

Course Code: CS223

Course Name: Digital Design Section: 4

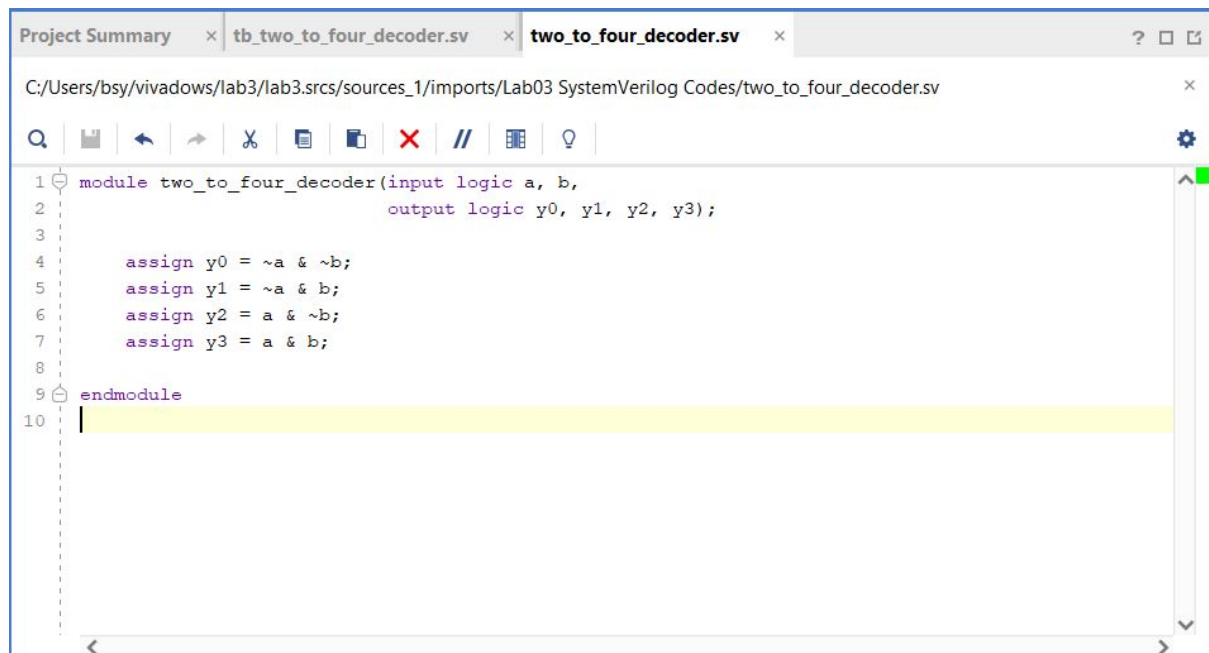
Lab 2

Name: Berk Saltuk Yılmaz ID: 21903419

Date: 16.10.2020

Part (b):

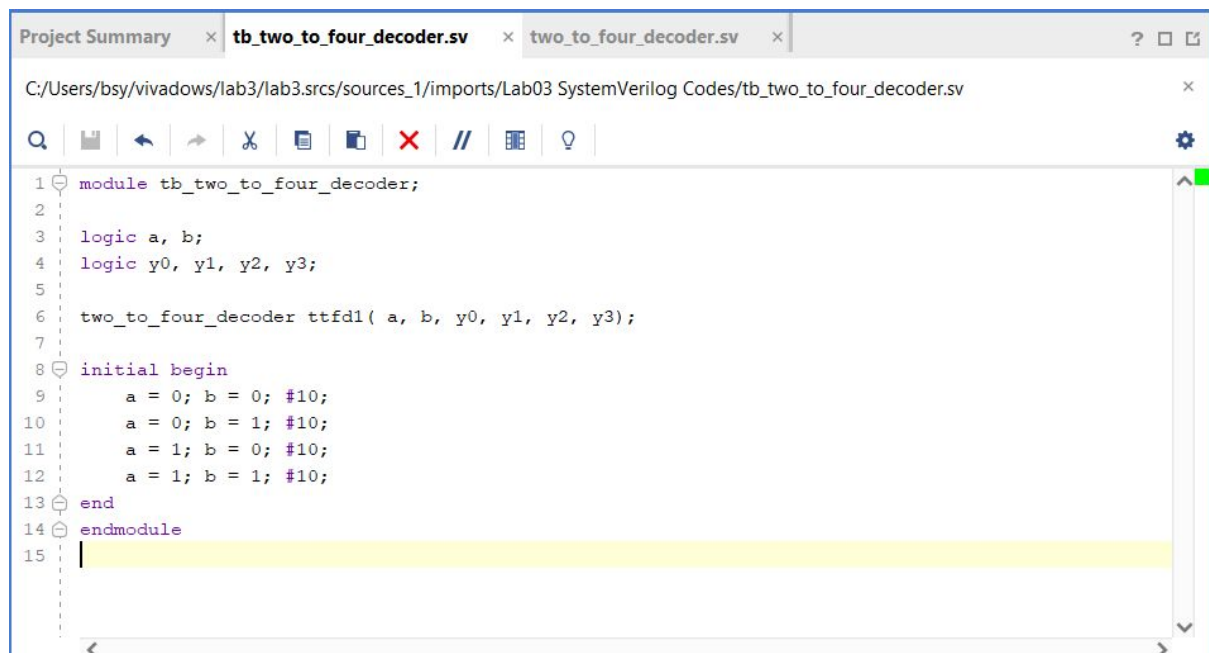
Behavioral SystemVerilog module for 2-to-4 decoder:



The screenshot shows a Vivado IDE window with the file `two_to_four_decoder.sv` open. The code defines a module `two_to_four_decoder` with two input logic signals `a` and `b`, and four output logic signals `y0`, `y1`, `y2`, and `y3`. The module uses four `assign` statements to implement the 2-to-4 decoder logic: `y0` is the complement of `a` and `b`, `y1` is the complement of `a` and `b`, `y2` is `a` and the complement of `b`, and `y3` is `a` and `b`. The module is enclosed in `module` and `endmodule` keywords.

```
1 module two_to_four_decoder(input logic a, b,  
2                             output logic y0, y1, y2, y3);  
3  
4     assign y0 = ~a & ~b;  
5     assign y1 = ~a & b;  
6     assign y2 = a & ~b;  
7     assign y3 = a & b;  
8  
9 endmodule
```

Testbench for behavioral SystemVerilog module for the full adder:

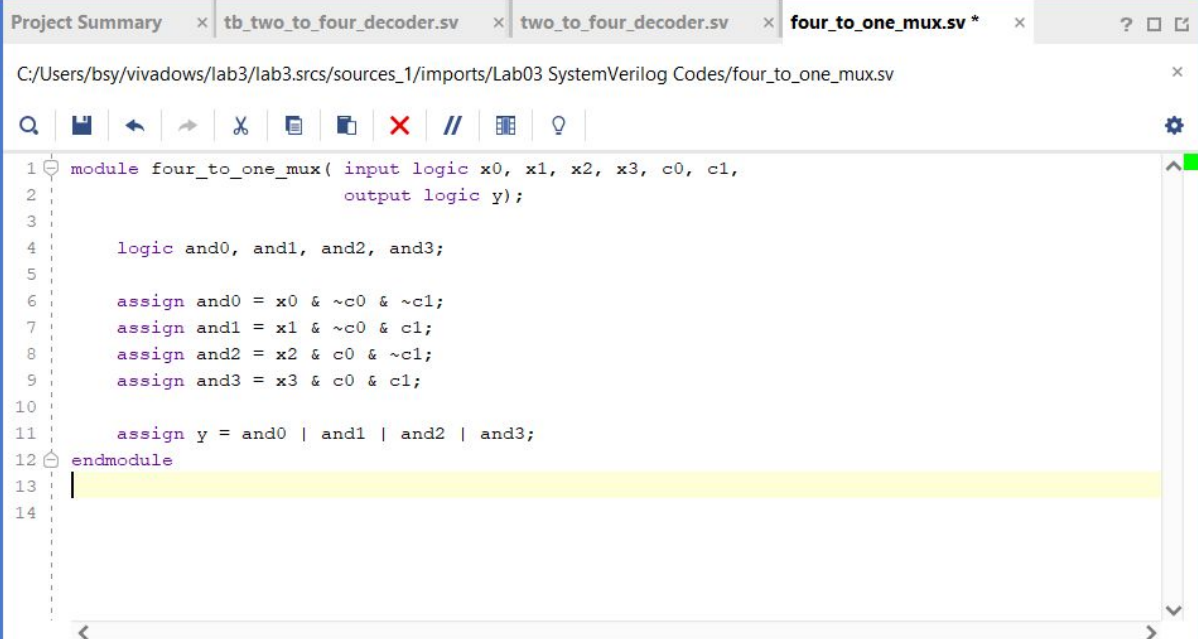


The screenshot shows a Vivado IDE window with the file `tb_two_to_four_decoder.sv` open. The code defines a testbench module `tb_two_to_four_decoder`. It declares two input logic signals `a` and `b`, and four output logic signals `y0`, `y1`, `y2`, and `y3`. It instantiates the `two_to_four_decoder` module as `ttfd1` with inputs `a`, `b` and outputs `y0`, `y1`, `y2`, `y3`. The testbench includes an `initial` block that sets the values of `a` and `b` to all possible combinations (00, 01, 10, 11) with a delay of 10 time units between each combination. The testbench is enclosed in `module` and `endmodule` keywords.

```
1 module tb_two_to_four_decoder;  
2  
3     logic a, b;  
4     logic y0, y1, y2, y3;  
5  
6     two_to_four_decoder ttfd1( a, b, y0, y1, y2, y3);  
7  
8     initial begin  
9         a = 0; b = 0; #10;  
10        a = 0; b = 1; #10;  
11        a = 1; b = 0; #10;  
12        a = 1; b = 1; #10;  
13    end  
14 endmodule
```

Part (c):

Behavioral SystemVerilog module for 4-to-1 multiplexer:

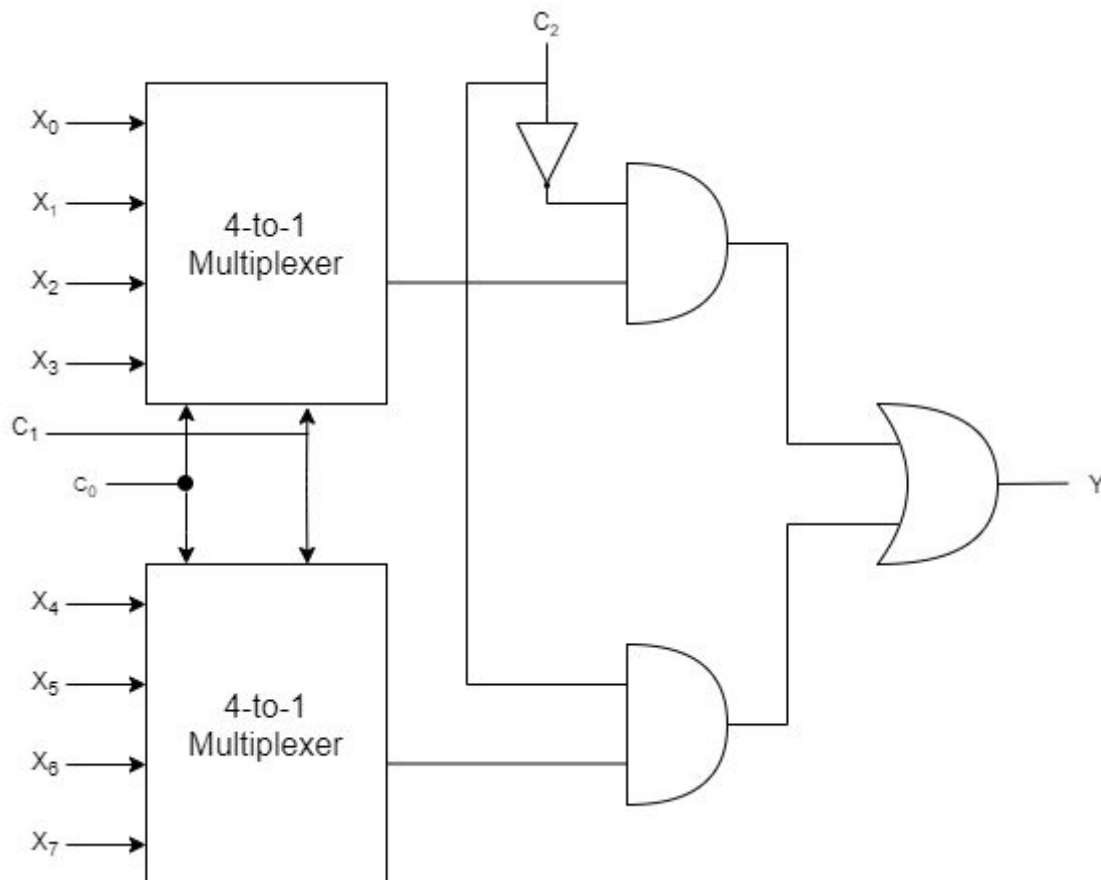


The screenshot shows a Vivado IDE window with the file explorer at the top displaying the project structure. The main editor window shows the code for the `four_to_one_mux` module. The code is as follows:

```
1 module four_to_one_mux( input logic x0, x1, x2, x3, c0, c1,
2                       output logic y);
3
4     logic and0, and1, and2, and3;
5
6     assign and0 = x0 & ~c0 & ~c1;
7     assign and1 = x1 & ~c0 & c1;
8     assign and2 = x2 & c0 & ~c1;
9     assign and3 = x3 & c0 & c1;
10
11     assign y = and0 | and1 | and2 | and3;
12 endmodule
13
14
```

Part (d):

Schematic for structural 8-to-1 MUX by using two 4-to-1 MUX modules, two AND gates, an INVERTER, and an OR gate:



Structural SystemVerilog module for 8-to-1 MUX by using two 4-to-1 MUX modules, two AND gates, an INVERTER, and an OR gate:

The screenshot shows the Vivado IDE with the file `eight_to_one_mux sv` open. The code defines a module `eight_to_one_mux` with eight input signals (`x0` through `x7`), two control signals (`c0`, `c1`), and one output signal (`y`). The module uses two instances of a `four_to_one_mux` module to implement the 8-to-1 MUX logic. The first instance, `ftom1`, takes inputs `x0`, `x1`, `x2`, `x3` and control signals `c0`, `c1` to produce `result1`. The second instance, `ftom2`, takes inputs `x4`, `x5`, `x6`, `x7` and control signals `c0`, `c1` to produce `result2`. The final output `y` is assigned as `(result1 & c2) | (result2 & c2)`.

```

1 module eight_to_one_mux( input logic x0, x1, x2, x3,
2                          x4, x5, x6, x7,
3                          c0, c1, c2,
4                          output logic y);
5
6     logic result1, result2;
7
8     four_to_one_mux ftom1(x0, x1, x2, x3, c0, c1, result1);
9     four_to_one_mux ftom2(x4, x5, x6, x7, c0, c1, result2);
10
11     assign y = (result1 & c2) | (result2 & c2);
12
13 endmodule
14

```

Testbench for structural SystemVerilog module for 8-to-1 MUX:

The screenshot shows the Vivado IDE with the file `tb_eight_to_one_mux sv` open. The testbench module `tb_eight_to_one_mux` instantiates the `eight_to_one_mux` module and applies a sequence of test vectors. It uses a loop to iterate through all possible combinations of the eight input signals (`x0` through `x7`) and the two control signals (`c0`, `c1`), with a delay of 1 time unit between each vector.

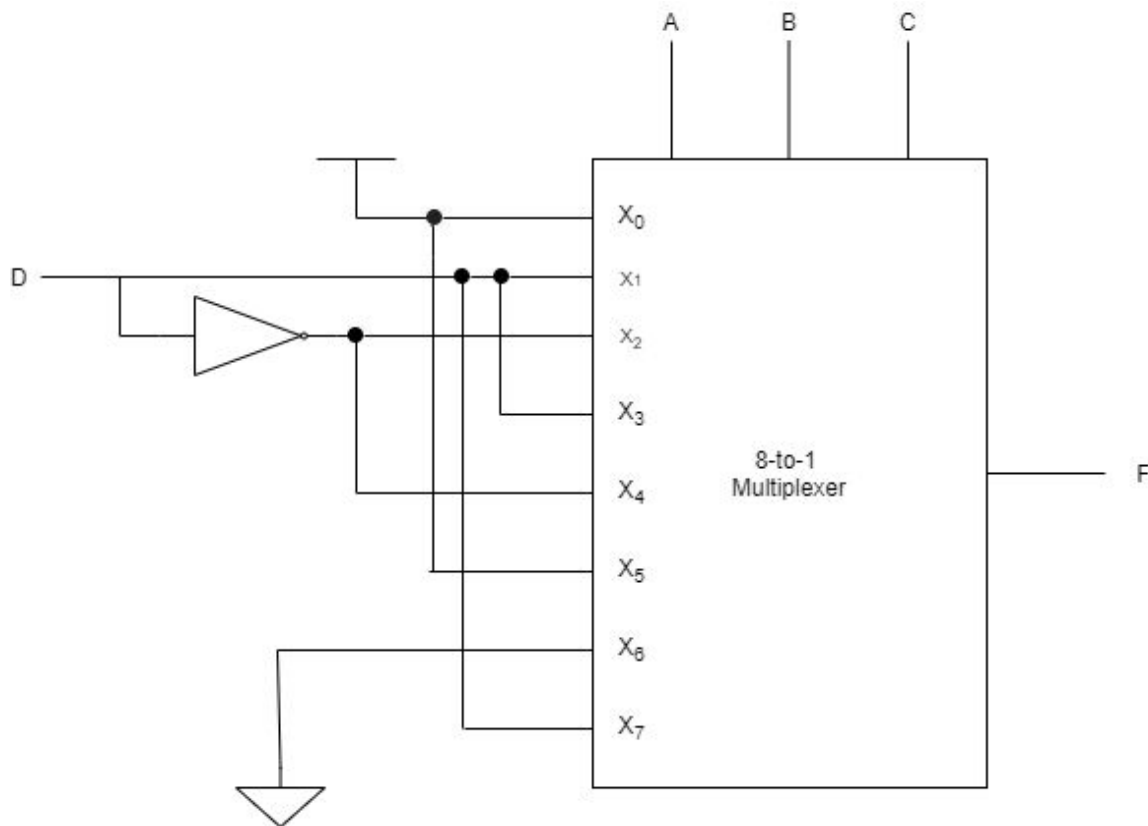
```

1 module tb_eight_to_one_mux;
2     logic x0, x1, x2, x3, x4, x5, x6, x7, c0, c1, c2, y;
3
4     eight_to_one_mux etom1(x0, x1, x2, x3, x4, x5, x6, x7, c0, c1, c2, y);
5
6     integer i;
7     initial begin
8
9         for( i = 0; i < 2**11; i++) begin
10             {x0, x1, x2, x3, x4, x5, x6, x7, c0, c1, c2} = i; #1;
11         end
12     end
13
14 endmodule
15

```

Part (e):

Schematic for $F(A,B,C,D)=\sum(0,1,3,4,7,8,10,11,15)$ function, using one 8-to-1 multiplexer and an inverter:



SystemVerilog module for $F(A,B,C,D)=\sum(0,1,3,4,7,8,10,11,15)$ function, using one 8-to-1 multiplexer and an inverter:

