Task 2

RAM Module (Verilog):

```
module sync_ram #(
    parameter DATA_WIDTH = 8
    parameter ADDR_WIDTH = 4
)(
    input  wire clk,
    input  wire we
    input  wire [ADDR_WIDTH-1:0] addr,
    input  wire [DATA_WIDTH-1:0] din,
    output reg  [DATA_WIDTH-1:0]
);
    reg [DATA_WIDTH-1:0] mem [0:(1<<ADDR_WIDTH)-1];

    always @(posedge clk) begin
        if (we)
            mem[addr] <= din;
        else
            dout <= mem[addr];
    end

endmodule
```

Testbench (Verilog)

```
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;
    sync_ram #(.DATA_WIDTH(DATA_WIDTH),
.ADDR_WIDTH(ADDR_WIDTH)) RAM (
        .clk(clk),
        .we(we),
        .addr(addr),
        .din(din),
        .dout(dout)
    );
    always #5 clk = ~clk;
    initial begin
        clk = 0;
        we = 0;
```

```
        addr = 0;
        din = 0;
        #10 we = 1; addr = 4'b0001; din = 8'hA5;
        #10 we = 1; addr = 4'b0010; din = 8'h3C;
        #10 we = 1; addr = 4'b0011; din = 8'h7F;
        #10 we = 0; addr = 4'b0001
        #10 we = 0; addr = 4'b0010;
        #10 we = 0; addr = 4'b0011;

        #10 $finish;
    end

    // Monitor signals
    initial begin
        $monitor("Time = %0t | WE = %b | Addr = %b | Din = %h | Dout =
%h",
                 $time, we, addr, din, dout);
    end

endmodule
```