LAB NO 11

Fall 2024

CSE-304L Computer Organization and Architecture Lab

Submitted by:

Name : **Hassan Zaib Jadoon**

Reg no**.** **: 22PWCSE2144**

Class Section **: A**

Signature: ___________

Submitted to:

Dr. Amaad Khalil

Department of Computer Systems Engineering

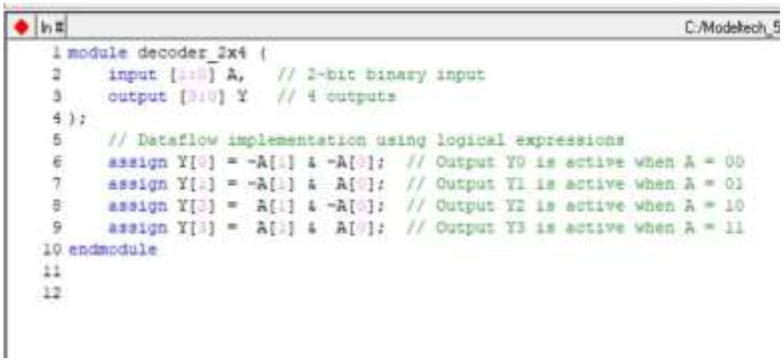
University of Engineering and Technology, Peshawar

Decoder:

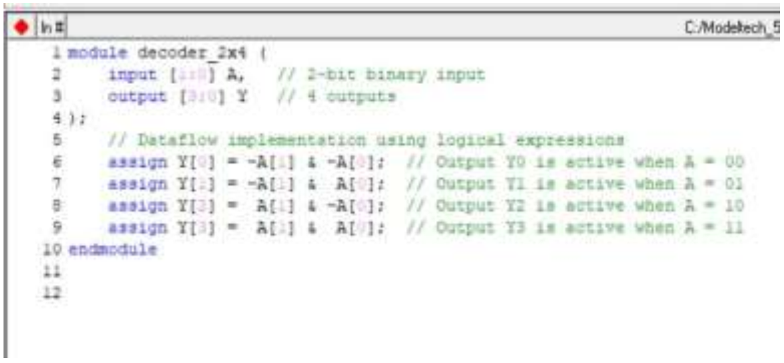
TASK1:

Write a Verilog code for 2x4 Decoder using Dataflow Level modeling.

CODE:



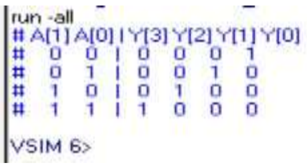
```
ln#                                                                  C:/Modeltech_5
module decoder_2x4 (
    input [1:0] A,    // 2-bit binary input
    output [3:0] Y    // 4 outputs
);
    // Dataflow implementation using logical expressions
    assign Y[0] = ~A[1] & ~A[0];  // Output Y0 is active when A = 00
    assign Y[1] = ~A[1] &  A[0];  // Output Y1 is active when A = 01
    assign Y[2] =  A[1] & ~A[0];  // Output Y2 is active when A = 10
    assign Y[3] =  A[1] &  A[0];  // Output Y3 is active when A = 11
endmodule
```



