analyzer is the comparator as shown above. The BIST module provides the Address, Data and the control signals (write or read) to memory that is being tested.

The Memory block has three inputs i.e. Address, Data in and control signals. The Address is given by the ALU. The address that is generated is a Linear Address. The control signal is active high, when the control signal is high it performs writ operation and when it goes low it performs the read operation. During write operation the data to be stored is sent to the memory from the Data in and the location where it has to be stored is specified by the Address line. During read operation the location from which the data has to be read is specified in the Address line. The Data is read out from the Data out. The Test signal is applied to the main block when Test is low the ALU performs normal operation and when Test is high then the ALU provides the Address to the Memory. The Data out line is connected to the Comparator in the BIST controller. The Comparator compares the data that is read. If the data doesn’t match then the fault signal goes high and fault address line gives out the Faulty address location in the memory ALU is present in every architecture, based on that implementation of the processor is done. In control unit the data path signals are enabled. In the place of ALU and DATA memory module, the MEMORY BIST module is replaced. The proposed design is designed and implemented by using FPGA board with the Xilinx ISE tools as mentioned in table 4.1.

Table list of tools used for ALU based address generation for memory.

Design Action	Tool Name
Design Entry	Verilog HDL
Synthesis	Xilinx Synthesis Tool(XST)
Implementation	FPGA Editor, Plan Ahead

MIPS processor is designed and implemented whose architecture is shown in Figure

V. SIMULATION RESULTS:

The design of ALU based address generation for memory implemented by tacking processor. The processor contains instruction memory or program memory which stores the instructions, data memory, register file stores the and finally ALU block. Here individual block of
Simulation Results of Top Module Processor
ALU is present in every architecture, based on that we implement the processor. In control unit, the data path signals are enabled. In the place of ALU and DATA memory module we, the MEMORY BIST module is replaced as shown in Figure 5.1.

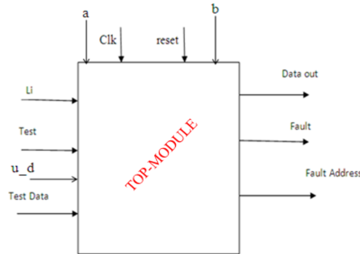


Figure : Top Module Block Diagram

Processor operated with frequency signal 124MHz and it consisted input signals two operands a, b and clk, reset, li, test u_d, test data and outputs are Data out, fault and finally fault address based on test and li inputs it generates the related ouputs, when input for linear address and address compliment address generation control pin Li=0 address com Li=1 linear address are generated. Test input for memory testing enable pin Test=1 work the memory Test=0 normal processor working. Test data 8 bit data input for writing data to memory. Output processor output if fault =1 it represents fault if Fault=0 it represents no fault and finally fault address output 8-bit fault address. When inputs test=1 and li=0 then simulation results as shown in Figure 5.2.

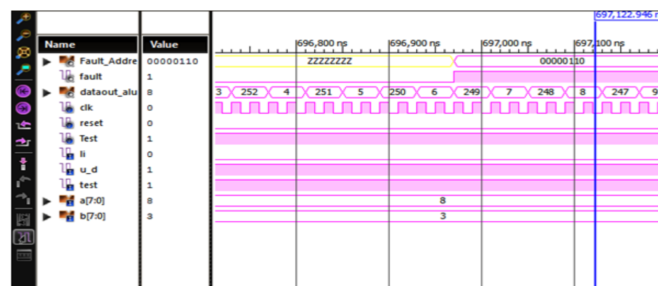


Figure Top Module Simulation Results

When Test=1Here, it compares expected data with circuit out or present output internally if doesn't match with each other it shows fault high and fault address as shown in above simulations results fault 1 and fault address 00000110. When test input=0 then simulation results as shown in Figure 5.3.

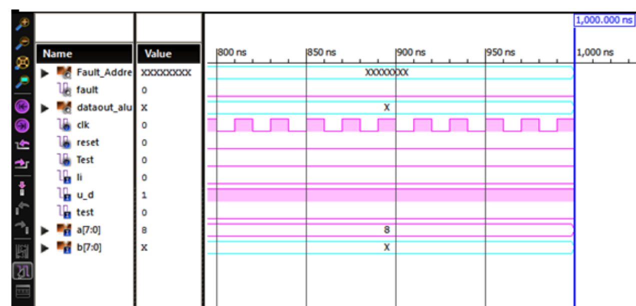


Figure 5.3: Top Module Simulation Results Test=0Here input b is undefined because b is taken from ALU updated result value here ALU not taken that operand. So it shows the circuit output is undefined by default fault output and fault address outputs also undefined. By given the operands a and b externally with test input = 0 then top module act as ALU simulation results as shown in Figure 5.4.

