**Synchronous RAM Verilog Code**

**// File: sync\_ram.v**

module sync\_ram #(

parameter DATA\_WIDTH = 8,

parameter ADDR\_WIDTH = 4

)(

input wire clk,

input wire we, // Write enable

input wire [ADDR\_WIDTH-1:0] addr,

input wire [DATA\_WIDTH-1:0] din,

output reg [DATA\_WIDTH-1:0] dout

);

// Memory declaration

reg [DATA\_WIDTH-1:0] mem [0:(1<<ADDR\_WIDTH)-1];

always @(posedge clk) begin

if (we)

mem[addr] <= din; // Write operation

dout <= mem[addr]; // Read operation (synchronous read)

end

endmodule

**Testbench for Synchronous RAM**

**// File: tb\_sync\_ram.v**

`timescale 1ns/1ps

module tb\_sync\_ram;

parameter DATA\_WIDTH = 8;

parameter ADDR\_WIDTH = 4;

reg clk;

reg we;

reg [ADDR\_WIDTH-1:0] addr;

reg [DATA\_WIDTH-1:0] din;

wire [DATA\_WIDTH-1:0] dout;

**// Instantiate the RAM**

sync\_ram #(.DATA\_WIDTH(DATA\_WIDTH), .ADDR\_WIDTH(ADDR\_WIDTH)) ram\_inst (

.clk(clk),

.we(we),

.addr(addr),

.din(din),

.dout(dout)

);

**// Clock generation**

always #5 clk = ~clk;

initial begin

$display("Starting RAM simulation...");

$dumpfile("sync\_ram.vcd"); // For waveform viewing

$dumpvars(0, tb\_sync\_ram);

clk = 0;

we = 0;

addr = 0;

din = 0;

**// Write some data**

#10;

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

we = 1; addr = 5; din = 8'hBB; #10;

**// Read back data**

we = 0; addr = 4; #10;

$display("Read addr 4: %h", dout);

addr = 5; #10;

$display("Read addr 5: %h", dout);

**// Finalize**

#10;

$finish;

end

endmodule

**Expected Simulation Behaviour**

* On write enable (we = 1), data is written to the specified address on rising edge of clock.
* dout always reflects the content of the address on every clock cycle.
* Both read and write are synchronous (registered outputs).