Design Specifications:

* RAM memory with byte-addressable system (1 address can access 1 byte of data)
* It has 1 read and 1 write port
* Write port has signals: wr\_clk(1bit), wr\_en(1bit), wr\_addr(8bit), wr\_data(8bit)
* Read port has signals: rd clk(1bit), rd\_addr(8bit), rd\_data(8bit)

1. Design the memory using Verilog/System Verilog.
2. Create a module test.sv to drive stimulus to design.
3. Create below stimulus for write port: wr\_en=1, wr\_addr=8’h7, wr\_data=8’h11
4. Create below stimulus for read port: wr\_en=0, rd\_addr=8’h7
5. Ensure the data read is same as the data written: rd\_data=8’h11

Design using System Verilog:

module ram (

input logic wr\_clk, // Write clock

input logic wr\_en, // Write enable

input logic [7:0] wr\_addr, // Write address

input logic [7:0] wr\_data, // Write data

input logic rd\_clk, // Read clock

input logic [7:0] rd\_addr, // Read address

output logic [7:0] rd\_data // Read data

);

// Memory storage: 256 bytes, each 1 byte wide (8 bits)

logic [7:0] mem [255:0];

// Write logic (on rising edge of write clock)

always\_ff @(posedge wr\_clk) begin

if (wr\_en) begin

mem[wr\_addr] <= wr\_data; // Write data to the memory location

end

end

// Read logic (on rising edge of read clock)

always\_ff @(posedge rd\_clk) begin

rd\_data <= mem[rd\_addr]; // Read data from the memory location

end

endmodule

Testbench

module test;

// Declare signals for the RAM module

logic wr\_clk;

logic wr\_en;

logic [7:0] wr\_addr;

logic [7:0] wr\_data;

logic rd\_clk;

logic [7:0] rd\_addr;

logic [7:0] rd\_data;

// Instantiate the RAM module

ram ram\_inst (

.wr\_clk(wr\_clk),

.wr\_en(wr\_en),

.wr\_addr(wr\_addr),

.wr\_data(wr\_data),

.rd\_clk(rd\_clk),

.rd\_addr(rd\_addr),

.rd\_data(rd\_data)

);

// Clock generation for write and read clocks

always begin

#5 wr\_clk = ~wr\_clk; // Write clock with a period of 10ns

#5 rd\_clk = ~rd\_clk; // Read clock with a period of 10ns

end

// Test stimulus generation

initial begin

// Initialize signals

wr\_clk = 0;

rd\_clk = 0;

wr\_en = 0;

wr\_addr = 8'b0;

wr\_data = 8'b0;

rd\_addr = 8'b0;

// Apply reset (if needed)

// In this case, no reset is used explicitly

// Write operation: wr\_en = 1, wr\_addr = 8'h7, wr\_data = 8'h11

wr\_en = 1;

wr\_addr = 8'h7;

wr\_data = 8'h11;

#10; // Wait for a clock cycle

// Disable write enable

wr\_en = 0;

// Read operation: rd\_addr = 8'h7

rd\_addr = 8'h7;

#10; // Wait for a clock cycle for read to happen

// Check the result

if (rd\_data == 8'h11) begin

$display("Test Passed: Data read from address 0x7 is 0x%h, as expected.", rd\_data);

end else begin

$display("Test Failed: Data read from address 0x7 is 0x%h, expected 0x11.", rd\_data);

end

// End simulation

$finish;

end

endmodule

Design in Verilog:

module ram (

input wire wr\_clk, // Write clock

input wire wr\_en, // Write enable

input wire [7:0] wr\_addr, // Write address

input wire [7:0] wr\_data, // Write data

input wire rd\_clk, // Read clock

input wire [7:0] rd\_addr, // Read address

output reg [7:0] rd\_data // Read data

);

// Memory storage: 256 bytes, each 1 byte wide (8 bits)

reg [7:0] mem [255:0];

// Write logic (on rising edge of write clock)

always @(posedge wr\_clk) begin

if (wr\_en) begin

mem[wr\_addr] <= wr\_data; // Write data to the memory location

end

end

// Read logic (on rising edge of read clock)

always @(posedge rd\_clk) begin

rd\_data <= mem[rd\_addr]; // Read data from the memory location

end

endmodule

Testbench in SV with random stimulus

module test;

// Declare signals for the RAM module

logic wr\_clk;

logic wr\_en;

logic [7:0] wr\_addr;

logic [7:0] wr\_data;

logic rd\_clk;

logic [7:0] rd\_addr;

logic [7:0] rd\_data;

// Instantiate the RAM module

ram ram\_inst (

.wr\_clk(wr\_clk),

.wr\_en(wr\_en),

.wr\_addr(wr\_addr),

.wr\_data(wr\_data),

.rd\_clk(rd\_clk),

.rd\_addr(rd\_addr),

.rd\_data(rd\_data)

);

// Clock generation for write and read clocks

always begin

#5 wr\_clk = ~wr\_clk; // Write clock with a period of 10ns

#5 rd\_clk = ~rd\_clk; // Read clock with a period of 10ns

end

// Test stimulus generation with random valid inputs

initial begin

// Initialize signals

wr\_clk = 0;

rd\_clk = 0;

wr\_en = 0;

wr\_addr = 8'b0;

wr\_data = 8'b0;

rd\_addr = 8'b0;

// Apply reset (not explicitly required but can be added if needed)

// In this case, no reset is used explicitly

// Randomly write data into a random address

repeat (100) begin

wr\_en = 1;

wr\_addr = $random % 256; // Random address between 0 and 255

wr\_data = $random % 256; // Random data between 0 and 255

#10; // Wait for a clock cycle

end

// Disable write enable after random writes

wr\_en = 0;

// Randomly read data from addresses that were written to

repeat (5) begin

rd\_addr = $random % 256; // Random address between 0 and 255

#10; // Wait for a clock cycle to read data

// Verify that the data read matches the data written to that address

if (rd\_data == wr\_data) begin

$display("Test Passed: Data read from address 0x%h is 0x%h, as expected.", rd\_addr, rd\_data);

end else begin

$display("Test Failed: Data read from address 0x%h is 0x%h, expected 0x%h.", rd\_addr, rd\_data, wr\_data);

end

end

// End simulation

$finish;

end

endmodule