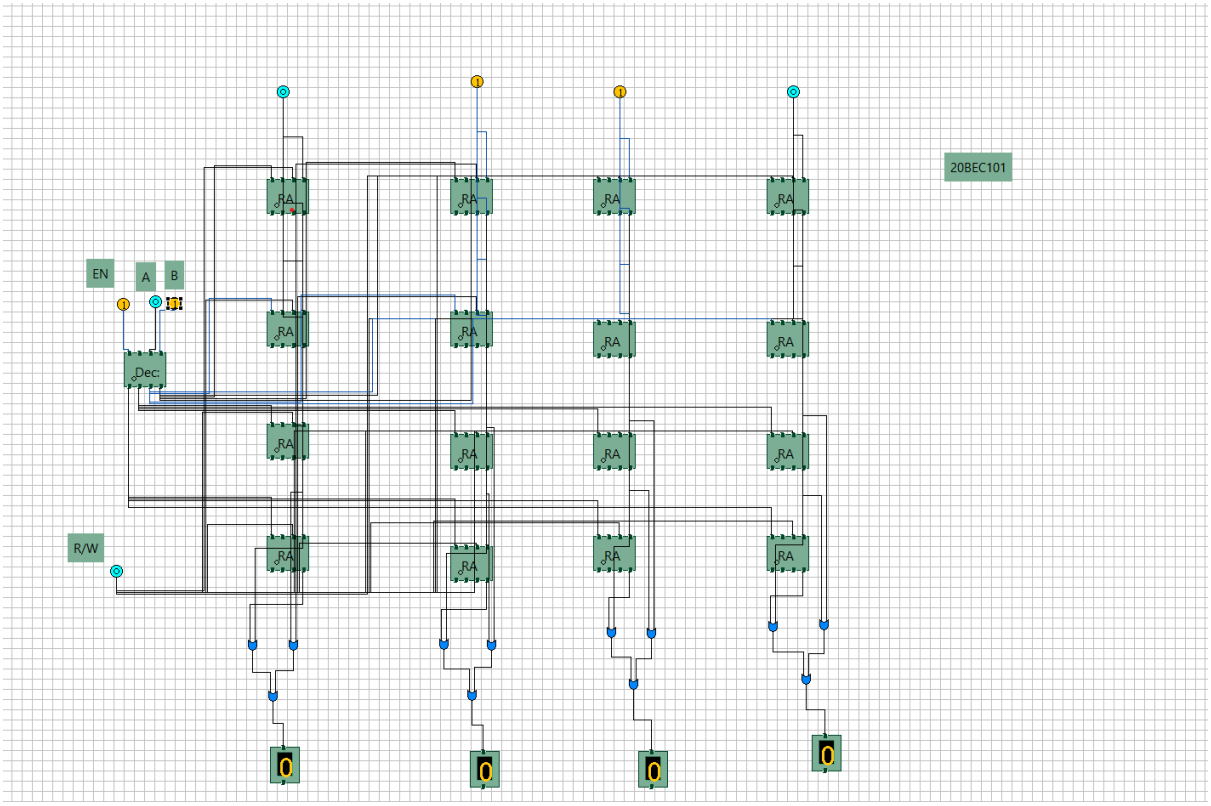
[redacted]

