INTERNSHIP TASK -2

- DELIVERABLE: CODE, TESTBENCH, AND SIMULATION DEMONSTRATING RAM FUNCTIONALITY.

1. RAM Module in Verilog

```
// Simple RAM Module in Verilog
module ram (
    input clk,              // Clock signal
    input reset,            // Reset signal
    input [3:0] addr,       // 4-bit address (16 locations)
    input [7:0] din,        // 8-bit data input
    output reg [7:0] dout,  // 8-bit data output
    input write_enable,     // Write enable signal
    input read_enable       // Read enable signal
);

    // Memory: 16 locations, each 8-bits wide
    reg [7:0] mem [15:0];

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            dout <= 8'b0;     // Clear output on reset
        end else begin
            if (write_enable) begin
                mem[addr] <= din; // Write to memory
            end
            if (read_enable) begin
                dout <= mem[addr]; // Read from memory
            end
        end
    end

endmodule
```

2. **Testbench for RAM**

```
// Testbench for RAM Module
module tb_ram;

    // Testbench signals
    reg clk;
    reg reset;
    reg [3:0] addr;
    reg [7:0] din;
    wire [7:0] dout;
    reg write_enable;
    reg read_enable;

    // Instantiate the RAM module
    ram uut (
        .clk(clk),
        .reset(reset),
        .addr(addr),
        .din(din),
        .dout(dout),
        .write_enable(write_enable),
        .read_enable(read_enable)
    );

    // Clock generation (50 MHz clock)
    always begin
        #10 clk = ~clk; // Toggle clock every 10ns (50 MHz)
    end

    // Test sequence
    initial begin
        // Initialize signals
        clk = 0;
        reset = 0;
        addr = 4'b0000;
        din = 8'b00000000;
        write_enable = 0;
        read_enable = 0;

        // Apply reset
        reset = 1; #20;
        reset = 0; #20;
```

```
        // Test 1: Write to RAM
        addr = 4'b0001; din = 8'b10101010; write_enable = 1; read_enable =
0; #20;
        write_enable = 0; #20;

        // Test 2: Read from RAM
        addr = 4'b0001; write_enable = 0; read_enable = 1; #20;

        // Test 3: Write another value
        addr = 4'b0010; din = 8'b11110000; write_enable = 1; read_enable =
0; #20;
        write_enable = 0; #20;

        // Test 4: Read from RAM
        addr = 4'b0010; write_enable = 0; read_enable = 1; #20;

        // End simulation
        $finish;
    end

    // Monitor output (for observing during simulation)
    initial begin
        $monitor("Time: %t | Addr: %b | Write Data: %b | Read Data: %b |
dout: %b",
                 $time, addr, din, dout, dout);
    end

endmodule
```

3. How to Run the Simulation

1. **Compile** the `ram` module and the `testbench` code in your Verilog simulator (like **ModelSim** or **Vivado**).
2. **Run** the simulation, and the testbench will automatically test the RAM by:
    - Writing values to different addresses.
    - Reading values from those addresses.
3. **Check the Output** in the waveform viewer or console. You should see something like this in the console:

- Writing `8'b10101010` to address `4'b0001`.
- Reading the value back from address `4'b0001` should give `8'b10101010`.

Expected Output

For example, during the simulation:

- **Write** operation to address `4'b0001` will store `8'b10101010` in memory.
- **Read** operation from address `4'b0001` will output `8'b10101010`.