CSE460: VLSI Design

Name- Ms Rodsy Tahmid
ID- 20101021
Section- 07

Lab Assignment 1

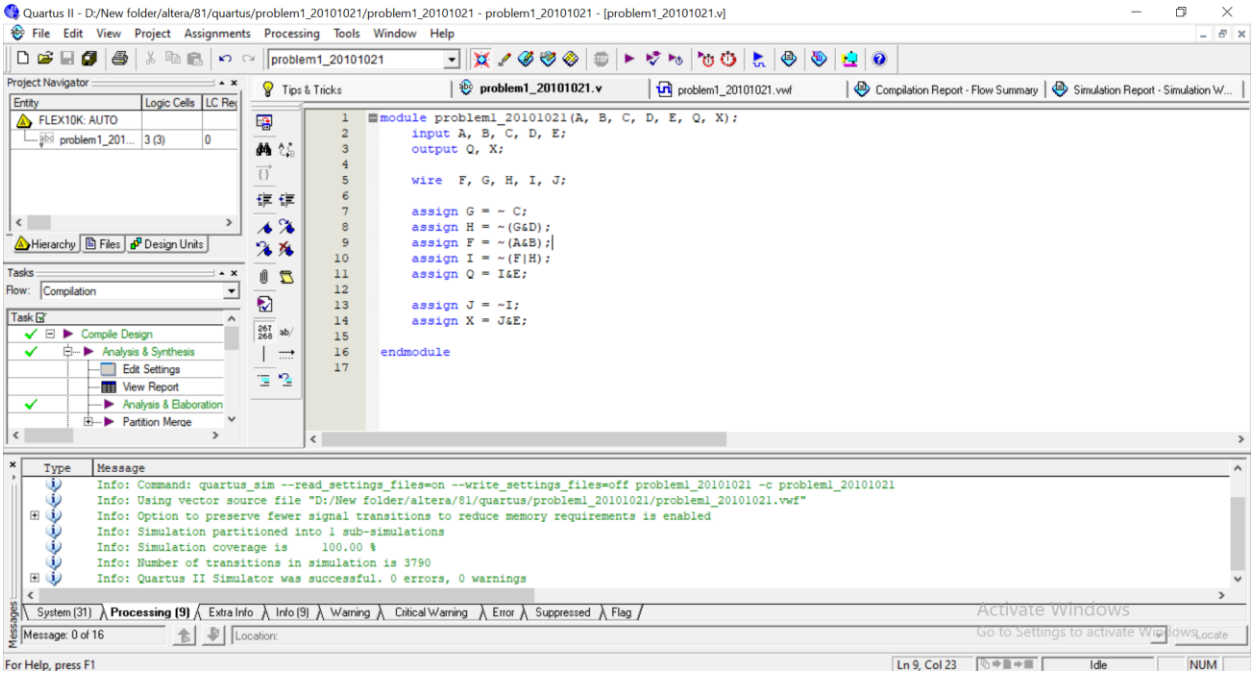
Problem 1:

1. Code:

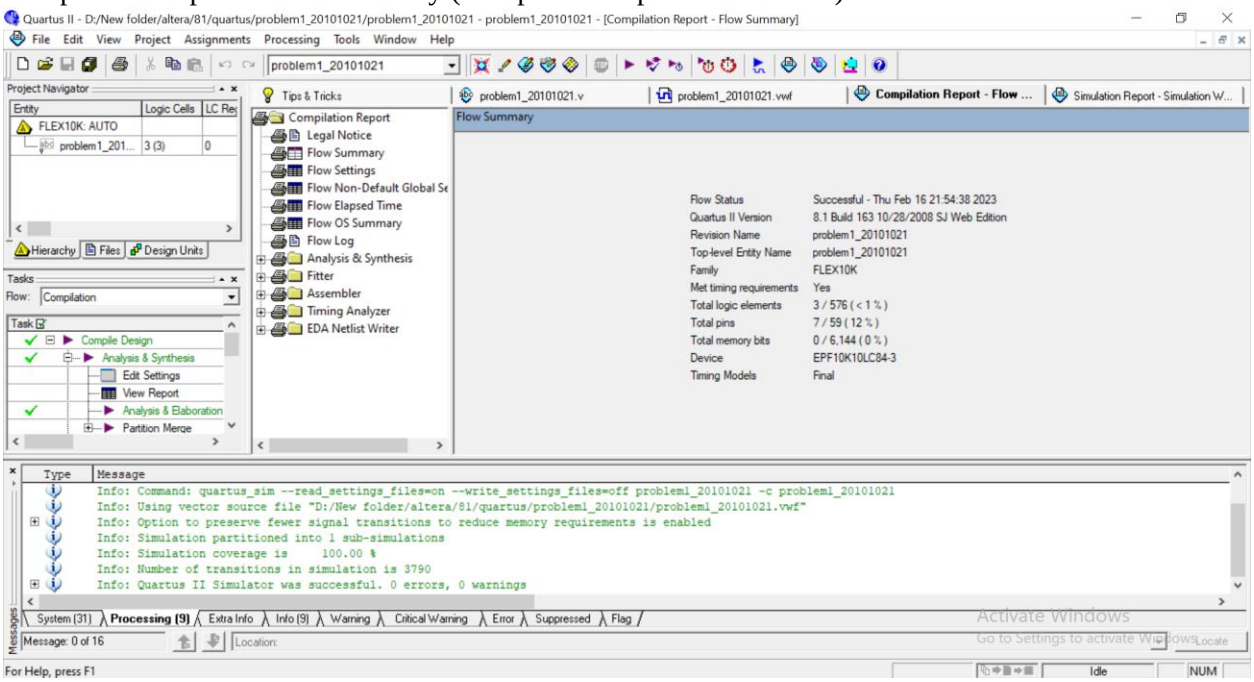
```
module problem1_20101021(A, B, C, D, E, Q, X);
        input A, B, C, D, E;
        output Q, X;
        wire  F, G, H, I, J;
        assign G = ~ C;
        assign H = ~(G&D);
        assign F = ~(A&B);
        assign I = ~(F|H);
        assign Q = I&E;t
        assign J = ~I;
        assign X = J&E;
endmodule
```

