CSCI 355 - Lab 4 Report

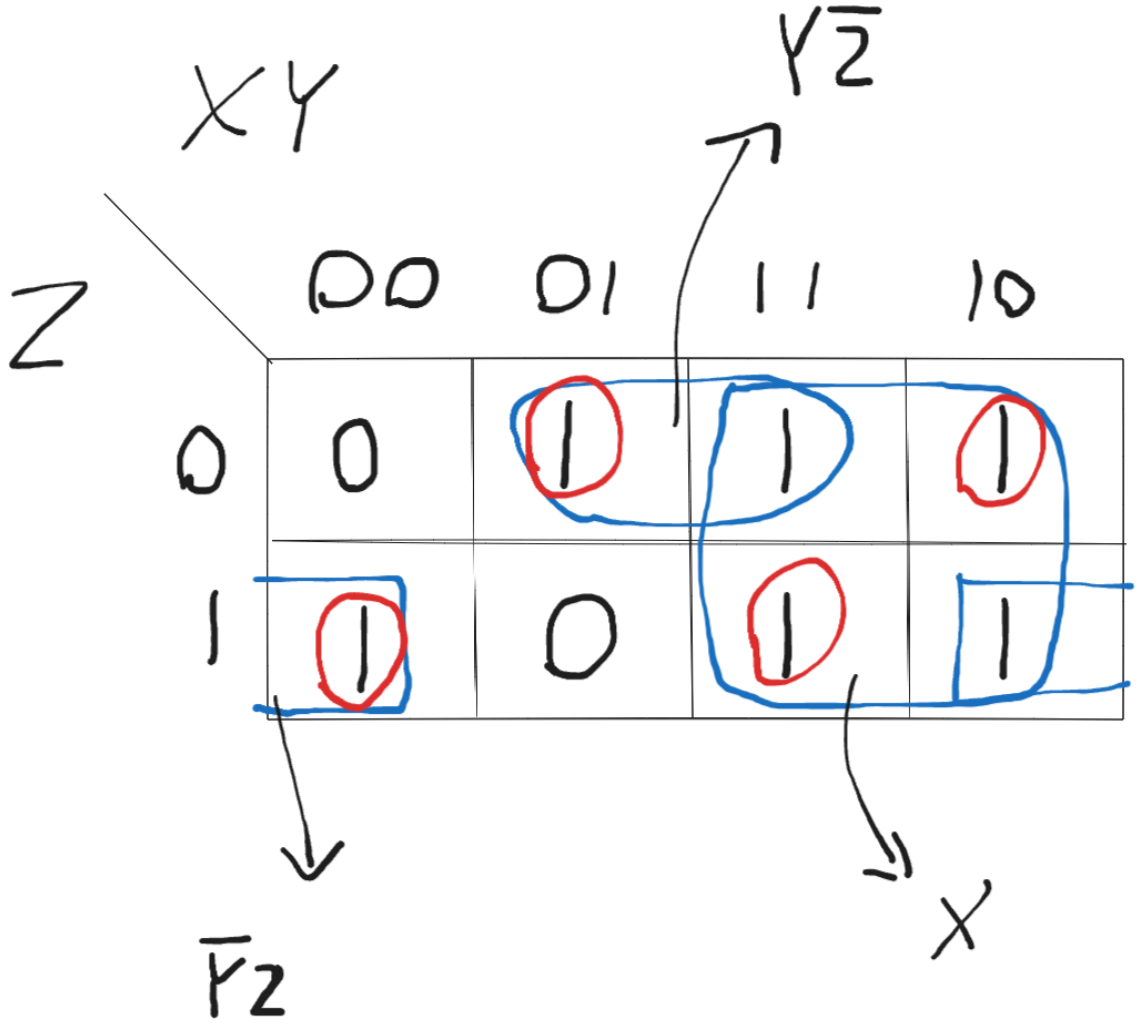
Student: Luka Karanovic
Student #: 665778833
Instructor: Ajay Shrestha

1 - Pre-Lab Answers

Note: All K-map grouping are circled in blue. All distinguished cells are circled in red.

