EE342 Lab-1
Instructions and Example Designs

Mehmet Çalı

Requirements

- 3 bit counter module design
  - Triggered by clock rising edge
  - <u>Active-low</u> reset
- Multiplexer module design
  - 2 data inputs 1 select input
  - Performs AND operation if select is low, perform EX-OR operation if select is high
- Top module design
  - Inputs are clock, reset (to connect first module) and function select (to connect second module)
  - Connects bit-0 and bit-2 outputs of the first module to data inputs of the multiplexer

Assignments in Verilog

Procedural Assignment

- Left side must be variable (reg, integer ...)

$$\begin{tikzcd}
& \text{Left side must be variable} \arrow[dl] \arrow[dr] & \\
\boxed{\text{always block}} & & \boxed{\text{initial block}}
\end{tikzcd}$$

always block

- Can be used to represent both combinational and sequantial logic circuits

initial block

- Generally for simulation purposes

Continuous Assignment

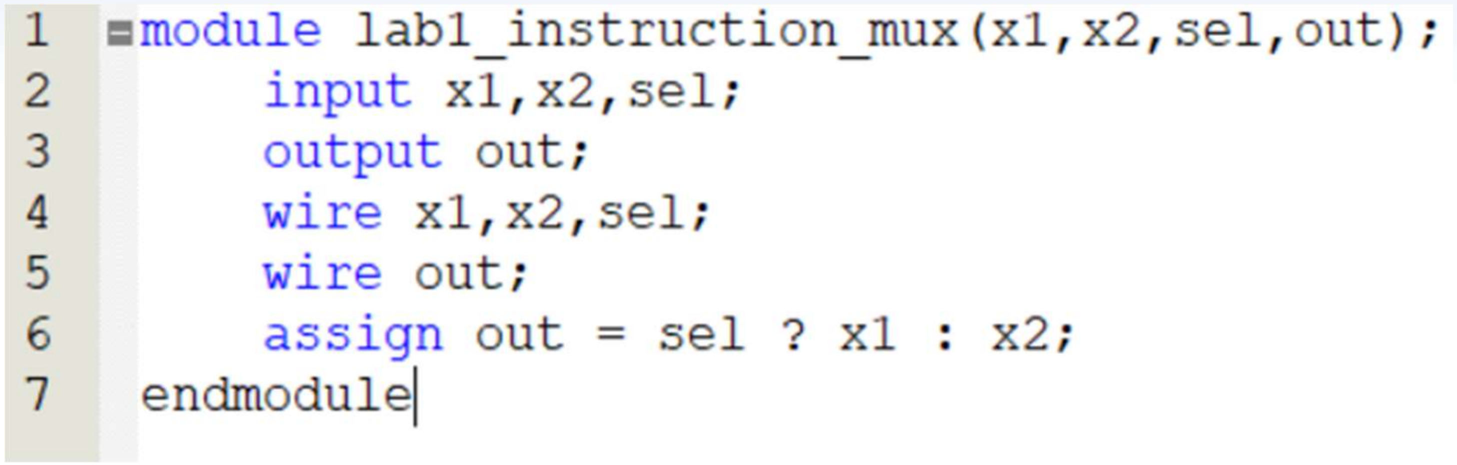
- Left side must be net (wire, tri ...)

$$\begin{tikzcd}
\text{Left side must be net} \arrow[d] \\
\boxed{\text{assign statements}}
\end{tikzcd}$$

assign statements

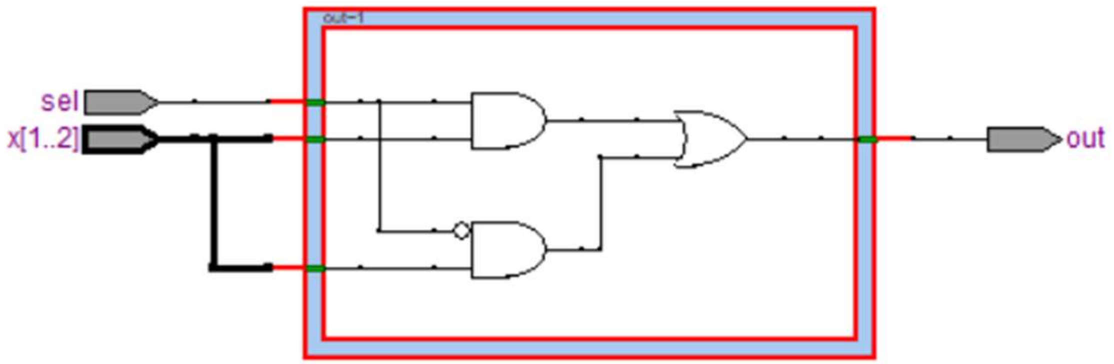
- Can only be used for combinational logic

Possible Multiplexer Designs

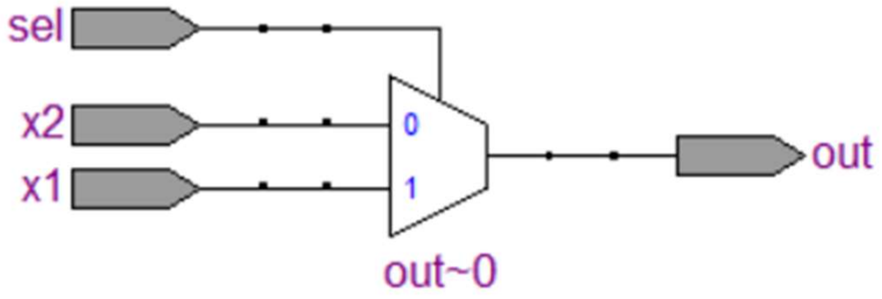


```verilog
module lab1_instruction_mux(x1,x2,sel,out);
    input x1,x2,sel;
    output out;
    wire x1,x2,sel;
    wire out;
    assign out = sel ? x1 : x2;
endmodule
```

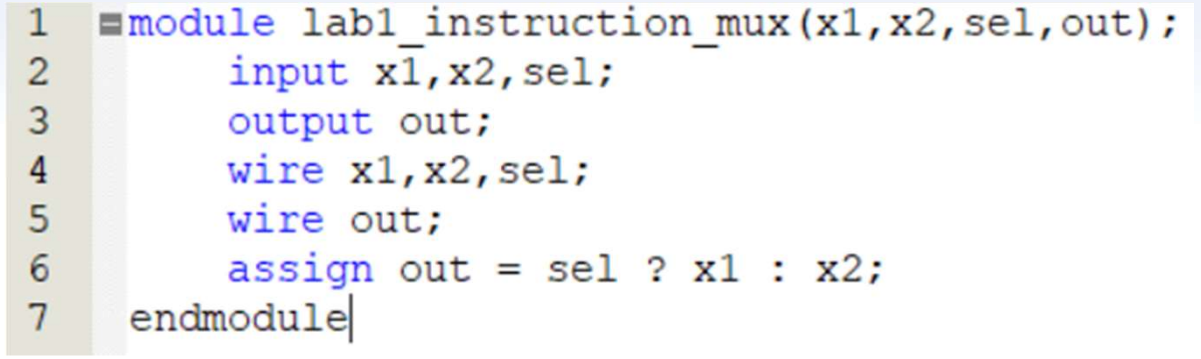
Technology map viewer:



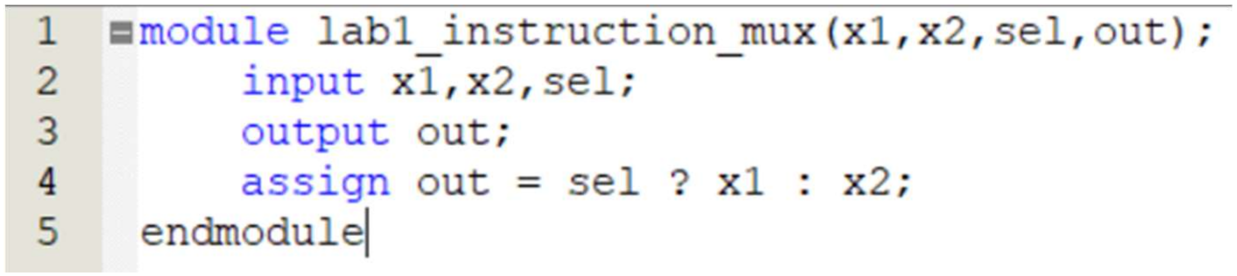
RTL viewer:



Possible Multiplexer Designs

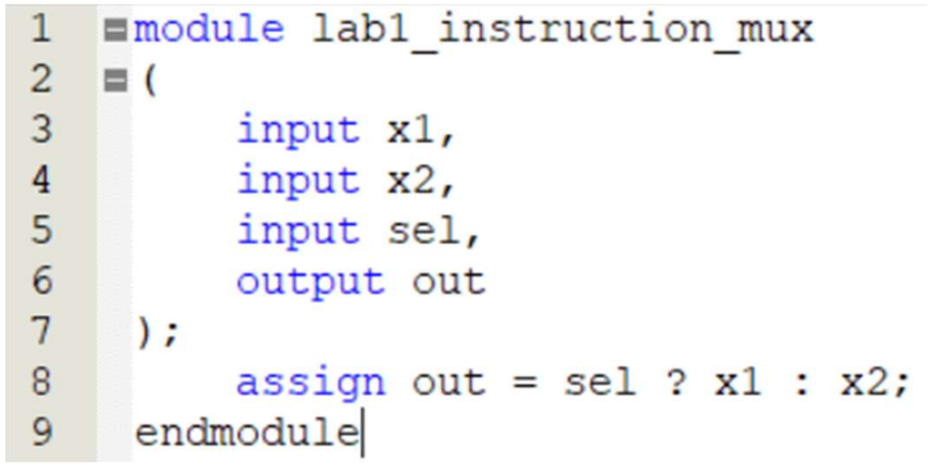


```
module lab1_instruction_mux(x1,x2,sel,out);
     input x1,x2,sel;
     output out;
     wire x1,x2,sel;
     wire out;
     assign out = sel ? x1 : x2;
endmodule
```



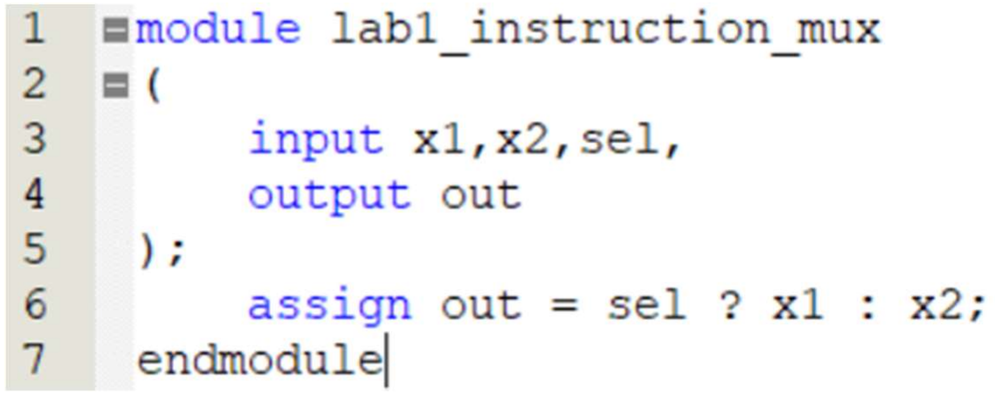
```
module lab1_instruction_mux(x1,x2,sel,out);
     input x1,x2,sel;
     output out;
     assign out = sel ? x1 : x2;
endmodule
```

Verilog 2001:



```
module lab1_instruction_mux
(
     input x1,
     input x2,
     input sel,
     output out
);
     assign out = sel ? x1 : x2;
endmodule
```

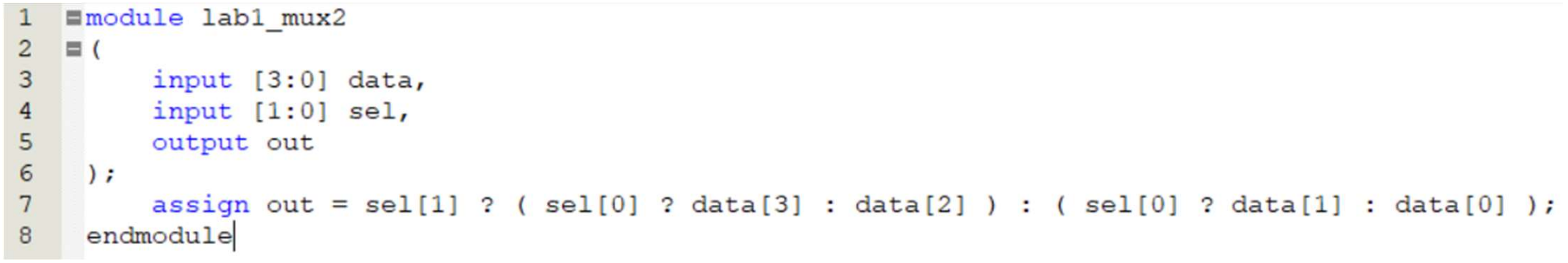
or simply:



```
module lab1_instruction_mux
(
     input x1,x2,sel,
     output out
);
     assign out = sel ? x1 : x2;
endmodule
```

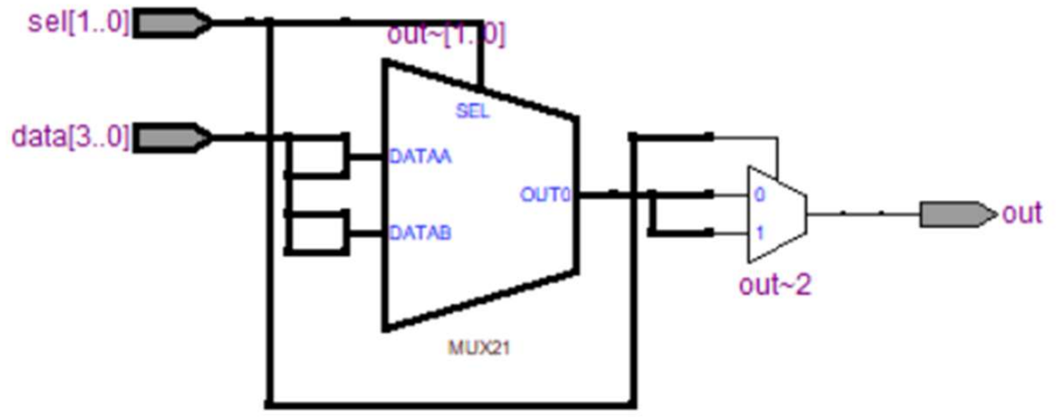
Possible Multiplexer Designs

- Lets try multiplexer with 4 data inputs:



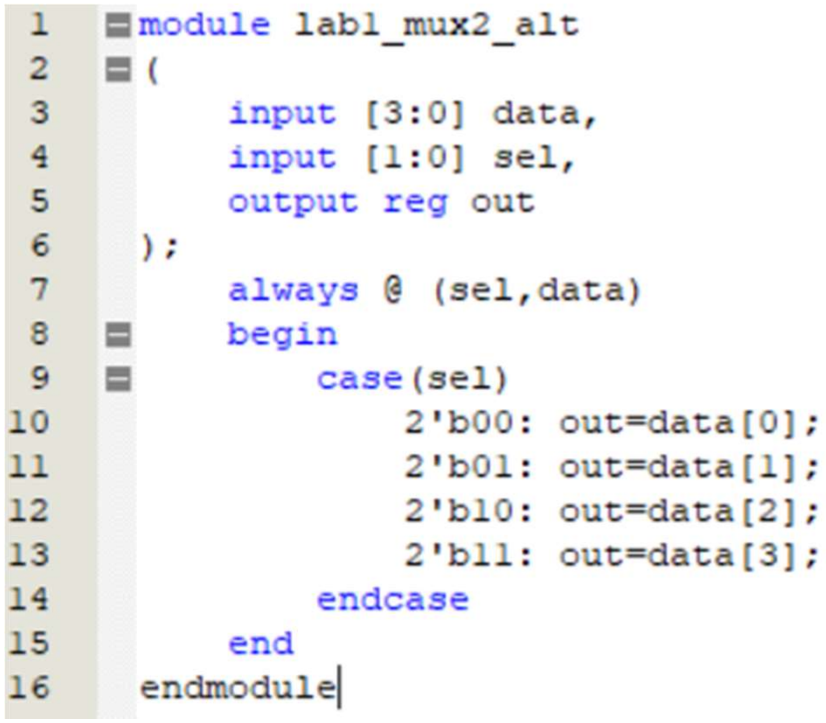
```
module lab1_mux2
(
    input [3:0] data,
    input [1:0] sel,
    output out
);
    assign out = sel[1] ? ( sel[0] ? data[3] : data[2] ) : ( sel[0] ? data[1] : data[0] );
endmodule
```

RTL viewer:



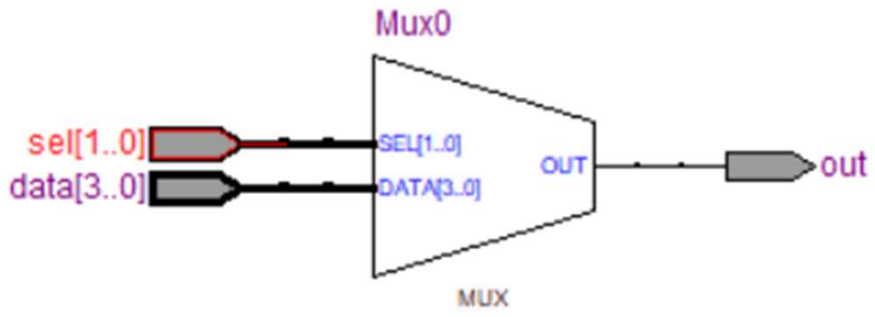
Possible Multiplexer Designs

- Lets try to design this with case statement:



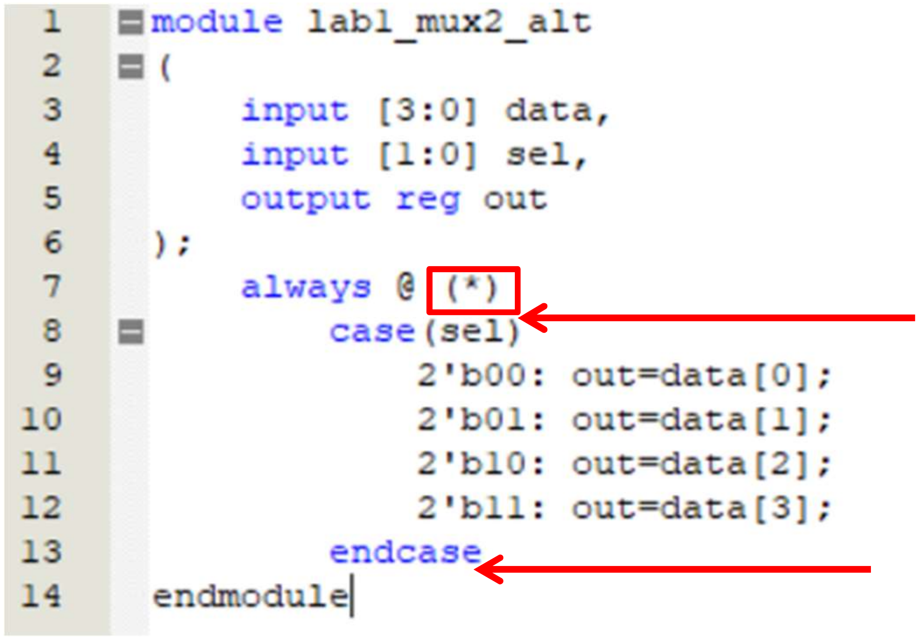
```
module lab1_mux2_alt
(
    input [3:0] data,
    input [1:0] sel,
    output reg out
);
    always @ (sel,data)
    begin
        case(sel)
            2'b00: out=data[0];
            2'b01: out=data[1];
            2'b10: out=data[2];
            2'b11: out=data[3];
        endcase
    end
endmodule
```

RTL viewer:



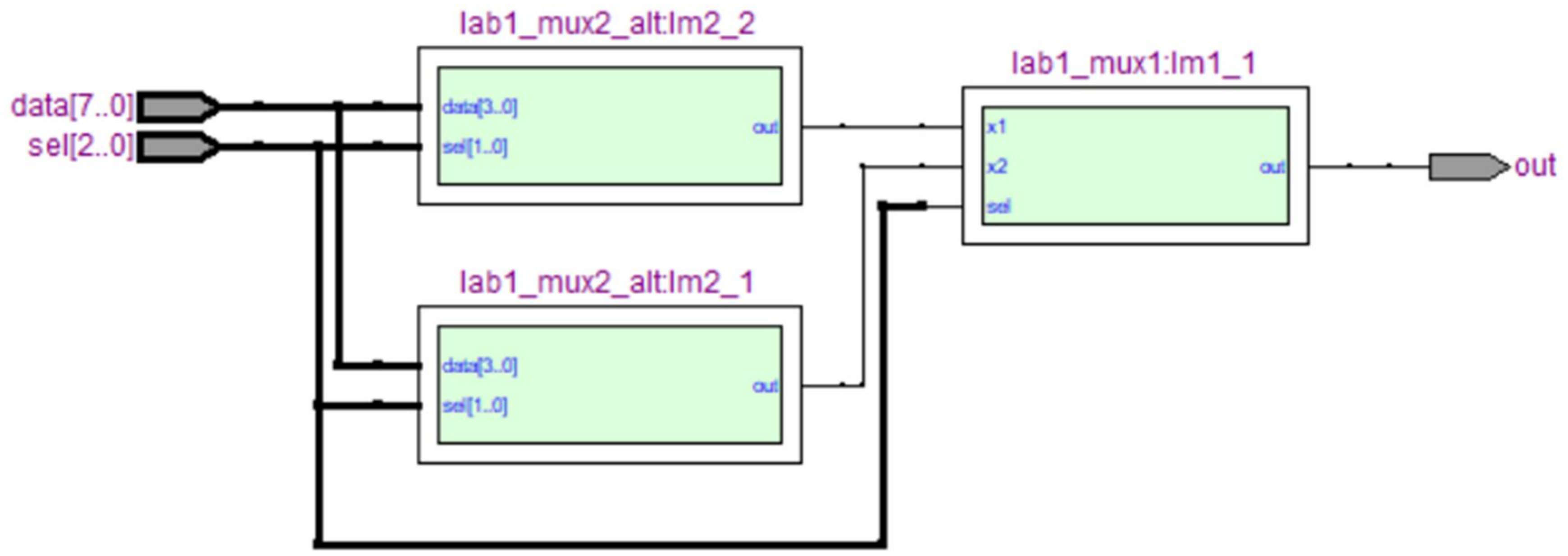
Module Instantiation

- In a more practical way:

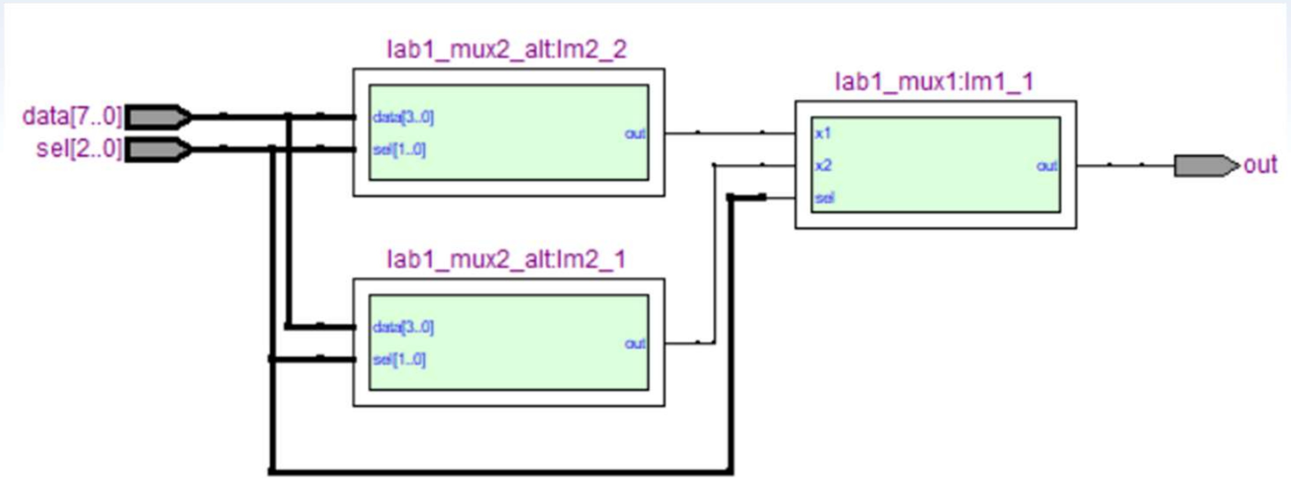


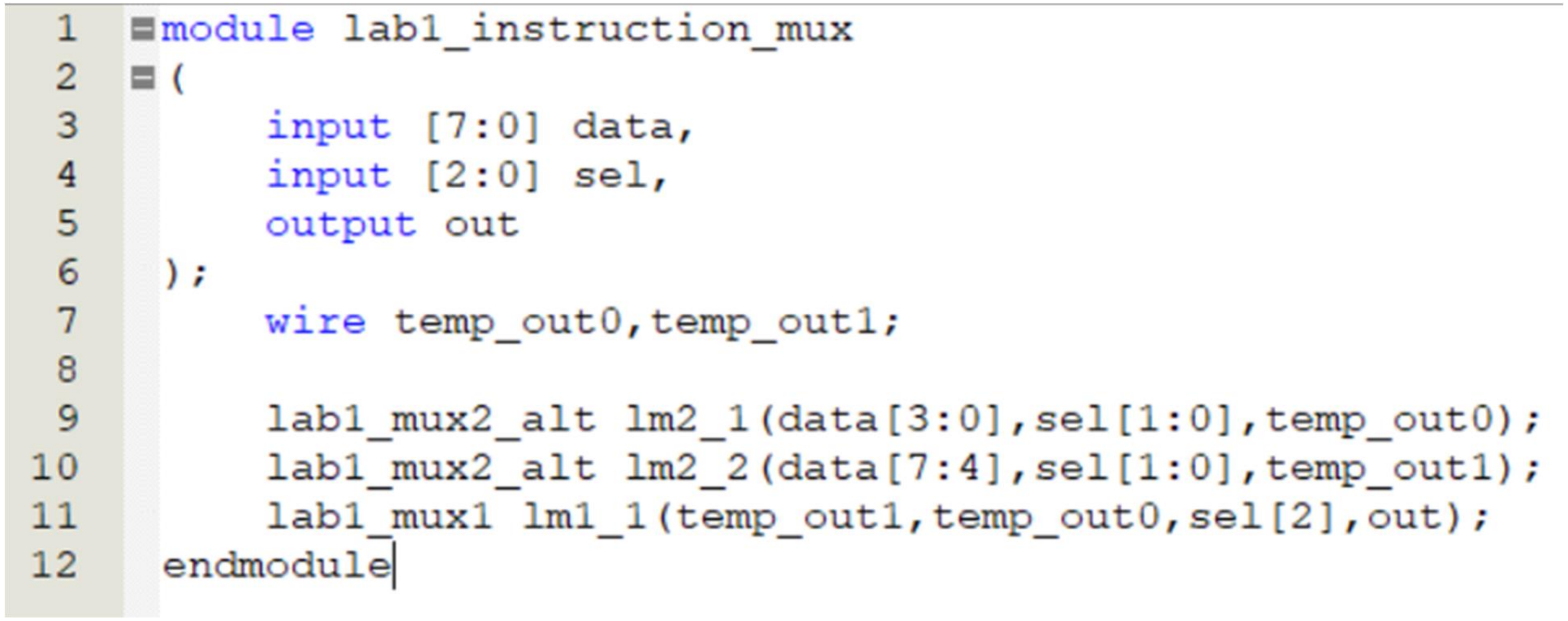
```
module lab1_mux2_alt
(
    input [3:0] data,
    input [1:0] sel,
    output reg out
);
    always @ (*)
        case(sel)
            2'b00: out=data[0];
            2'b01: out=data[1];
            2'b10: out=data[2];
            2'b11: out=data[3];
        endcase
endmodule
```

Module Instantiation



Module Instantiation

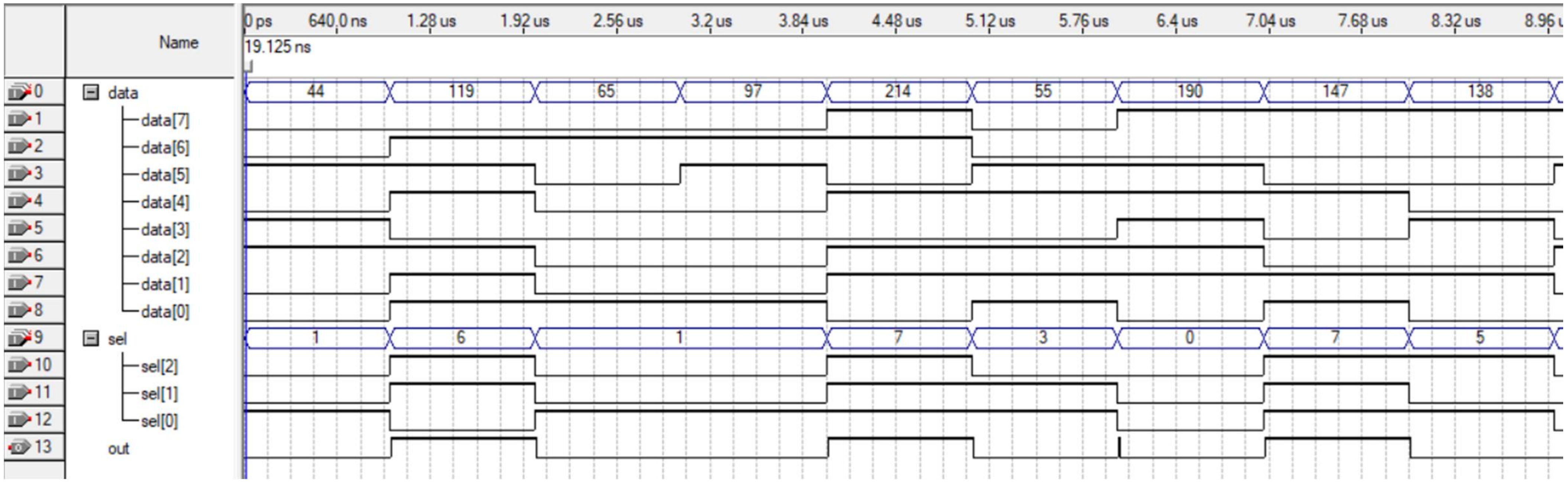




```
module lab1_instruction_mux
(
    input [7:0] data,
    input [2:0] sel,
    output out
);
    wire temp_out0,temp_out1;

    lab1_mux2_alt lm2_1(data[3:0],sel[1:0],temp_out0);
    lab1_mux2_alt lm2_2(data[7:4],sel[1:0],temp_out1);
    lab1_mux1 lm1_1(temp_out1,temp_out0,sel[2],out);
endmodule
```

Test Results



Tips for the Assessment

- Be careful about the indentation
- You should be able to explain the every decision you made during your design steps
- You should be able came up with alternative solution of the same problem and solution of the alternative similar problems
- You should adjust your waveform file for many different input combinations and you should be able to interpret the result at a specific time instant quickly