2. Output:

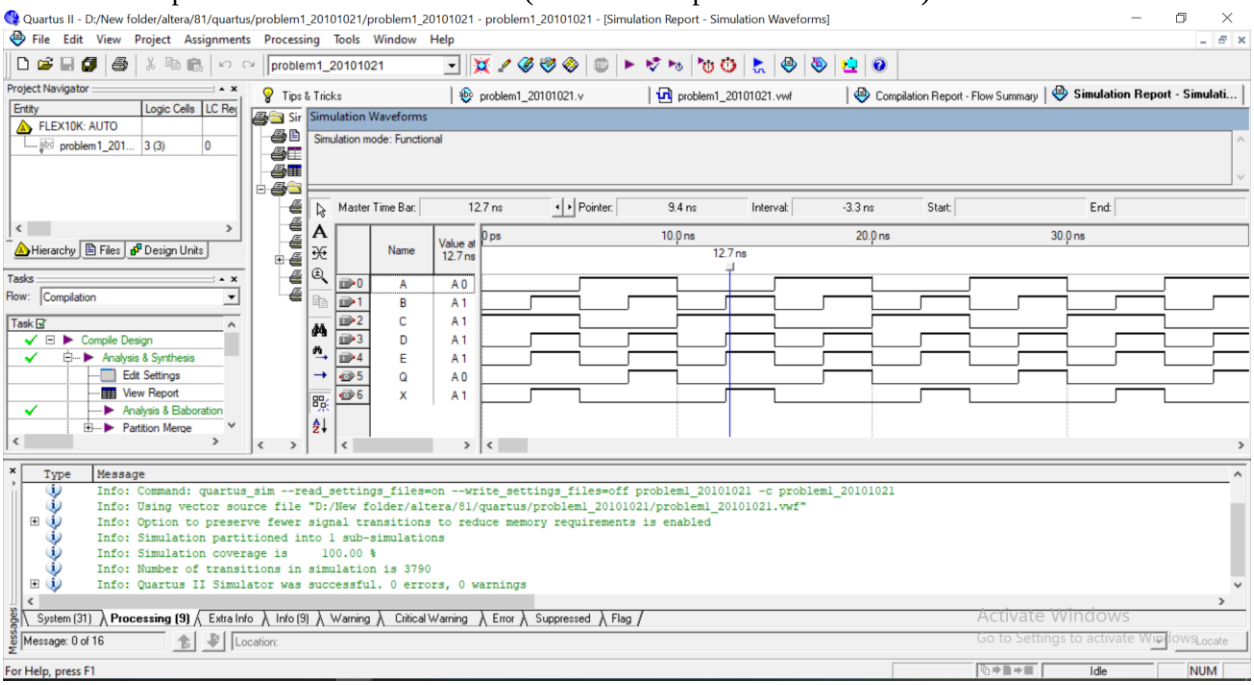
Verilog code:



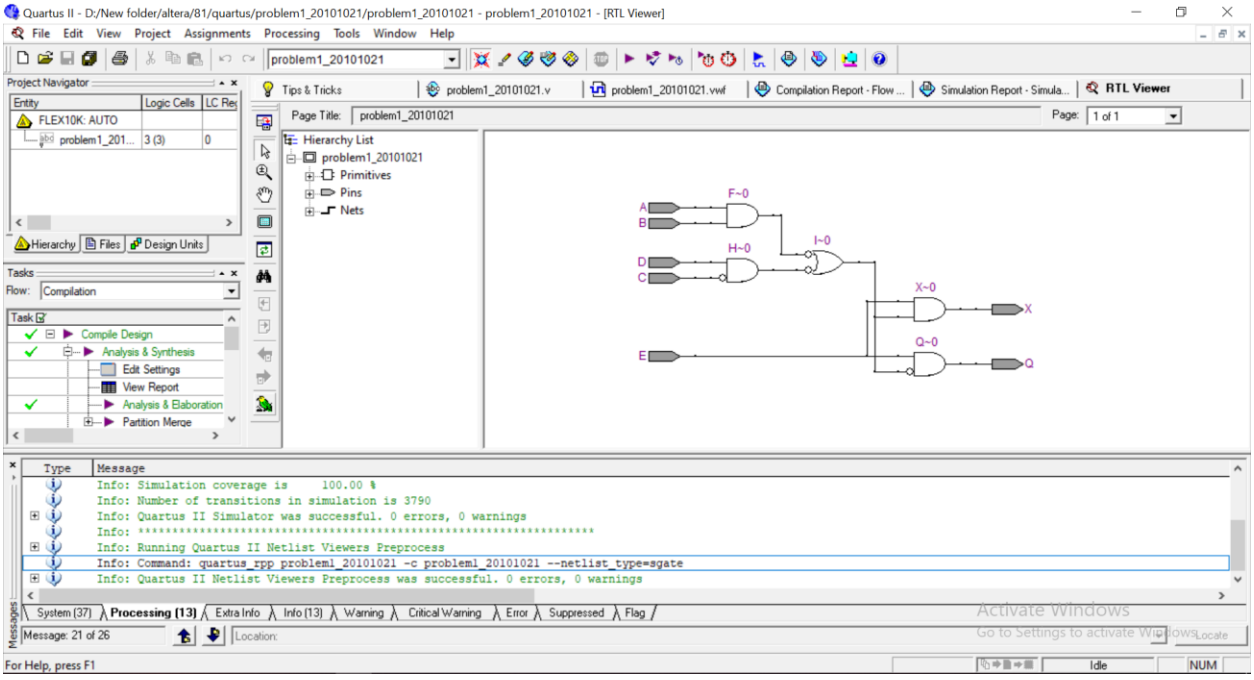
Compilation Report - Flow Summary (Compilation report of the .v file):