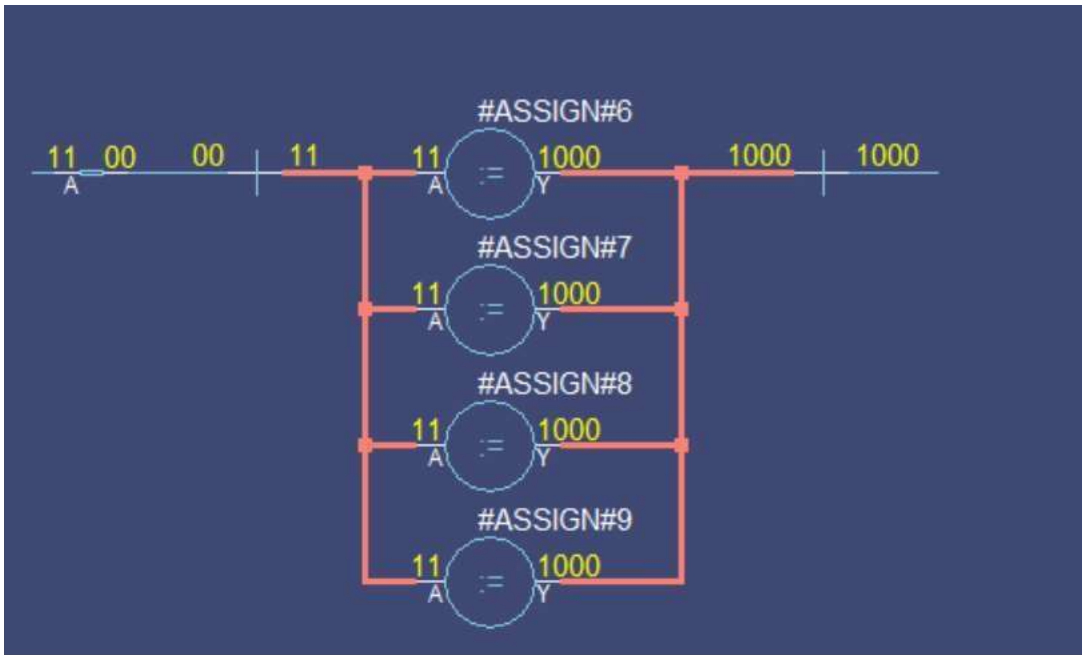
```
ln#                                                                  C:/Modeltech_5
module decoder_2x4 (
    input [1:0] A,    // 2-bit binary input
    output [3:0] Y    // 4 outputs
);
    // Dataflow implementation using logical expressions
    assign Y[0] = ~A[1] & ~A[0];  // Output Y0 is active when A = 00
    assign Y[1] = ~A[1] &  A[0];  // Output Y1 is active when A = 01
    assign Y[2] =  A[1] & ~A[0];  // Output Y2 is active when A = 10
    assign Y[3] =  A[1] &  A[0];  // Output Y3 is active when A = 11
endmodule
```

Output:

Table:



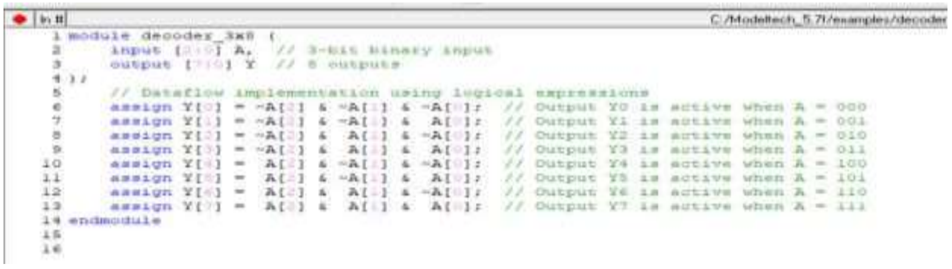
```
run -all
# A[1] A[0] | Y[3] Y[2] Y[1] Y[0]
#   0    0  |  0    0    0    1
#   0    1  |  0    0    1    0
#   1    0  |  0    1    0    0
#   1    1  |  1    0    0    0

VSIM 6>
```



Task 2:

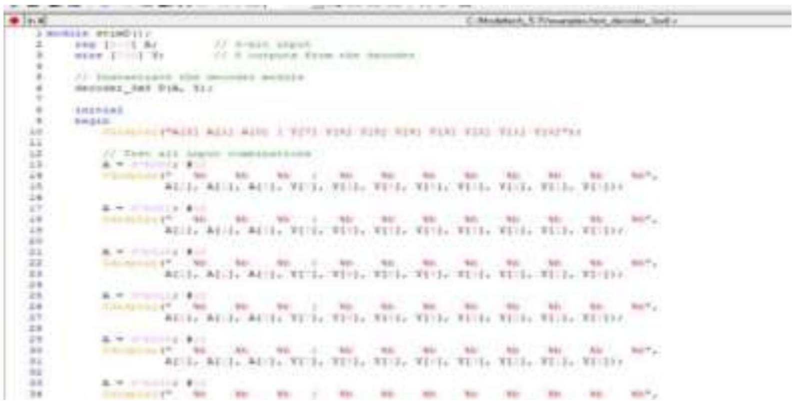
Write a Verilog code for 3x8 Decoder using Dataflow Level modeling.