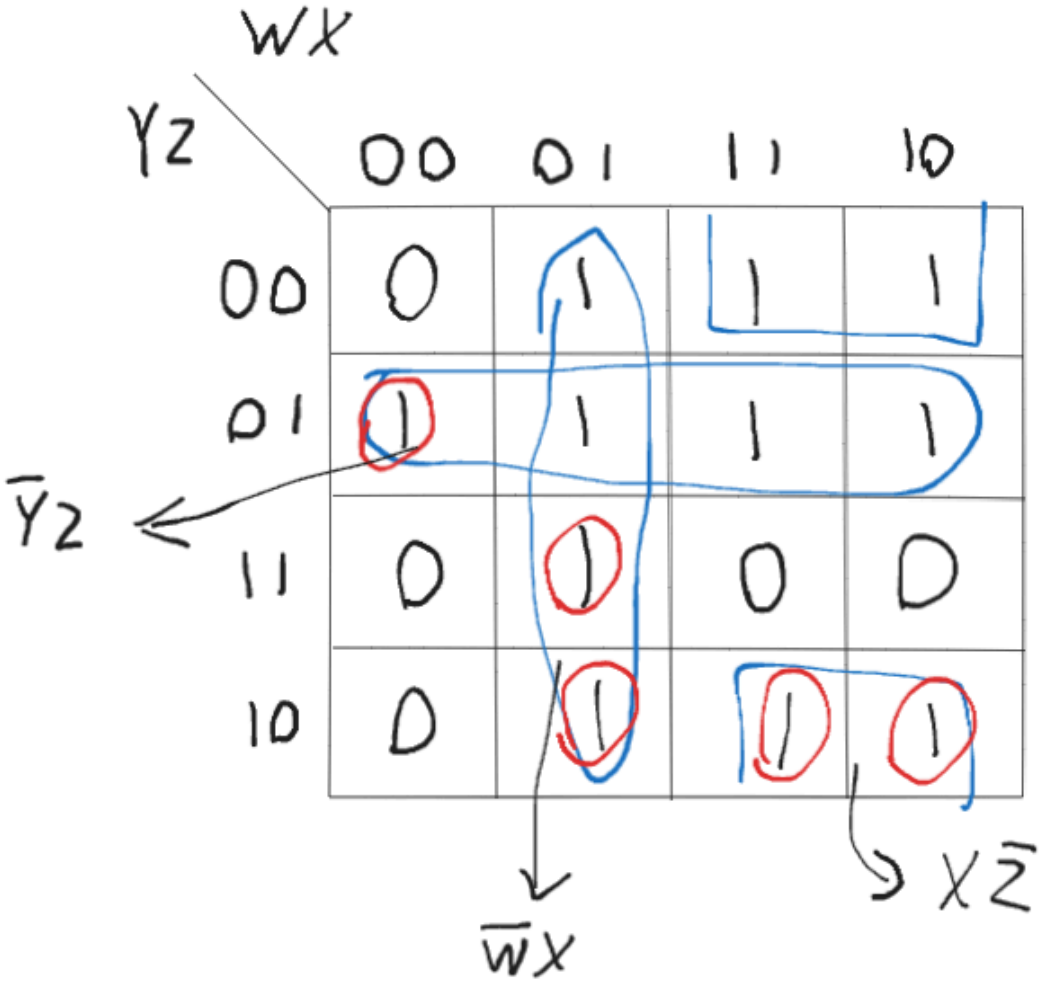
4.1 - Using Karnaugh maps, find minimal sum-of-products expressions for the following logic functions. Indicate the distinguished 1-cells in each one that you do.

a) **F = ~XY~Z + X~Z + XY + XZ + ~YZ**



Minimum-cost SOP is F = ~YZ + Y~Z + X

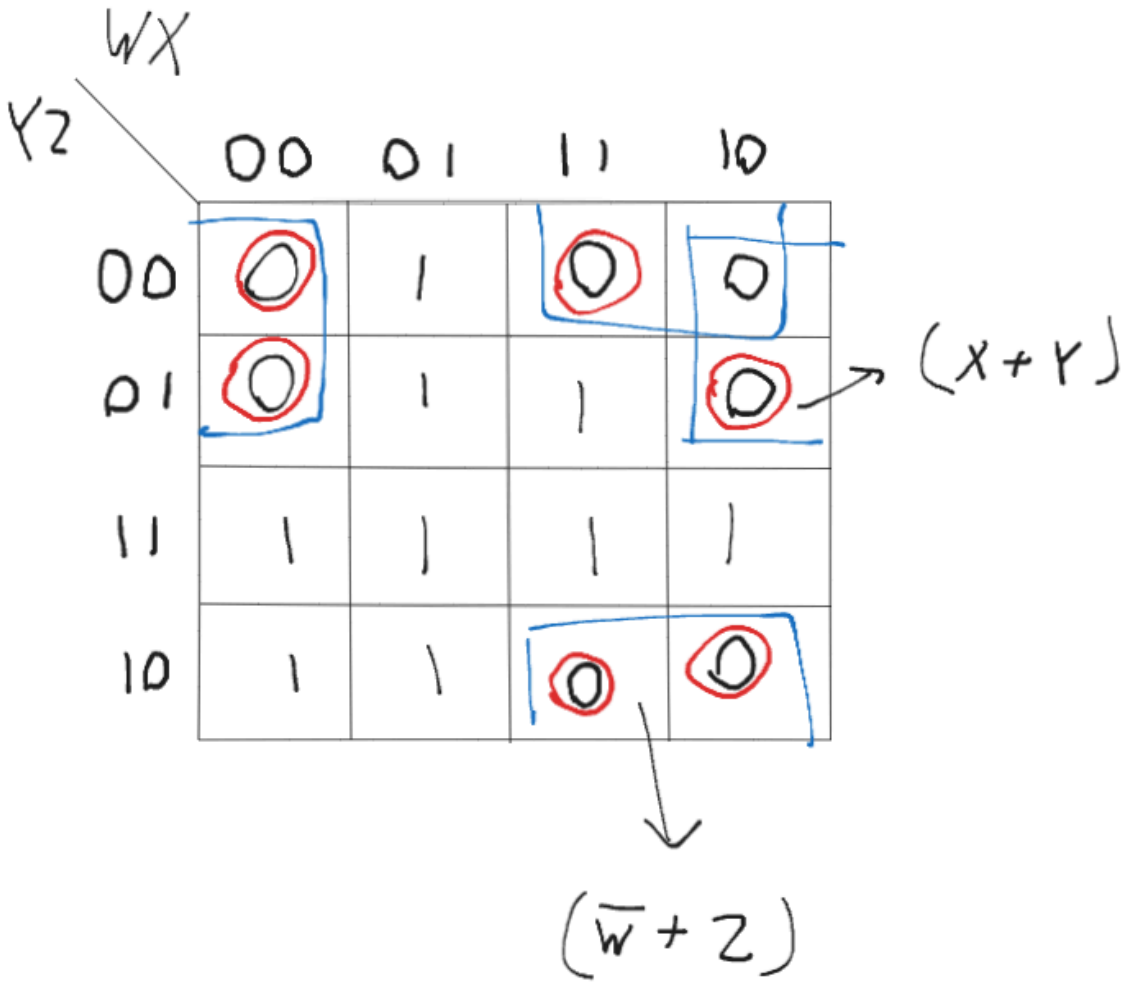
b) **F = X~Y + W~Y + X~Z + W~Z + ~YZ + ~WX**



Minimum-cost SOP is F = ~YZ + ~WX + X~Z

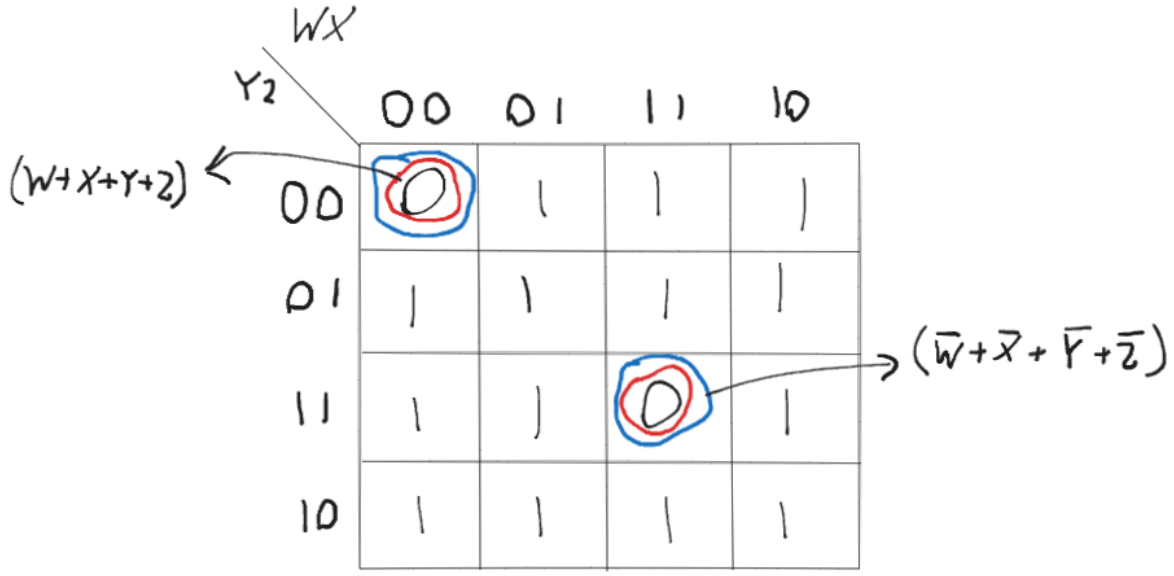
4.2 - Using Karnaugh maps, find minimal product-of-sums expressions for the following logic functions. Indicate the distinguished 0-cells in each one that you do.

a) **F = ΠW, X, Y, Z(0, 1, 8, 9, 10, 12, 14)**



Minimum-cost POS is F = (~W + Z)(X + Y)

b) F = ΠW, X, Y, Z(0, 15)

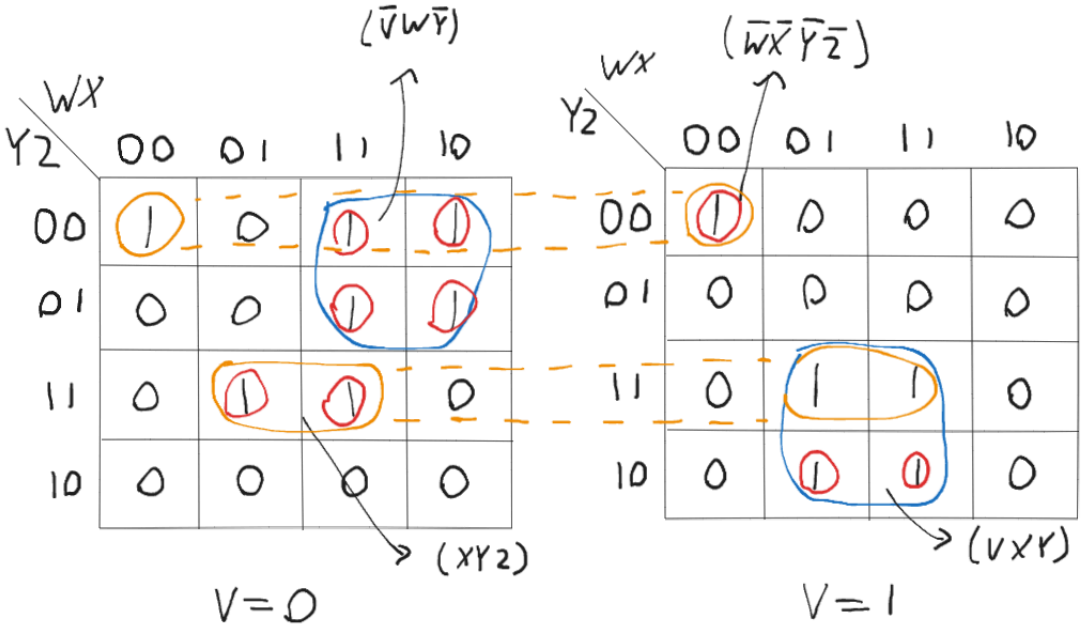


Minimum-cost POS is F = (W + X + Y + Z)(~W + ~X + ~Y + ~Z)

4.3 Find a minimal sum of products expression for the following logic function using a 5-variable K-map.

$$F = \sum_{V,W,X,Y,Z} (0,7,8,9,12,13,15,16,22,23,30,31)$$

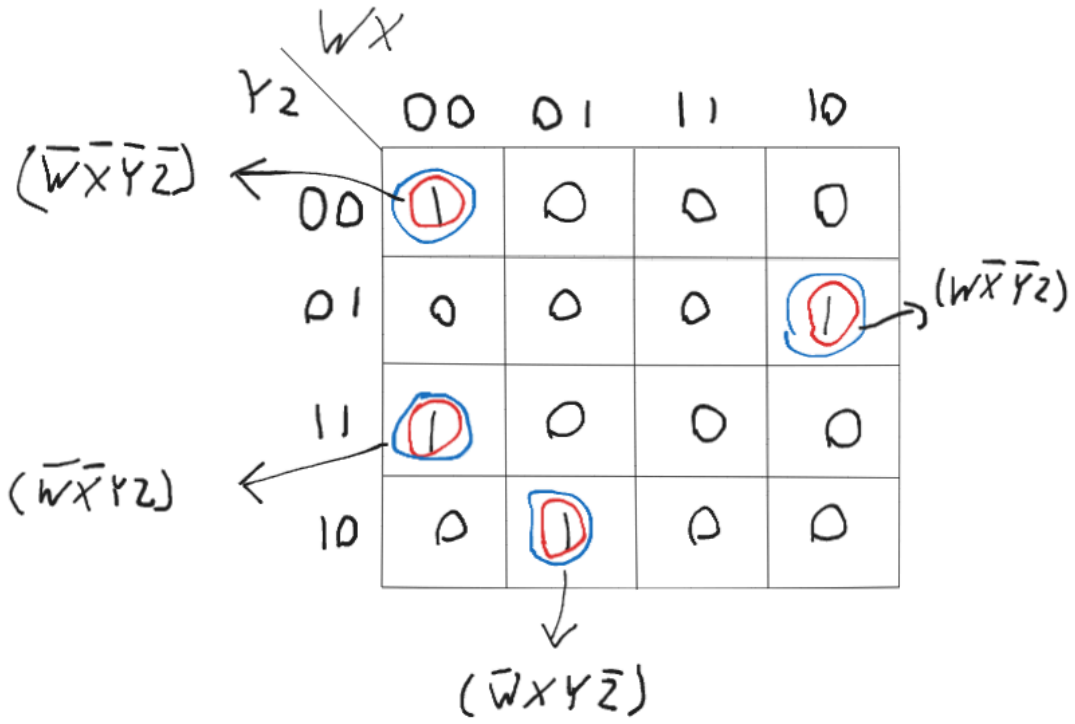
Orange circles and dotted lines mean that the prime implicants are connected between boards (V doesn't matter).



Minimum-cost SOP is F = (~VW~Y) + (XYZ) + (~W~X~Y~Z) + (VXY)

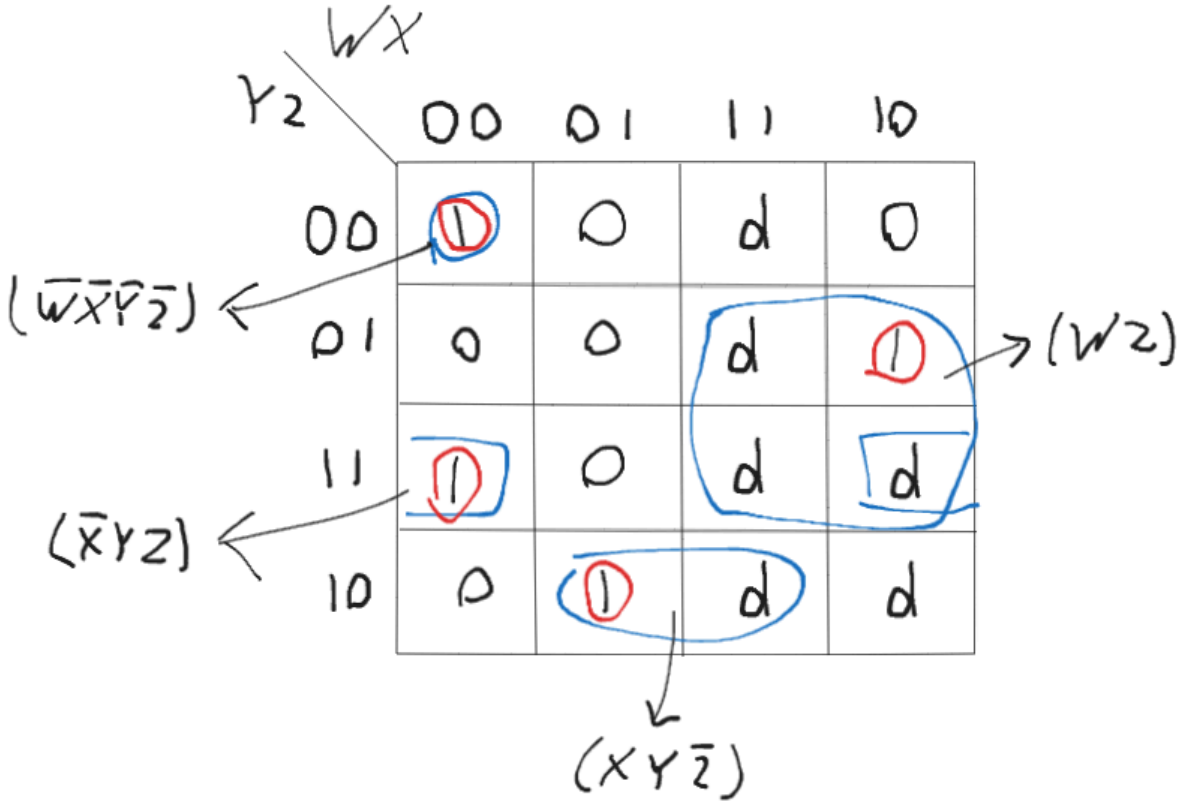
4.4 K-map "Don't Care"s

a) **Using Karnaugh maps, find a minimal sum-of-products expression for a BCD evenly divisible-by-3 detector NOT using don't-cares, represented by F = ΣW, X, Y, Z m(0,3,6,9)**



Minimum-cost SOP not using don't-cares is F = (~W~X~Y~Z) + (~W~XYZ) + (W~X~YZ) + (~WXY~Z)

b) Using Karnaugh maps, find a minimal sum-of-products expression for a BCD evenly divisible-by-3 detector using don't-cares, represented by F = ΣW, X, Y, Z m(0,3,6,9) + ΣD(10,11,12,13,14,15).



Minimum-cost SOP using don't-cares is F = (~W~X~Y~Z) + (~XYZ) + (XY~Z) + (WZ)

2 - Lab Procedures

6.1 Three-Way Light Control using structural Verilog

i) Write the structural (i.e., using gate primitives) Verilog HDL code for a three-way light control (see Section 3.2) with the module interface as shown below, implemented as a sum-of-products. Simulate in the Vivado with testbench to verify operation.

The SOP is the three-way light control is: f = (~x1 + ~x2 + x3)(~x1 + x2 + ~x3)(x1 + ~x2 + ~x3)(x1 + x2 + x3)

ii) You should include a Verilog main code, testbench, and picture of the timing diagram in your lab report

Filename: ThreeWayLight_structural.v

```
module ThreeWayLight_structural(x1, x2, x3, light);
        input x1, x2, x3;
        output light;

        wire w1, w2, w3, w4, n1, n2, n3;

        not(n1, x1);
        not(n2, x2);
        not(n3, x3);

        or(w1, n1, n2, x3);
        or(w2, n1, x2, n3);
```

```
        or(w3, x1, n2, n3);
        or(w4, x1, x2, x3);

        and(f, w1, w2, w3, w4);
endmodule
```

Filename: threewaylight_tb.v

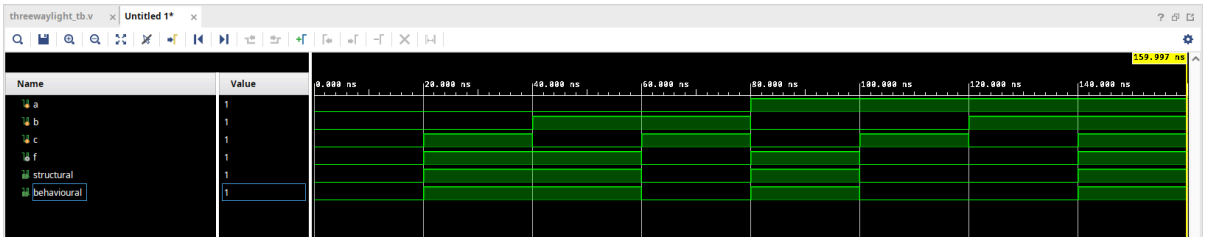
```
module threewaylight_tb();

        reg a, b, c;
        wire f;
        ThreeWayLight_structural vt(a, b, c, f);
        ThreeWayLight_behavioural ut(a, b, c, f);

        initial begin
                {a,b,c} = 3'd0; #20;
                {a,b,c} = 3'd1; #20;
                {a,b,c} = 3'd2; #20;
                {a,b,c} = 3'd3; #20;
                {a,b,c} = 3'd4; #20;
                {a,b,c} = 3'd5; #20;
                {a,b,c} = 3'd6; #20;
                {a,b,c} = 3'd7; #20;
                $display("Done");
                $finish;
        end

endmodule
```

Waveform (used same testbench for both):



iii) Fill out the truth table with the result from the timing diagram and comment if the waveform is as expected.

x1	**x2**	**x3**	**f**
0	0	0	0
0	0	1	1
0	1	0	1

0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Waveform is working as expected.

6.2 Three-Way Light Control using behavioural Verilog

ii) You should include a Verilog main code, testbench, and picture of the timing diagram in your lab report

Filename: ThreeWayLight_behavioural.v

```
module ThreeWayLight_behavioural(x1, x2, x3, light);

    input x1, x2, x3;
    output light;

    wire w1, w2, w3, w4, n1, n2, n3;

    assign n1 = ~x1;
    assign n2 = ~x2;
    assign n3 = ~x3;

    assign w1 = (x1 | n2 | n3);
    assign w2 = (n1 | x2 | n3);
    assign w3 = (n1 | n2 | x3);
    assign w4 = (x1 | x2 | x3);

    assign light = (w1 & w2 & w3 & w4);

endmodule
```

Note: you could do the entire SOP in one line but I think this is more readable.

Filename: threewaylight_tb.v

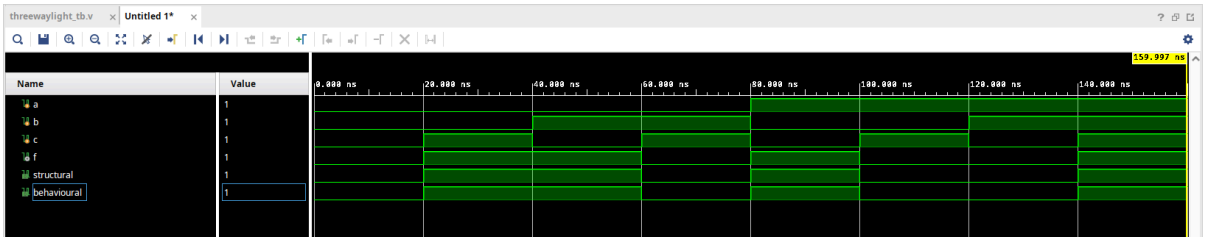
```
module threewaylight_tb();

    reg a, b, c;
    wire f;
    ThreeWayLight_structural vt(a, b, c, f);
    ThreeWayLight_behavioural ut(a, b, c, f);

    initial begin
    {a,b,c} = 3'd0; #20;
```

```
        {a,b,c} = 3'd1; #20;
        {a,b,c} = 3'd2; #20;
        {a,b,c} = 3'd3; #20;
        {a,b,c} = 3'd4; #20;
        {a,b,c} = 3'd5; #20;
        {a,b,c} = 3'd6; #20;
        {a,b,c} = 3'd7; #20;
        $display("Done");
        $finish;
        end
endmodule
```

Waveform (used same testbench for both):



iii) Fill out the truth table with the result from the timing diagram and comment if the waveform is as expected.

x1	x2	x3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Waveform is working as expected.