Simulation Report - Simulation Waveforms (Simulation report of the .vwf file):



RTL Viewer:

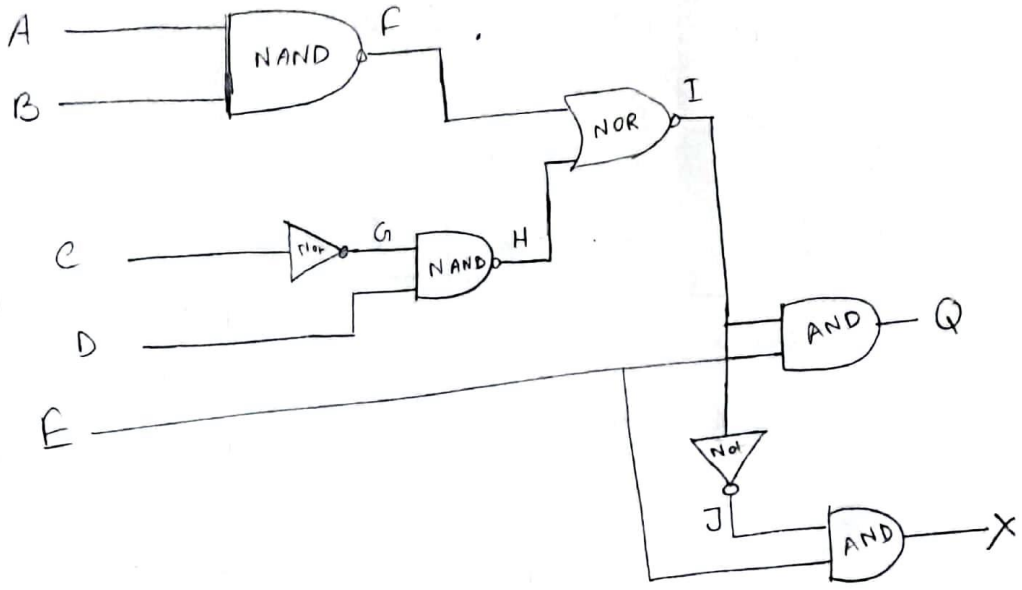


3. Explanation:

Ms Rodry Tahmid
Id: 20101021

Lab- 1
Problem - 1

Explanation:



Truth Table:

A	B	F	C	G	D	H	I (F NOR H)	J
0	0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0	1
1	0	1	0	1	0	1	0	1
1	1	0	0	1	1	0	1	0

Id : 20101021

Truth Table (continuation):

Output1 ↑ Output2 →

I	E	Q	J	F	X
0	0	0	1	0	0
0	1	0	1	1	1
0	0	0	1	0	0
1	1	1	0	1	0

$\overline{A.B} = F$	$I.E = Q$
$\overline{C} = G$	$\overline{I} = J$
$\overline{G.D} = H$	$J.E = X$
$\overline{F+H} = I$	

$Q = I \,\&\, E$

$= (F \text{ NOR } H) \,\&\, E$

$Q = (\sim (F | H)) \,\&\, E$

$F = \sim (A \,\&\, B)$

$H = \sim (G \,\&\, D)$

$G = \sim C$

$X = J \,\&\, E$

$J = \sim I$

NOR Verilog

$Y = \sim (A | B)$

↓ Not + ↓ OR

NAND Verilog

↓

Not + AND

$Y = \sim (A \,\&\, B)$

↓ Not + AND

5 inputs $2^5 = 32$ combinations.

$$Q = \sim\left(\sim(A \& B) \mid \sim(G \& D)\right) \& E$$

$$Q = \sim\left(\sim(A \& B) \mid \underbrace{\sim((\sim C) \& D)}_{I}\right) \& E.$$

$$X = (\sim I) \& E.$$

$$X = \sim\left(\sim\left(\sim(A \& B) \mid \sim((\sim C) \& D)\right)\right) \& E.$$

In this diagram,

1. First Half Cycle: $A = B = 0$, $\overline{A \cdot B} = 1 = F$

$G = \overline{C} = 0$, $\overline{G \cdot D} = 1 = H$, $\overline{F + H} = 0 = I$.

$\boxed{I \cdot E = 0 = Q}$, $\overline{I} = 1 = J$, $\boxed{J \cdot E = 0 = X}$

According to problem-1, $\overline{A \cdot B}$ and $\overline{\overline{C} \cdot D}$ are connected to NOR gate, $\overline{F + H}$ and E are connected to AND forming Q, $\overline{\overline{F+H}} = \underbrace{F + H}_{J}$ and E are connected to AND forming X.

So, $Q = I \cdot E$

$X = \overline{I} \cdot E$

Id: 20101021

Truth Table:

	A	B	F $= \overline{A.B}$	C	G $= \overline{C}$	D	H $= \overline{G.D}$	I $= \overline{F+H}$	J $= \overline{I}$	E	Output 1: Q $= I.E$	Output 2: X $= J.E$
1st Half Cycle	0	0	1	1	0	0	1	0	1	0	0	0
2nd Half Cycle	0	1	1	1	0	1	1	0	1	1	0	1
3rd Half Cycle	1	0	1	0	1	0	1	0	1	0	0	0
4th Half Cycle	1	1	0	0	1	1	0	1	0	1	1	0

Id : 20101021

2. Second Half Cycle:

$A = 0$, $B = 1$, $C = 1$, $D = 1$, $E = 1$

$\overline{A \cdot B} = 1$, $\overline{C} = 0$, $\overline{C \cdot D} = 1$ $\overline{\overline{A \cdot B} + H} = 0$

$\overline{\overline{A \cdot B} + H} = \overline{A \cdot B} + H = 1$, So $Q = 0$ & $X = 1$

3. Third Half Cycle:

$A = 1$, $B = 0$, $C = 0$, $D = 0$, $E = 0$

$\overline{A \cdot B} = 1$ $\overline{C} = 1$, $\overline{C \cdot D} = 1$, $\overline{\overline{A \cdot B} + H} = 0$

$\overline{\overline{A \cdot B} + H} = \overline{A \cdot B} + H = 1$, So, $Q = 0$, & $X = 0$.

4. Fourth Half Cycle:

$A = 1$, $B = 1$, $C = 0$, $D = 1$, $E = 1$

$\overline{A \cdot B} = 0$, $\overline{C} = 1$, $\overline{C \cdot D} = 0$, $\overline{\overline{A \cdot B} + H} = 1$

$\overline{\overline{A \cdot B} + \overline{C \cdot D}} = \overline{A \cdot B} + \overline{C \cdot D} = 0$

Therefore, $Q = 1$ and $X = 0$.

In this way, this pattern is continued and this circuit continued to show outputs.

There are 2^5 = 32 combinations as we can see from the question there are 5 inputs.

We are told to perform the task in two ways one is structural and another one is behavioral.

The first one is structural and the second one is behavioral. For getting 2^5=32 combinations for input we set the clock value of a, b, c, d, e by doubling their value from a to e.

Like input a has clock period of 10ns so input b has 20ns, input c has 40ns and so on. Now if we look the first simulation picture which is for structural representation, we can see in the 2nd half of the clock period 50ns – 60ns output q = 1 and x = 0. Now if we look the second simulation picture which is for behavioral representation we can see in the 2nd half of the clock period 50ns – 60ns output q = 1 for q = (~(~(a & b)| ~ (~ c & d)) | e); and x = 0 for x = ((~(a & b)| ~ (~ c & d)) & e). In the both case structural and behavioral we are getting the same output q = 1 and x = 0.

For verification we will use the equation of q and x which we have used in behavioral code.

Verification for q=1. From the timing diagram a=b=d=1 and c=e=0. Now,

q= (~ (~ (a & b| ~ (~c & d)) | e)
Or, q= (~ (~ (1 & 1) | ~ (~0 & 1)) | 0)
Or, q= (~ (~ 1 | ~ (1 & 1)) | 0)
Or, q= (~ (~ 1 | ~ 1) | 0)
Or, q= (~ (0 | 0) | 0)
Or, q= (~ 0 | 0)
Or, q= (1| 0)
Or, q= 1
Verification for x=0. From the timing diagram a=b=d=1 and c=e=0. Now,
x= ((~ (a & b| ~ (~c & d)) & e)
Or, x= ((~ (1 & 1) | ~ (~0 & 1)) & 0)
Or, x= ((~ 1 | ~ (1 & 1)) & 0)
Or, x= ((~ 1 | ~ 1) & 0)
Or, x= ((0 | 0) & 0)
Or, x= (0 & 0)
Or, x=0

So, our code is correct as theoretical output matches with timing diagram's output.

Truth Table of 32 combinations:

	A	B	C	D	E	F	G	H	I	J		
1	0	0	0	0	0	1	1	1	0	1	0	0
2	0	0	0	0	1	1	1	1	0	1	0	1
3	0	0	0	1	0	1	1	0	0	1	0	0
4	0	0	0	1	1	1	1	0	0	1	0	1
5	0	0	1	0	0	1	0	1	0	1	0	0
6	0	0	1	0	1	1	0	1	0	1	0	1
7	0	0	1	1	0	1	0	1	0	1	0	0
8	0	0	1	1	1	1	0	1	0	1	0	1
9	0	1	0	0	0	1	1	1	0	1	0	0
10	0	1	0	0	0	1	1	1	0	1	0	0
11	0	1	0	1	1	1	1	0	0	1	0	1
12	0	1	0	1	0	1	1	0	0	1	0	0
13	0	1	1	0	1	1	0	1	0	1	0	1
14	0	1	1	0	0	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1	0	1
16	0	1	1	1	0	1	0	1	0	1	0	0
17	1	0	0	0	1	1	1	1	0	1	0	1
18	1	0	0	0	0	1	1	1	0	1	0	0
19	1	0	0	1	1	1	1	0	0	1	0	1
20	1	0	0	1	0	1	1	0	0	1	0	0
21	1	0	1	0	1	1	0	1	0	1	0	1
22	1	0	1	0	0	1	0	1	0	1	0	0
23	1	0	1	1	1	1	0	1	0	1	0	1
24	1	0	1	1	0	1	0	1	0	1	0	0
25	1	1	0	0	1	0	1	1	0	1	0	1
26	1	1	0	0	0	0	1	1	0	1	0	0
27	1	1	0	1	1	0	1	0	1	0	1	0
28	1	1	0	1	0	0	1	0	1	0	0	0
29	1	1	1	0	1	0	0	1	0	1	0	1
30	1	1	1	0	0	0	0	1	0	1	0	0
31	1	1	1	1	1	0	0	1	0	1	0	1
32	1	1	1	1	0	0	0	1	0	1	0	0
33												

Problem 2:

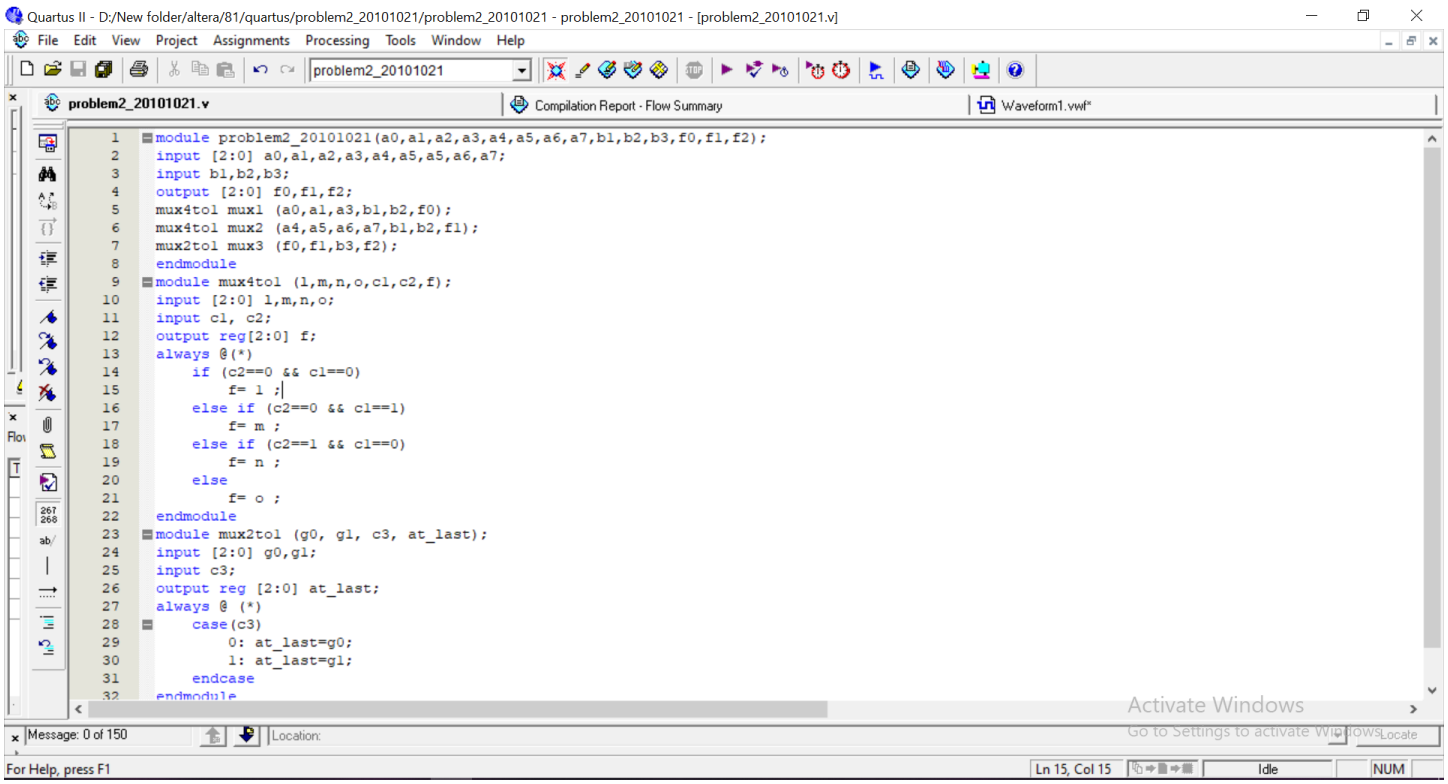
1. Code:

```
module problem2_20101021(a0,a1,a2,a3,a4,a5,a6,a7,b1,b2,b3,f0,f1,f2);
input [2:0] a0,a1,a2,a3,a4,a5,a5,a6,a7;
input b1,b2,b3;
output [2:0] f0,f1,f2;
mux4to1 mux1 (a0,a1,a3,b1,b2,f0);
mux4to1 mux2 (a4,a5,a6,a7,b1,b2,f1);
mux2to1 mux3 (f0,f1,b3,f2);
endmodule
module mux4to1 (l,m,n,o,c1,c2,f);
input [2:0] l,m,n,o;
input c1, c2;
output reg[2:0] f;
always @(*)
	if (c2==0 && c1==0)
		f= 1;
	else if (c2==0 && c1==1)
		f= m;
	else if (c2==1 && c1==0)
		f= n;
	else
		f= o ;
endmodule
module mux2to1 (g0, g1, c3, at_last);
input [2:0] g0,g1;
input c3;
output reg [2:0] at_last;
always @ (*)
	case(c3)
		0: at_last=g0;
```

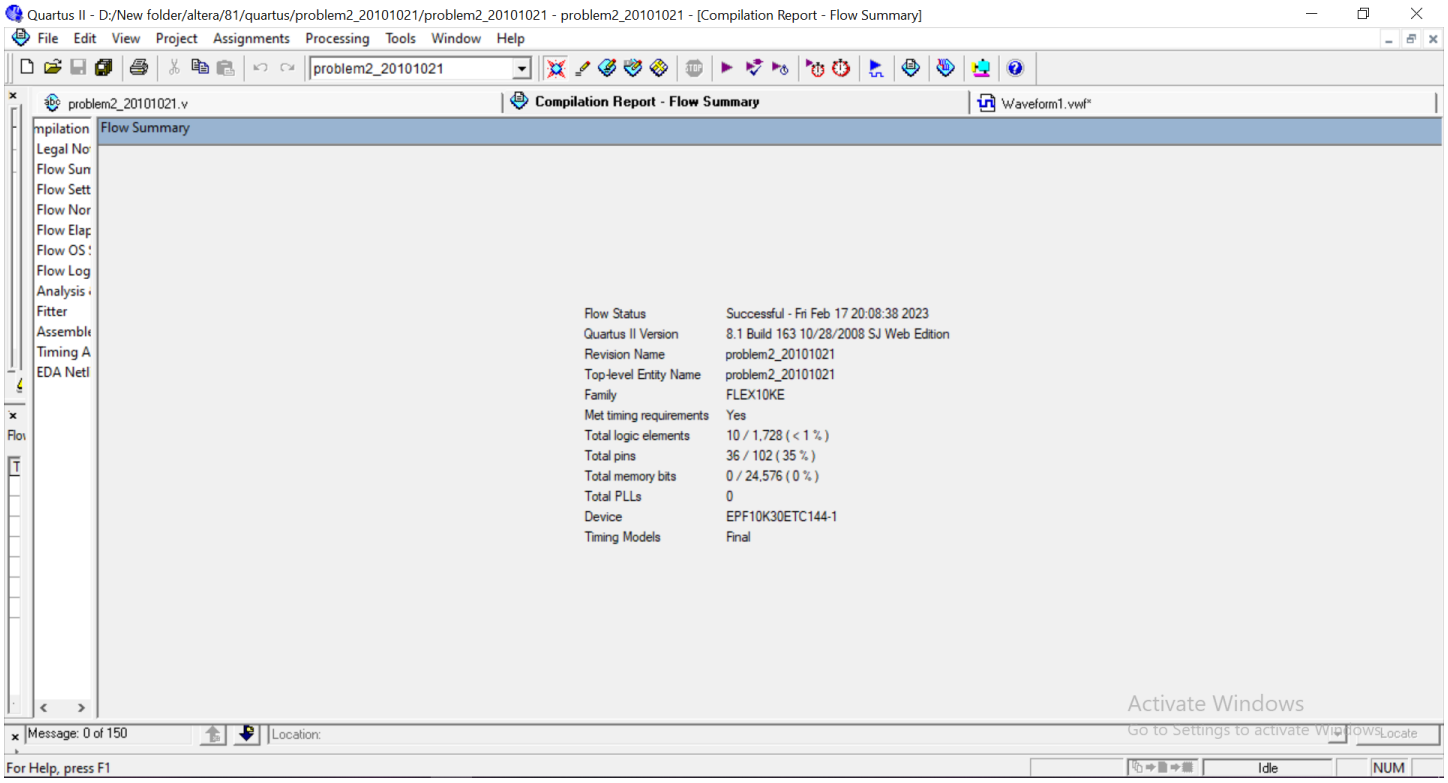
```
            1: at_last=g1;
        endcase
endmodule
```

2. Output:

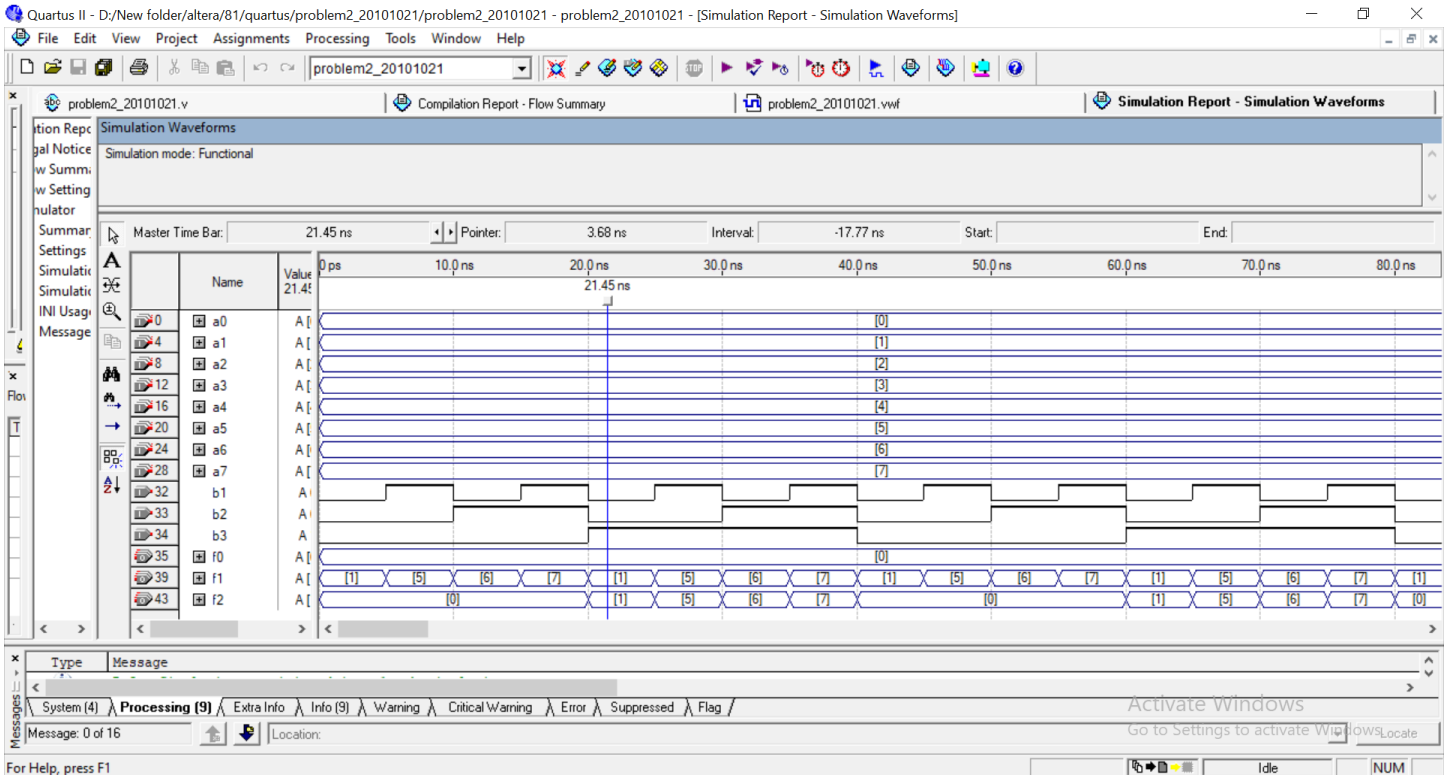
○ Verilog code:



○ Compilation Report - Flow Summary (Compilation report of the .v file):



○ Simulation Report - Simulation Waveforms (Simulation report of the .vwf file):



3. Explanation:

According to the question, an 8-to-1 mux is designed using one 2-to-1 mux and two 4-to-1 mux sub-circuits.

We can see all the inputs (a0, a1, …. a7) have a decimal value sequentially (0-7) and they are absolutely in 3 bits binary value from the timing diagram.
And the outputs are shown as f0, f1, f3 in the timing diagram. The other 3 inputs, b1, b2, b3, are selectors and they represent clock pulses in the timing diagram and help to produce the output.
For instance, from the timing diagram if we notice 1st half of the clock period 10-20ns here b3=0, b2=1, b1=0 means the binary is 010 which is 2 in decimal.
It means the selector is 2 here so in the timing diagram we can see the final output f2 is 2. Again, for the 2nd half of the clock period 10-20ns b3=0, b2=1, b1=1 means the binary is 011 which is 3 in decimal.
It means the selector is 3 here so in the timing diagram we can see the final output f2 is 3.
And the rest of the timing diagram produces output f2 in this behaviour.

Lab Assignment 2

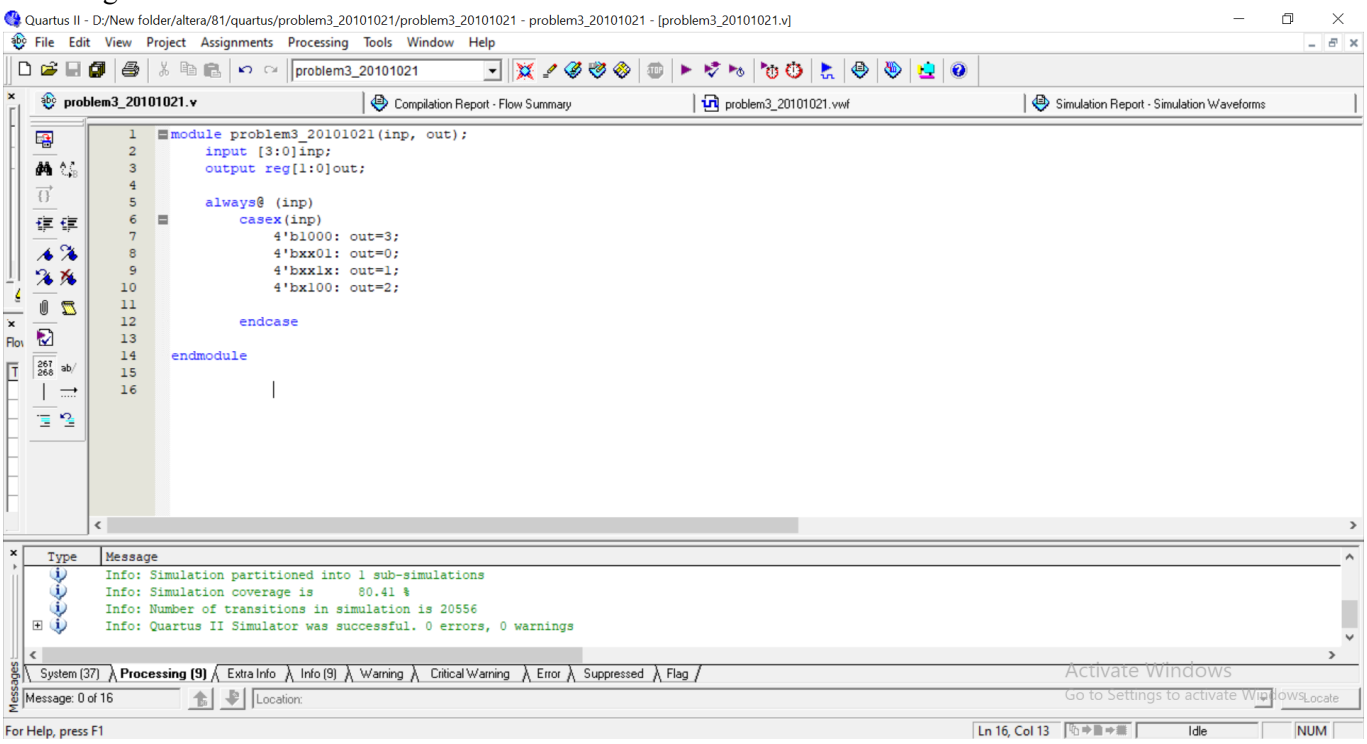
Problem 1:

1. Code:

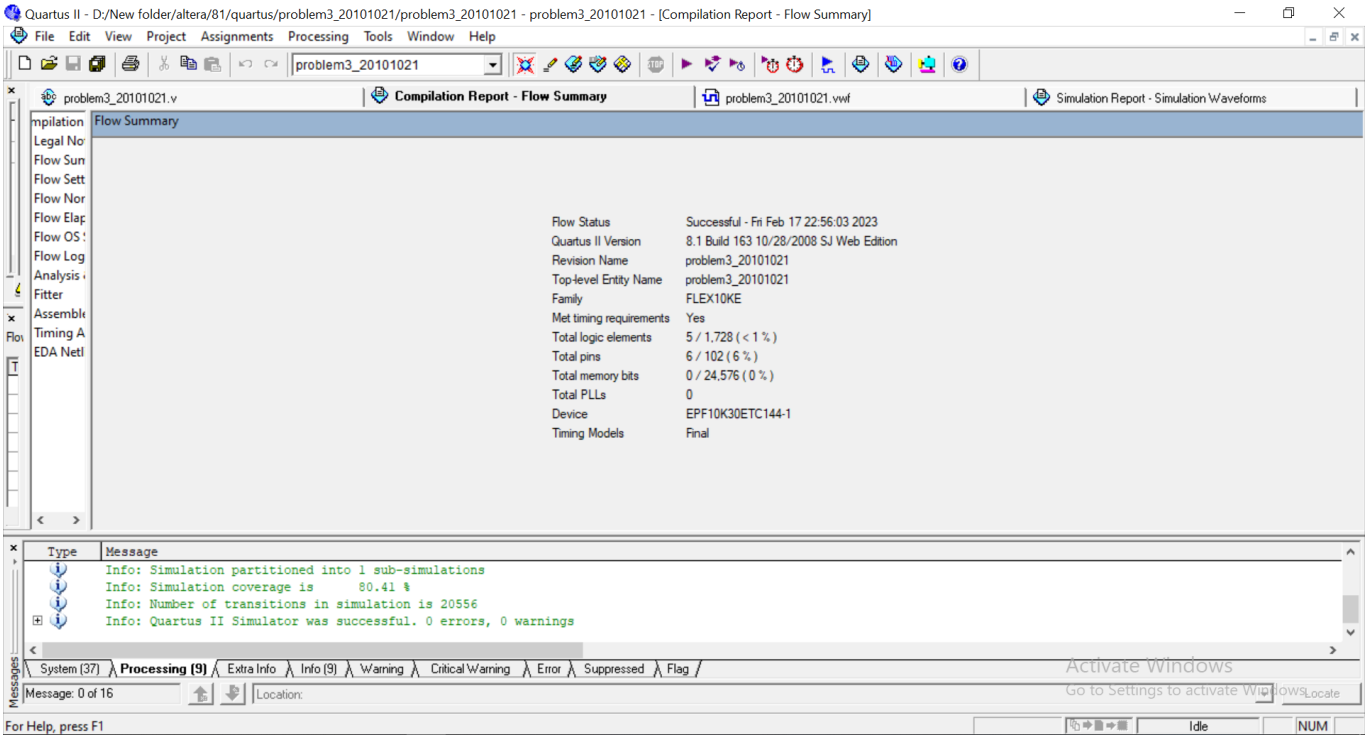
```
module problem3_20101021(inp, out);
        input [3:0]inp;
        output reg[1:0]out;

        always@ (inp)
                casex(inp)
                        4'b1000: out=3;
                        4'bxx01: out=0;
                        4'bxx1x: out=1;
                        4'bx100: out=2;

                endcase

endmodule
```