Code:



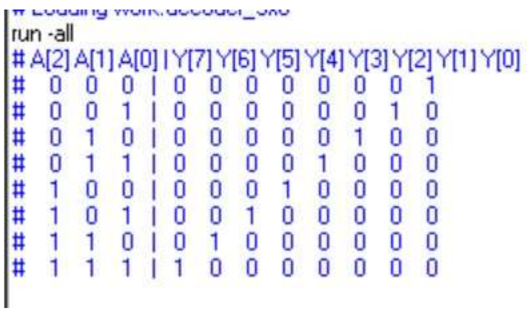
C:/Modeltech_5.7f/examples/decoder

```verilog
module decoder_3x8 (
    input [2:0] A,   // 3-bit binary input
    output [7:0] Y   // 8 outputs
);
    // Dataflow implementation using logical expressions
    assign Y[0] = ~A[2] & ~A[1] & ~A[0];  // Output Y0 is active when A = 000
    assign Y[1] = ~A[2] & ~A[1] &  A[0];  // Output Y1 is active when A = 001
    assign Y[2] = ~A[2] &  A[1] & ~A[0];  // Output Y2 is active when A = 010
    assign Y[3] = ~A[2] &  A[1] &  A[0];  // Output Y3 is active when A = 011
    assign Y[4] =  A[2] & ~A[1] & ~A[0];  // Output Y4 is active when A = 100
    assign Y[5] =  A[2] & ~A[1] &  A[0];  // Output Y5 is active when A = 101
    assign Y[6] =  A[2] &  A[1] & ~A[0];  // Output Y6 is active when A = 110
    assign Y[7] =  A[2] &  A[1] &  A[0];  // Output Y7 is active when A = 111
endmodule
```

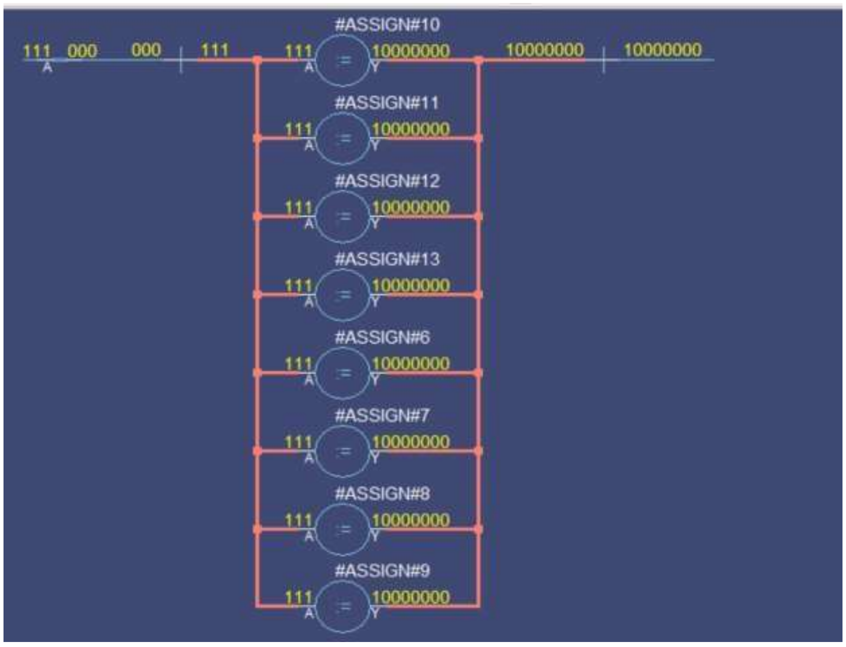


```verilog
module [illegible]();
    reg [2:0] A;        // 3-bit input
    wire [7:0] Y;       // 8 outputs from the decoder

    // Instantiate the decoder module
    decoder_3x8 [illegible](A, Y);

    initial
    begin
        $display("A[2] A[1] A[0] | Y[7] Y[6] Y[5] Y[4] Y[3] Y[2] Y[1] Y[0]");

        // Test all input combinations
        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
                 A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0]);

        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
                 A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0]);

        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
                 A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0]);

        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
                 A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0]);

        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
                 A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0]);

        A = [illegible]; #[illegible]
        $display(" %b   %b   %b  |  %b   %b   %b   %b   %b   %b   %b   %b",
```