COMPUTER ARCHITECTURE

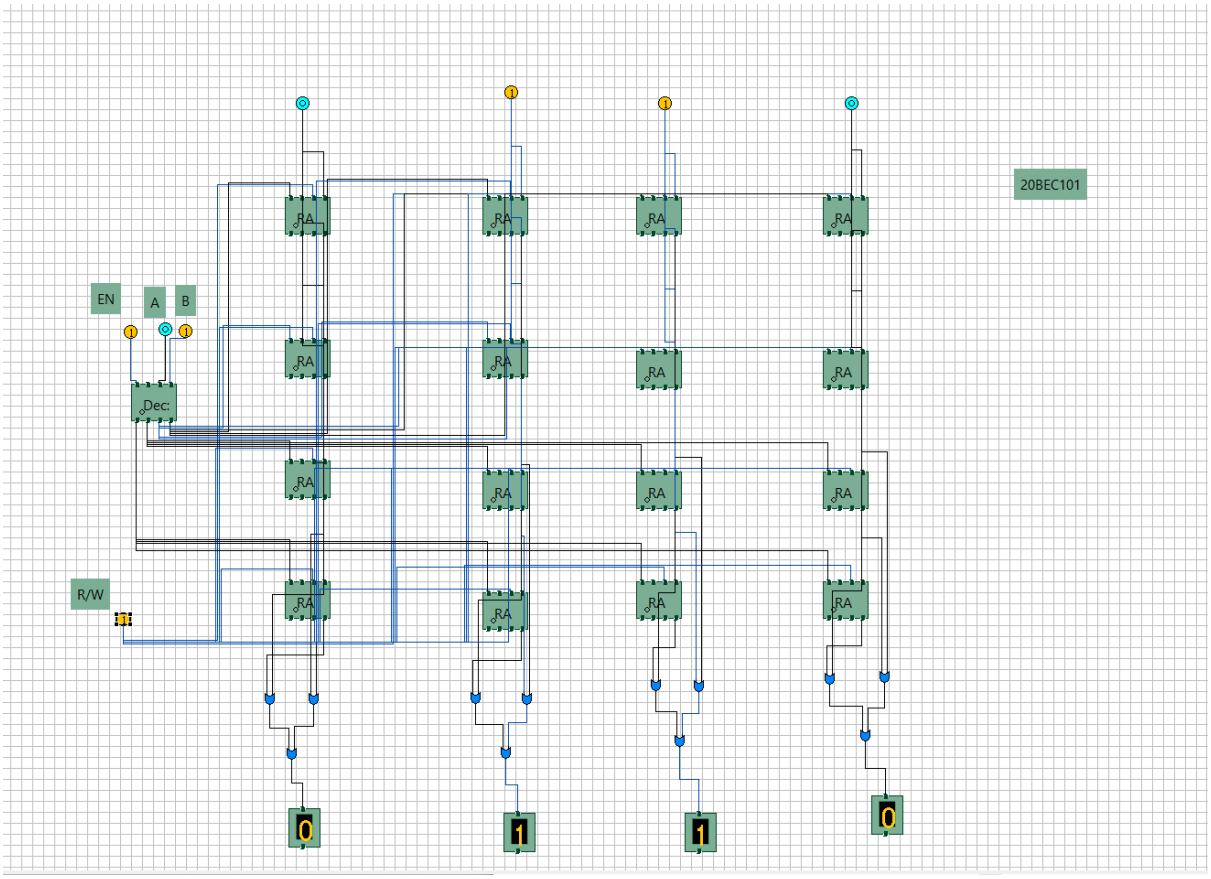
[redacted]

COAS RAM

Write:



Read:



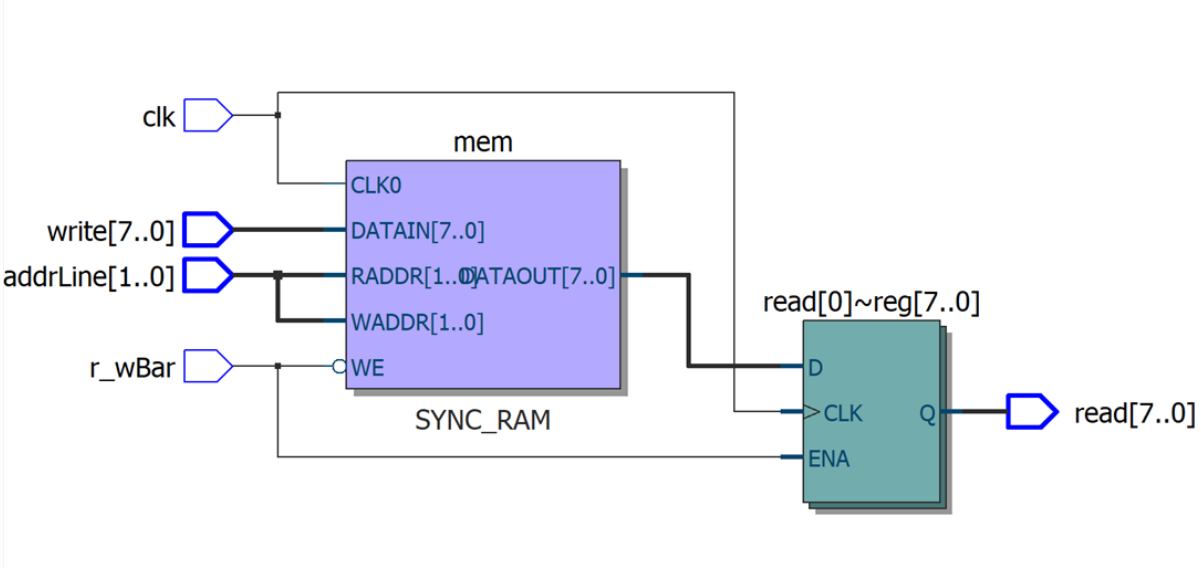
Q4) Write 4x8 RAM using Verilog HDL

Code:

```
module ram4x8(
        input [1:0] addrLine,
        input r_wBar,clk,
        input    [7:0] write,
        output reg [7:0] read
);

reg [7:0] mem [3:0];

always@(posedge clk)begin
        if(r_wBar)begin
                read <= mem[addrLine];
```