2. Output:

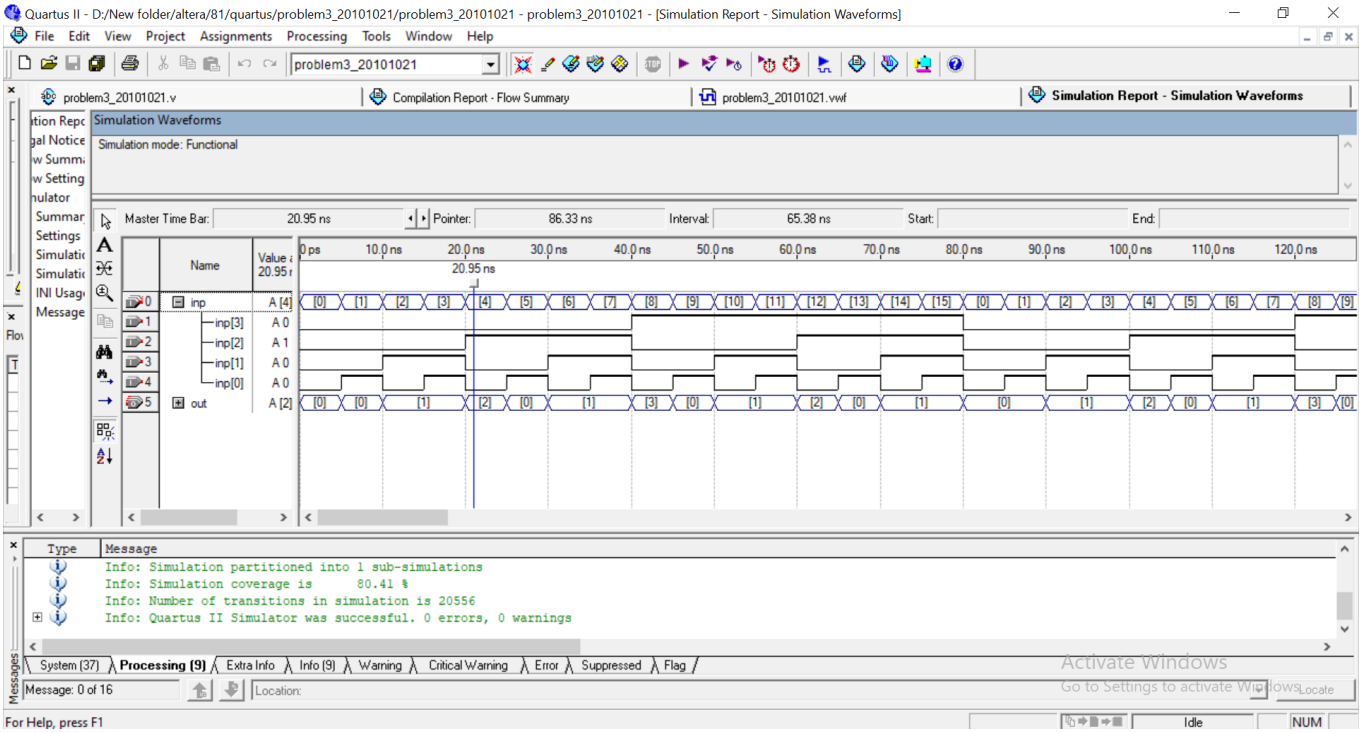
○ Verilog code



○ Compilation Report - Flow Summary (Compilation report of the .v file)



○ Simulation Report - Simulation Waveforms (Simulation report of the .vwf file)



3. Explanation:

Here, a priority encoder with a priority of w[3]>w[0]>w[1]>w[2] with ‘inp’ as the input and ‘out’ as the output.

In the simulation, we see that the encoder works by giving priority to w[3] before anything else means w[3] has the highest priority so when it will be kept set output ‘out’ will produce 1 as output.

When all other are off means 0, then the encoder chooses w[2] as it has the lowest priority.

The priority goes w[3] then w[0] then w[1] and w[2].

From the simulation we can see encoder outputs ‘0’ when input inp[3] is kept 0 and inp[0] is kept 1, and other 2 cycles for inp[1], inp[2] will be considered as don’t care here as they have lower priority than w[3].

And to produce ‘2’ as output from the simulation we can see inp[3], inp[0] are kept low as they have higher priority than inp[1] and inp[2] is don’t care here as it has lower priority. And this is how the entire simulation works.