Table output:



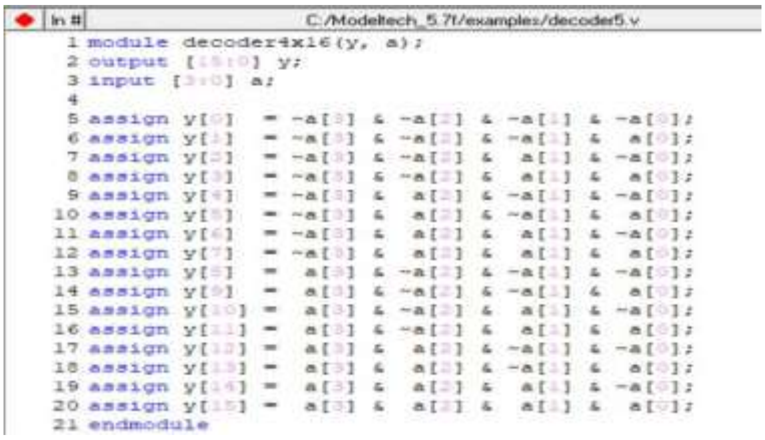
```
# Loading work.decoder_3x8
run -all
# A[2] A[1] A[0] | Y[7] Y[6] Y[5] Y[4] Y[3] Y[2] Y[1] Y[0]
#   0    0    0  |  0    0    0    0    0    0    0    1
#   0    0    1  |  0    0    0    0    0    0    1    0
#   0    1    0  |  0    0    0    0    0    1    0    0
#   0    1    1  |  0    0    0    0    1    0    0    0
#   1    0    0  |  0    0    0    1    0    0    0    0
#   1    0    1  |  0    0    1    0    0    0    0    0
#   1    1    0  |  0    1    0    0    0    0    0    0
#   1    1    1  |  1    0    0    0    0    0    0    0
```



TASK 3:

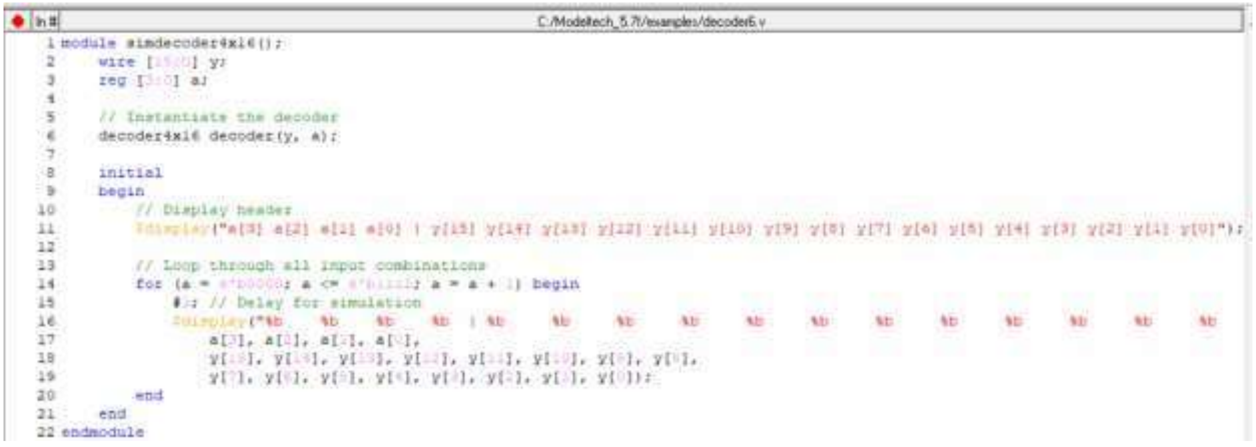
Write a Verilog code for 4x16 Decoder using Dataflow Level modeling.

