Simulation Report for Synchronous RAM Design in Verilog

Name: Md. Sameer Ahmed

Internship Domain: VLSI

Task: Design a simple synchronous RAM with read and write operations

**Objective**

The objective of this task is to design a simple synchronous RAM module using Verilog HDL that supports both read and write operations. The design is verified using simulation and waveform analysis.

**Tools Used**

- Language: Verilog HDL

- Simulator: Icarus Verilog

- Waveform Viewer: GTKWave

**Design Description**

The synchronous RAM consists of a clock input, a write enable signal, an address input, a data input, and a data output. When the write enable signal is active, data is written to the specified memory address on the rising edge of the clock. The memory content is available on the data output for reading.

**Synchronous RAM Verilog Code (with Comments)**

module SyncRAM (

input clk, // Clock signal

input we, // Write enable signal (1: write, 0: read)

input [3:0] addr, // 4-bit address input (supports 16 locations)

input [7:0] din, // 8-bit data input (for write)

output reg [7:0] dout // 8-bit data output (for read)

);

// Declare memory array with 16 locations, each 8-bit wide

reg [7:0] mem [15:0];

// Synchronous process (triggers on rising edge of clock)

always @(posedge clk) begin

if (we) begin

// If write enable is high, write data 'din' to memory at 'addr'

mem[addr] <= din;

end

// Read operation: output data from memory at address 'addr'

dout <= mem[addr];

end

endmodule

**Testbench Code (with Comments)**

module tb\_SyncRAM;

reg clk; // Clock signal

reg we; // Write enable

reg [3:0] addr; // Address bus

reg [7:0] din; // Data input bus

wire [7:0] dout; // Data output wire

// Instantiate the RAM module

SyncRAM uut (

.clk(clk),

.we(we),

.addr(addr),

.din(din),

.dout(dout)

);

// Generate clock: toggles every 5 time units (10-unit clock period)

always #5 clk = ~clk;

// Initial block to apply test vectors

initial begin

$display("Time\tWE\tADDR\tDIN\tDOUT");

$monitor("%0t\t%b\t%h\t%h\t%h", $time, we, addr, din, dout);

// Initialize signals

clk = 0; we = 0; addr = 0; din = 0;

// Write 8'hA5 to address 4

#10 we = 1; addr = 4; din = 8'hA5;

// Write 8'h3C to address 7

#10 we = 1; addr = 7; din = 8'h3C;

// Disable write; read from address 4

#10 we = 0; addr = 4;

// Read from address 7

#10 addr = 7;

// Read from an unwritten address (should be 0)

#10 addr = 2;

#10 $finish;

end

endmodule

**Simulation Output**

Time WE ADDR DIN DOUT

10 1 4 A5 xx // Write A5 to addr 4

20 1 7 3C A5 // Write 3C to addr 7

30 0 4 3C A5 // Read from addr 4 → A5

40 0 7 3C 3C // Read from addr 7 → 3C

50 0 2 3C 00 // Read unwritten addr → 0

**Conclusion**

The synchronous RAM module was designed and simulated successfully. The testbench demonstrated correct write and read operations as per the functionality. The RAM design met the expectations of a simple memory unit.