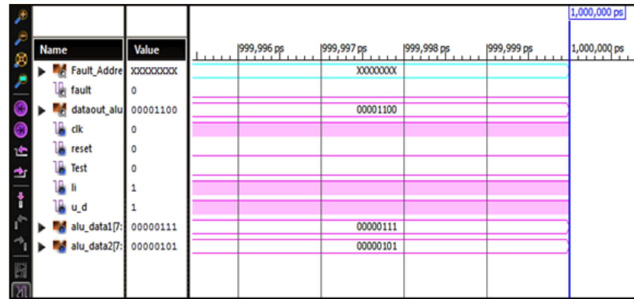


Figure Top Module ALU Simulation Results

ALU BLOCK

ALU Block consisted the inputs are a, b, li, test, clk, rest and u_d and out puts are aluop and dout .based on inputs it perform the operation. Whenever test is high it generates the address otherwise it performs the normal operation. Based on li input it generates the address linear or complement address in normal based on aluop input it perform the arithmetic operations. Here we observe different address generations and alu normal operations. The inputs to the ALU are the data to be operated on (called operands) and a code from the control unit indicating which operation to perform. Its output is the result of the computation. Most of a processor's operations are performed by one or more ALUs. An ALU loads data from input registers then an external control unit tells the ALU what operation to perform on that data, and then the ALU stores its result into an output register. The control unit is responsible for moving the processed data between these registers, ALU and memory. In below we observe the different operation of ALU with different inputs. whenever li is high it generates the linear address generations and li is low then it generates the linear address simulation results as shown in Figure 5.5.

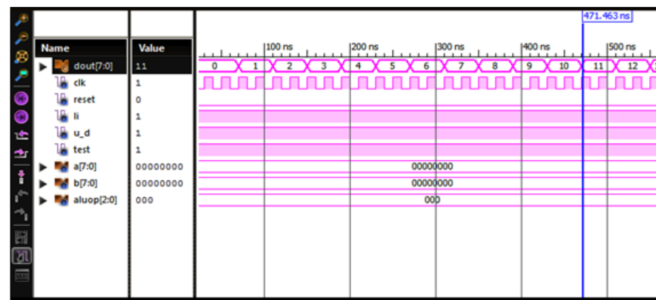


Figure :Linear Address Generation

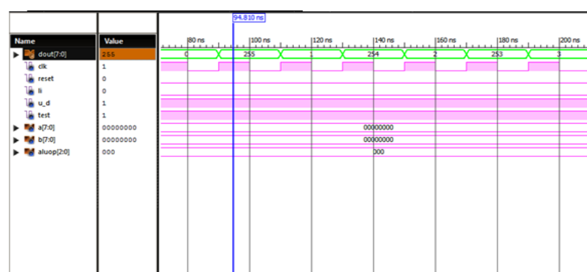


Figure: Linear and Complement Address Generation

Here the ALU normal operation is observed by keeping input as test=0 based on ALU-Operation as shown in Figure

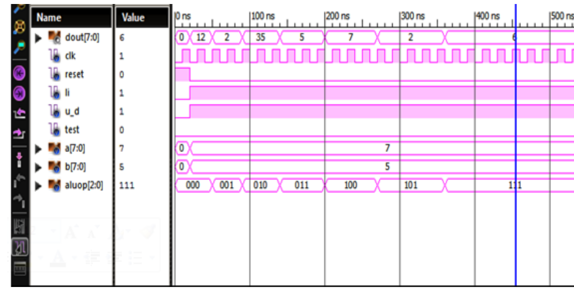


Figure: ALU Normal Operations

Program Memory

In program memory we defined the instruction sets.16 bit instruction format first 6bis are used for Opcode--→-6bits, source resistors and destination register occupies 3bits each RS_addr,RT_addr and Rd_add and four bits for offset as shown in Figure 5.8.

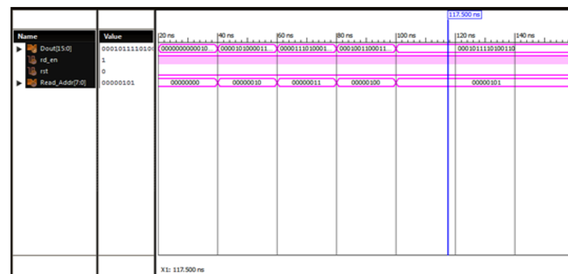


Figure: Simulation Results of Program Memory

Register Block

Based on RS add, RT add and RD add are 3bit instructions it generates the related source operands and results and those results are stored in destination address with related register. Here the related instructions related operands are fetched for register block as shown in Figure 5.9.

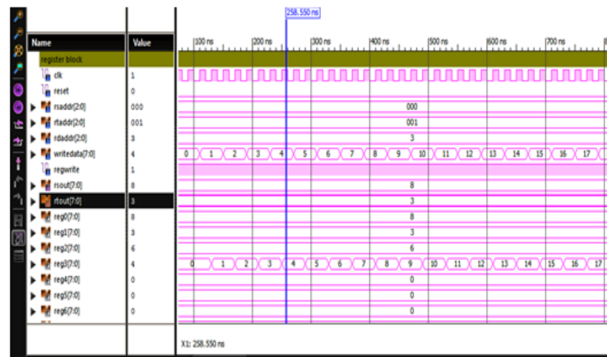


Figure: Simulation Results of Register Block

Data Memory

Here we observe the related address related output as shown in Figure 5.10. Based on wr enable, write the data and read is done.

Figure: Simulation Results of Data Memory Comparator

Here in testing mode when test is one we are comparing expected data and memory(cut) output and detecting faults and generating fault address as shown Figure 5.11.

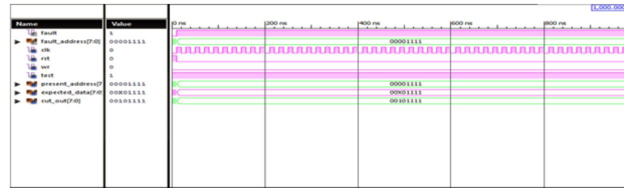
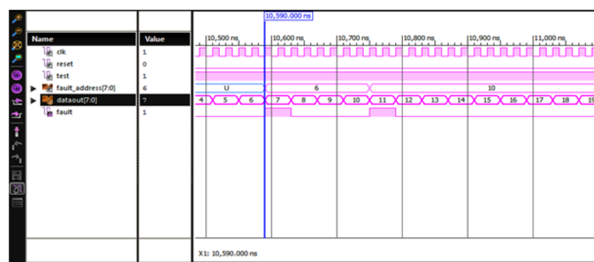


Figure: Simulation Results of Comparators

BIST :

Here, it performs March Algorithm so that all the faults get detected, on comparing actual data with expected data faults get detected and fault address is also generated and ALU Output also generated it means test pattern generations as shown in Figure 5.12.



Simulation Results of BIST

VI. Applications:

Smartcard Chips: Smart cards generally include an integrated circuit (IC) which has a non-volatile memory for storing sensitive information, and also a microprocessor and it supports testing

Digital Signal Processors: High speed digital signal processing (DSP) systems, as used in telecommunications applications, primarily includes a self-testable digital signal processor.

Microcontrollers: IT supports the testing of all system and peripheral RAM

VII. Conclusion:

The ALU based address generation for memory is successfully designed by using Xilinx ISE design suite 13.2 version with Verilog HDL. The design has been simulated for functionality and inputs and output samples are verified by using Xilinx ISE Simulator tool. The proposed one present's lower hardware overhead when the combined power of linear and address complement counting methods are required. The synthesized ALU based address generatin has 6060 LUT slices, 2178 slice registers and 2 bufferrs. Timing analysis show that the critical path is 1.308ns i.e., maximum clock frequency is 124.46MHz. The synthesized design is implemented on Xilinx Spartan3E FPGA.

VIII. Future scope:

The ALU generates linear addresses as well as address complement addresses. In future ALU can be used as other address generators (i.e., gray addresses) and testing data. Gray addresses requires less number of transitions compared with other generators. If it generates gray addresses, the power consumption can be also reduced.

REFERENCES:

[1]. R. Aitken, et. al, 'Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories', Proc. of Int. Test Conference.

[2]. Conroy, et.al, "A practical perspective on reducing ASIC NTFs", Proc. of Int. Test Conference.

[3]. L. Dilillo, et.al, 'Dynamic read destructive fault in embedded-SRAMs: analysis and march test solution', Proceedings Ninth IEEE European Test Symposium.

[4]. X. Du, N. Mukherjee, W.T Cheng and S.M Reddy, 'Full-speed field-programmable memory BIST architecture', Proc. of Int. Test Conference.

[5]. X. Du, N. Mukherjee, W-T Cheng, S. M. Reddy 'A Field-Programmable Memory BIST Architecture Supporting Algorithms and Multiple Nested Loops', Proc. of the Asian Test

[6]. A.J. van de Goor, S. Hamdioui and G. N. Gaydadjiev, 'New Algorithms for Address Decoder Delay Faults and Bit Line Imbalance faults', Proc. of the 18th Asian Test Symposium, pp. 31-36, 2009.[7] H. Kukner, 'Generic and Orthogonal March Element based Memory BIST Engine' Master Thesis, CE-MS-2010-01, Delft University of Technology, September 2010.

[7]. TheNetherlands,1998,Ad.vd.