Code:



C:/Modeltech_5.7f/examples/decoder5.v

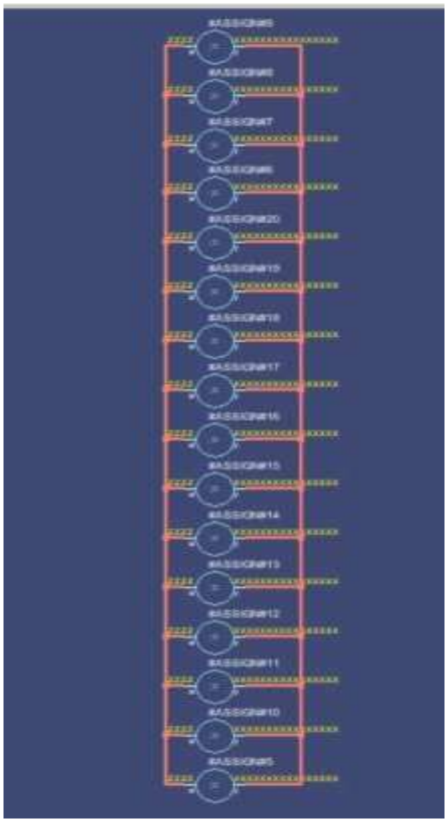
```verilog
module decoder4x16(y, a);
output [15:0] y;
input [3:0] a;

assign y[0]  = ~a[3] & ~a[2] & ~a[1] & ~a[0];
assign y[1]  = ~a[3] & ~a[2] & ~a[1] &  a[0];
assign y[2]  = ~a[3] & ~a[2] &  a[1] & ~a[0];
assign y[3]  = ~a[3] & ~a[2] &  a[1] &  a[0];
assign y[4]  = ~a[3] &  a[2] & ~a[1] & ~a[0];
assign y[5]  = ~a[3] &  a[2] & ~a[1] &  a[0];
assign y[6]  = ~a[3] &  a[2] &  a[1] & ~a[0];
assign y[7]  = ~a[3] &  a[2] &  a[1] &  a[0];
assign y[8]  =  a[3] & ~a[2] & ~a[1] & ~a[0];
assign y[9]  =  a[3] & ~a[2] & ~a[1] &  a[0];
assign y[10] =  a[3] & ~a[2] &  a[1] & ~a[0];
assign y[11] =  a[3] & ~a[2] &  a[1] &  a[0];
assign y[12] =  a[3] &  a[2] & ~a[1] & ~a[0];
assign y[13] =  a[3] &  a[2] & ~a[1] &  a[0];
assign y[14] =  a[3] &  a[2] &  a[1] & ~a[0];
assign y[15] =  a[3] &  a[2] &  a[1] &  a[0];
endmodule
```



C:/Modeltech_5.7f/examples/decoder6.v

```verilog
module simdecoder4x16();
    wire [15:0] y;
    reg [3:0] a;

    // Instantiate the decoder
    decoder4x16 decoder(y, a);

    initial
    begin
        // Display header
        $display("a[3] a[2] a[1] a[0] | y[15] y[14] y[13] y[12] y[11] y[10] y[9] y[8] y[7] y[6] y[5] y[4] y[3] y[2] y[1] y[0]");

        // Loop through all input combinations
        for (a = 4'b0000; a <= 4'b1111; a = a + 1) begin
            #1; // Delay for simulation
            $display("%b   %b   %b   %b | %b   %b   %b   %b   %b   %b   %b   %b   %b   %b   %b   %b
                a[3], a[2], a[1], a[0],
                y[15], y[14], y[13], y[12], y[11], y[10], y[9], y[8],
                y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        end
    end
endmodule
```

Output:



Remarks:

1. **Definition**: A decoder is a combinational circuit that converts binary input data into a specific output line, with only one output line active (logic 1) at a time.

2. **Purpose**: Decoders are commonly used in applications such as memory address decoding, data routing, and enabling specific devices in digital systems.

3. **Types**:

   - **2x4 Decoder**: Converts 2 input lines into 4 output lines.
   - **3x8 Decoder**: Converts 3 input lines into 8 output lines.
   - **4x16 Decoder**: Converts 4 input lines into 16 output lines. □ Larger decoders, like 5x32, can also be constructed.

4. **Working Principle**: For an n-input decoder, exactly one of the 2^n output lines is active based on the binary input combination.