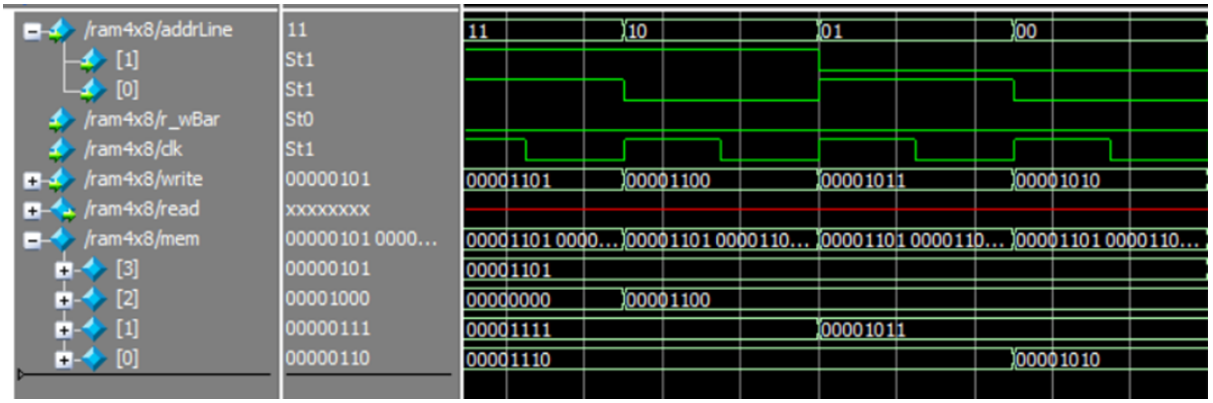
```
        end
        else begin
                mem[addrLine] <= write;
        end
end


Endmodule
```

RTL:

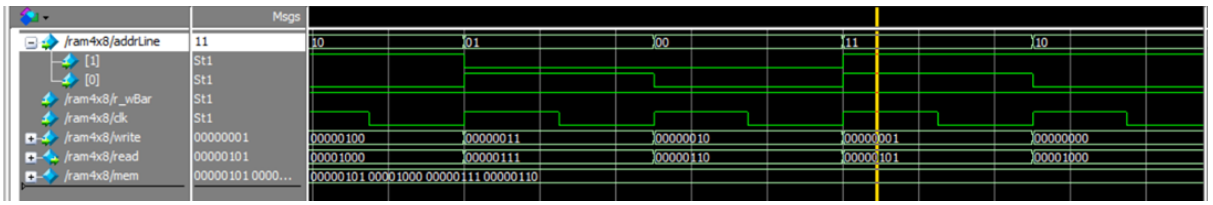


Simulation:

Write:



Read:

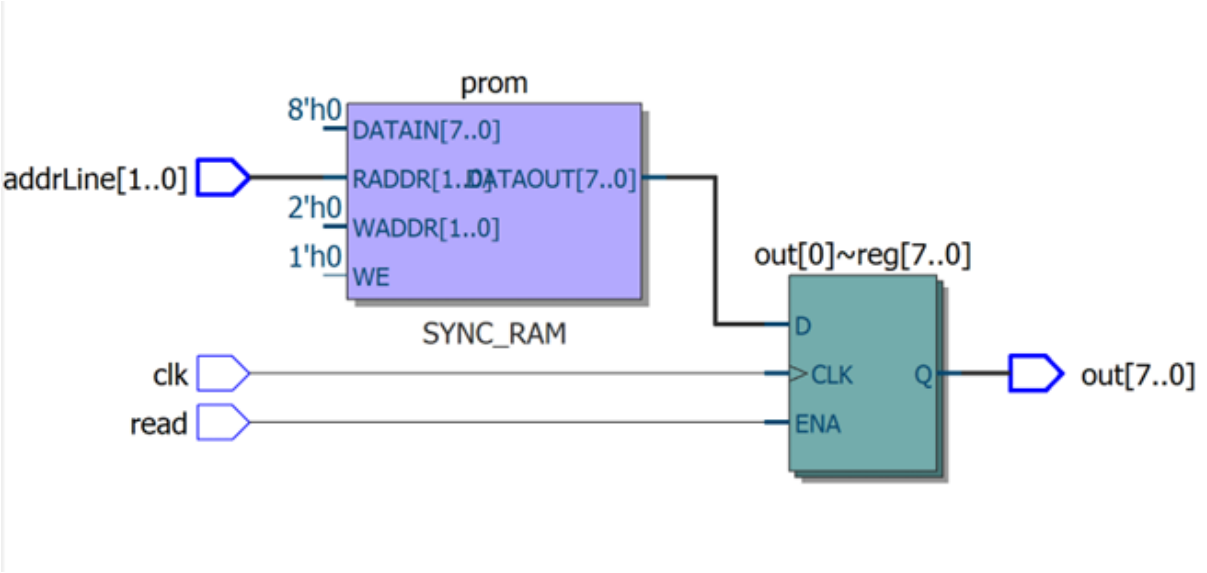


Q5) Write a Verilog code for 4X8 PROM and perform the functional simulation.

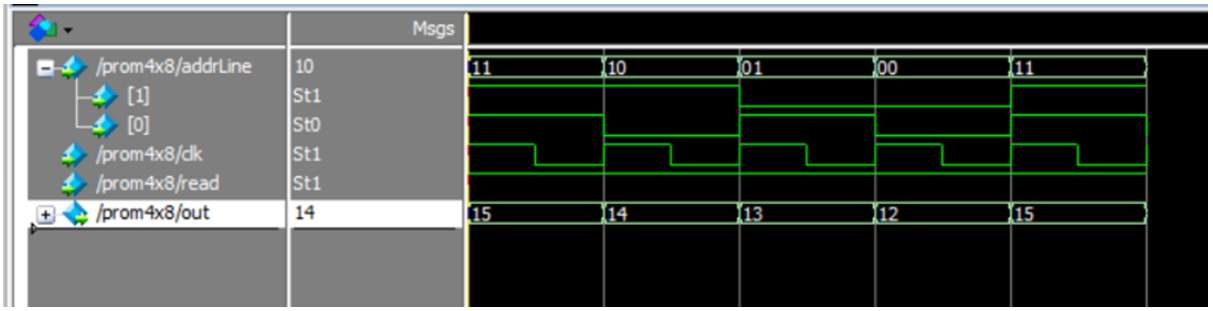
Code:

```
module prom4x8(
        input [1:0] addrLine,
        input clk,read,
        output reg [7:0] out
);

reg [7:0] prom [3:0];

initial begin
        prom[0] = 12;
        prom[1] = 13;
        prom[2] = 14;
        prom[3] = 15;
end

always@(posedge clk)begin
        if(read) begin
                out <= prom[addrLine];
        end
end

endmodule
```

RTL:



Simulation:



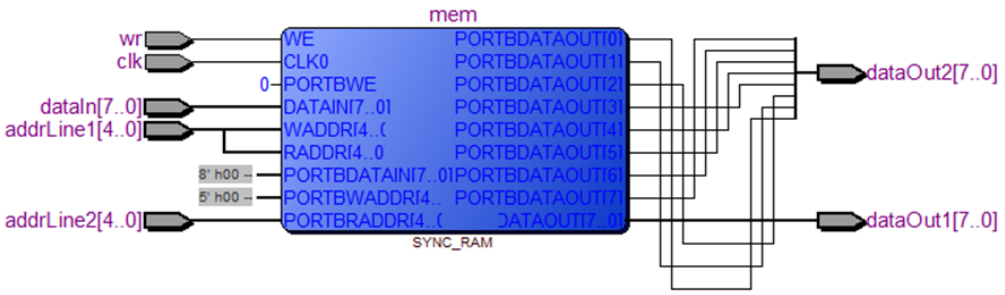
Q.6) dual RAM

Code:

```
module dualPortRam #(parameter N = 32,
                                                    parameter Add = $clog2(N),
                                                    parameter B = 8)(
        input [B-1:0]dataIn,
        input     wr,clk,
        input [Add-1:0] addrLine1,addrLine2,
        output [B-1:0] dataOut1, dataOut2
);

reg [B-1:0] mem [N-1:0];
```

```
integer i;

initial begin
        for(i=0; i < N; i=i+1)begin
                mem[i] <= i+1;
        end
end


always @ (posedge clk)begin
        if(wr)begin
                mem[addrLine1] <= dataIn;
        end
end

assign dataOut1 = mem[addrLine1];

assign dataOut2 = mem[addrLine2];


endmodule
```

RTL:



Simulation: