CSE460 Lab Assignment 1

Name : Abdullah Hasan Sajjad Rafi
ID : 19301097
Section : 5

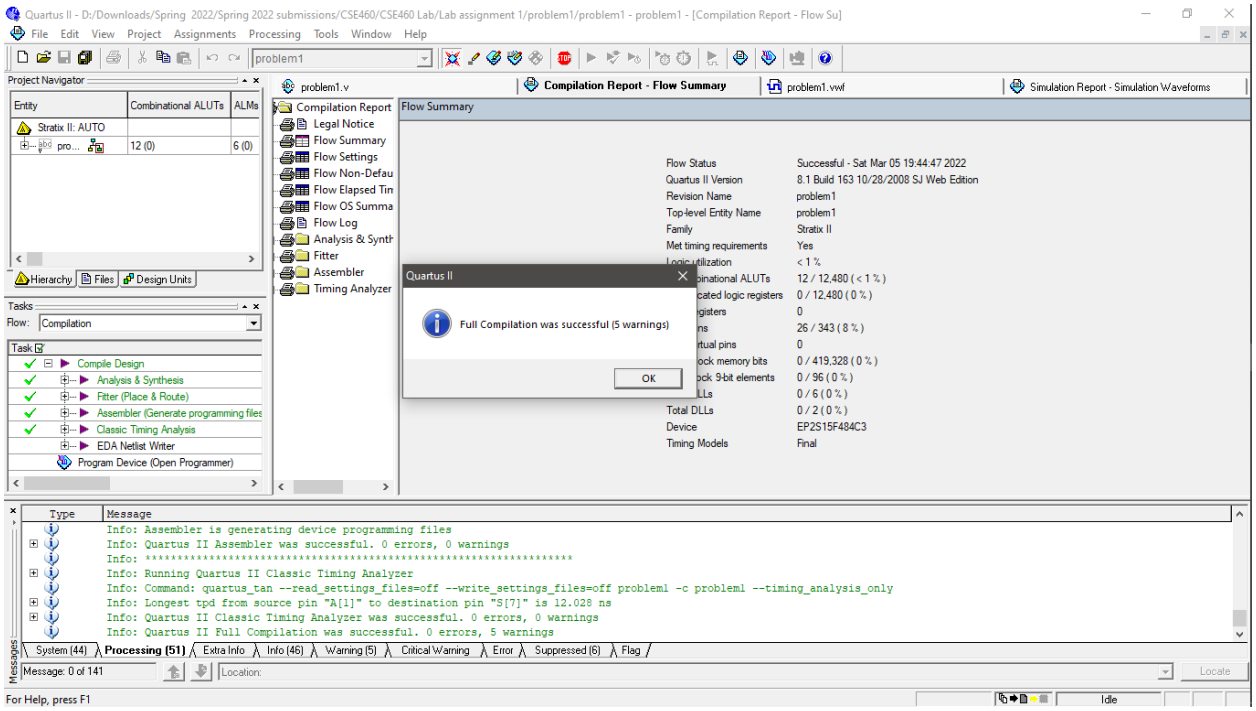
Problem 1 : Design an 8 bit full adder using 1 bit full adder modules and add these two binary numbers: 8'b10110101 and 8'b10100111

Answer :

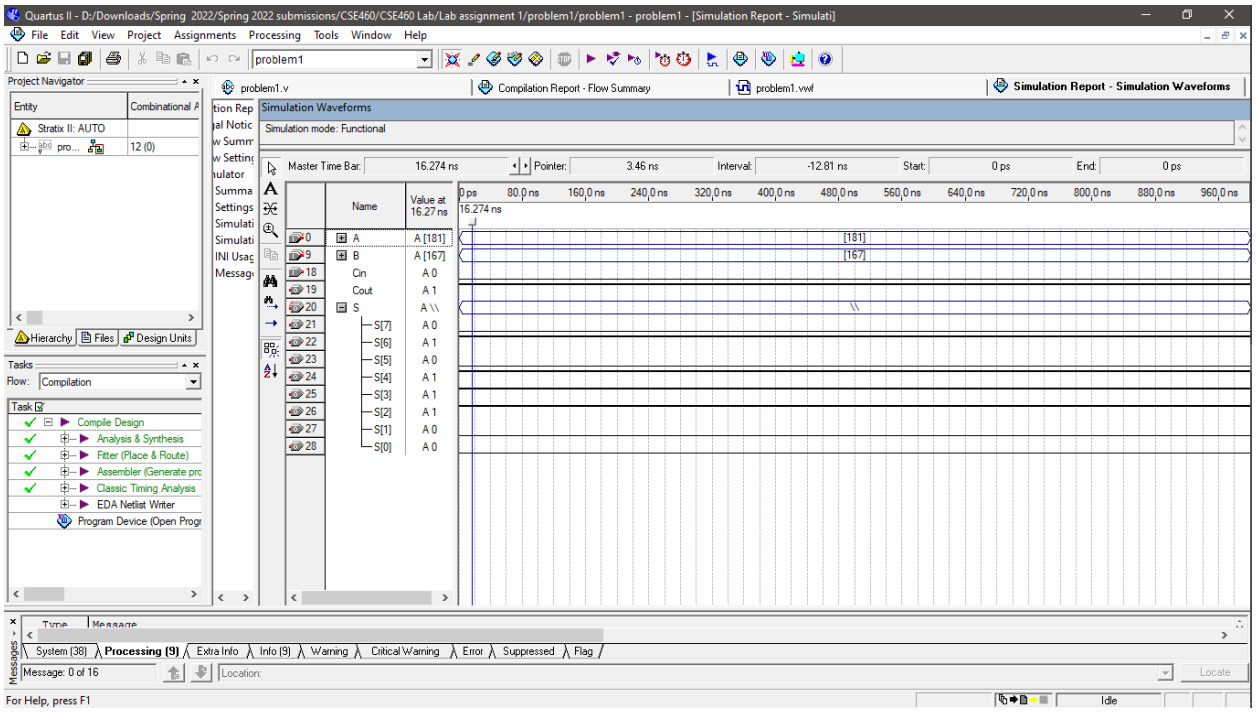
Code :

```
module problem1(S,Cout, A, B, Cin);

    input Cin;
    input [7:0]A,B;

    output [7:0]S;
    output Cout;
    wire Cout0, Cout1, Cout2, Cout3, Cout4, Cout5, Cout6;

    // A = 8'b10110101
    // B = 8'b10100111

    fulladd stage0 (S[0], Cout0, A[0], B[0], Cin);
    fulladd stage1 (S[1], Cout1, A[1], B[1], Cout0);
    fulladd stage2 (S[2], Cout2, A[2], B[2], Cout1);
    fulladd stage3 (S[3], Cout3, A[3], B[3], Cout2);
    fulladd stage4 (S[4], Cout4, A[4], B[4], Cout3);
    fulladd stage5 (S[5], Cout5, A[5], B[5], Cout4);
    fulladd stage6 (S[6], Cout6, A[6], B[6], Cout5);
    fulladd stage7 (S[7], Cout,  A[7], B[7], Cout6);

endmodule


// This module implements a 1-bit full adder

module fulladd(sum, carryout, x, y, carryin);

   input x, y, carryin;
   output sum, carryout;

   assign sum = x ^ y ^ carryin;
   assign carryout = (x & y) | (carryin & (x ^ y));

endmodule
```

Compilation report :



Simulation report :



Brief description :

Problem-01

In the problem 1 we take two 8 bit binary numbers and add these two binary numbers. The full adder circuit is designed using 1 bit full adders modules.

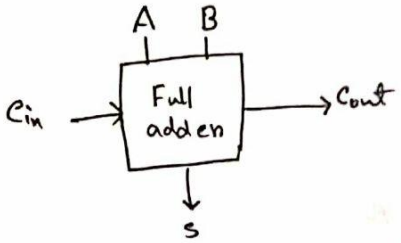


figure: 1 bit full adders circuit.

A 1 bit fulladder circuit takes 2 input each of them are 1 bit and takes another 1 bit binary number called Cin (carry in) Then the circuit add these three binary numbers each of them are 1 bit. The output of these three bit binary numers is S and another output is the carryout (Cout).

The problem requires two 8 bit binary numbers and have to use 1 bit full adder module. So, we need 8 total 8 piece of 1 bit full adders circuit.

In the time diagram we see the addition result

of $A = 10110101_{(2)} = 181_{(10)}$

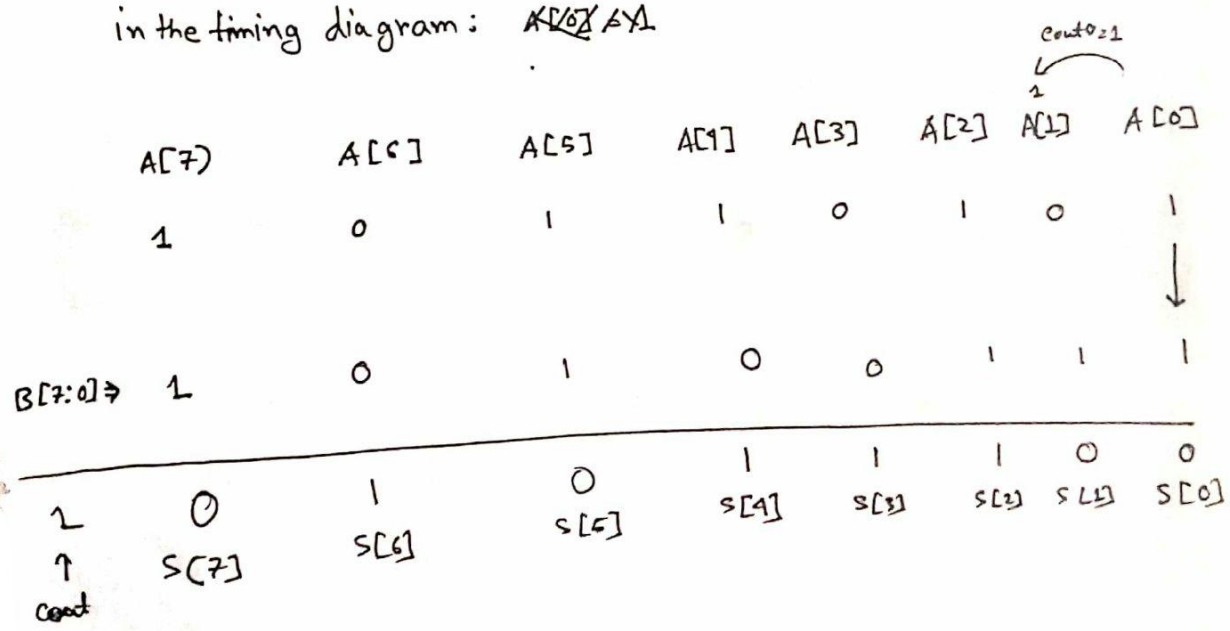
$B = 10100111_{(2)} = 167_{(10)}$

the circuit provides the output Cout = 1

S = 01 011100

So, the answer is 10101100

in the timing diagram:



As the whole circuit performs addition bit by bit that's why when A[0] = 1 and B[0] = 1 then S[0] = 0 and cout0 = 1

In time interval 0 – 80ns

A[0] = 1	B[0] = 1	cout0 = 1	S[0] = 0
A[1] = 0	B[1] = 1	Cout1 = 1	S[1] = 0
A[2] = 1	B[2] = 1	Cout2 = 0	S[2] = 1
A[3] = 0	B[3] = 0	Cout3 = 0	S[3] = 1
A[4] = 1	B[4] = 0	Cout4 = 0	S[4] = 1
A[5] = 1	B[5] = 1	Cout5 = 1	S[5] = 0
A[6] = 0	B[6] = 0	Cout6 = 0	S(6) = 1
A[7] = 1	B[7] = 1	cout = 1	S[7] = 0

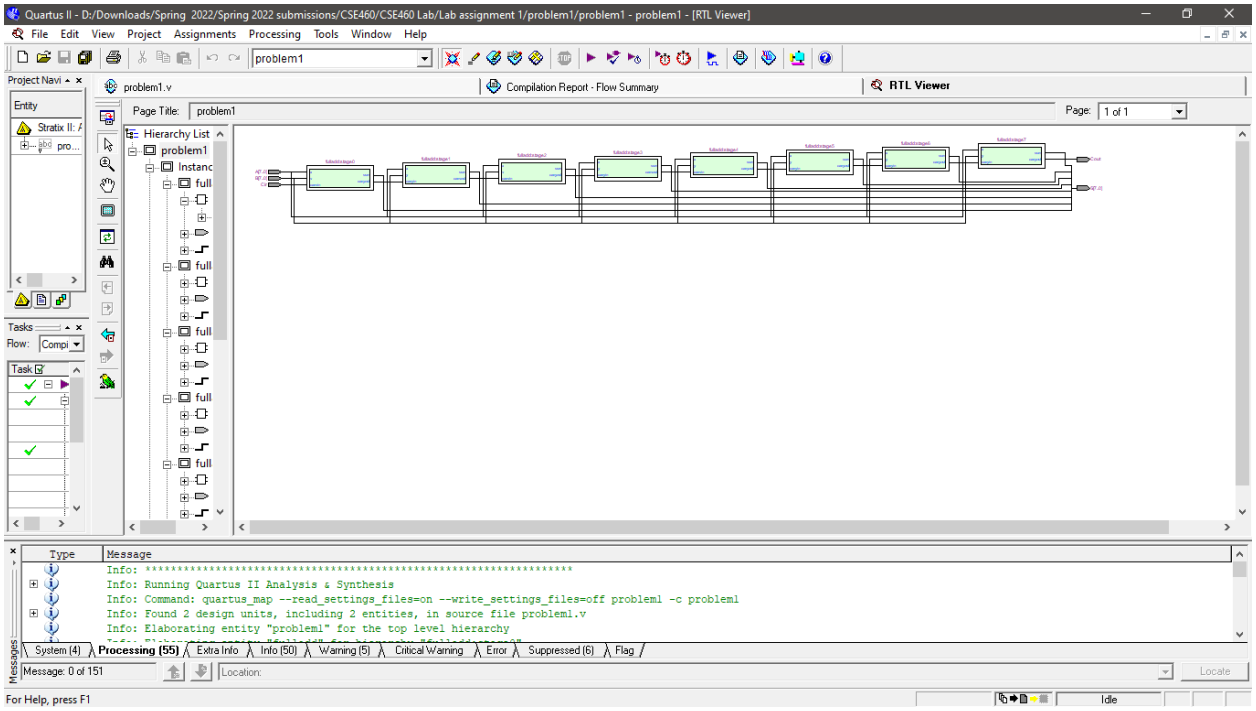
This ~~the~~ is the table representation of the time diagram.

when, A[0] = 1 and B[0] = 1 then the S[0] = 0 and the cout0 = 1, this cout0 = 1 ~~==~~ will be the Cin for A[1] and B[1].

So inthe timing diagram from 0 – 80ns the S[0] = 0 or S[0] is low. S[1] = 0, S[2] = 1, S[3] = 1, S[4] = 1, S[5] = 0 S[6] = 1, S[7] = 0 and cout = 1

This represent the binary number 10101100 which is the addition of A = 8'b10110101 and B = 8'b10100111

RTL schematic :



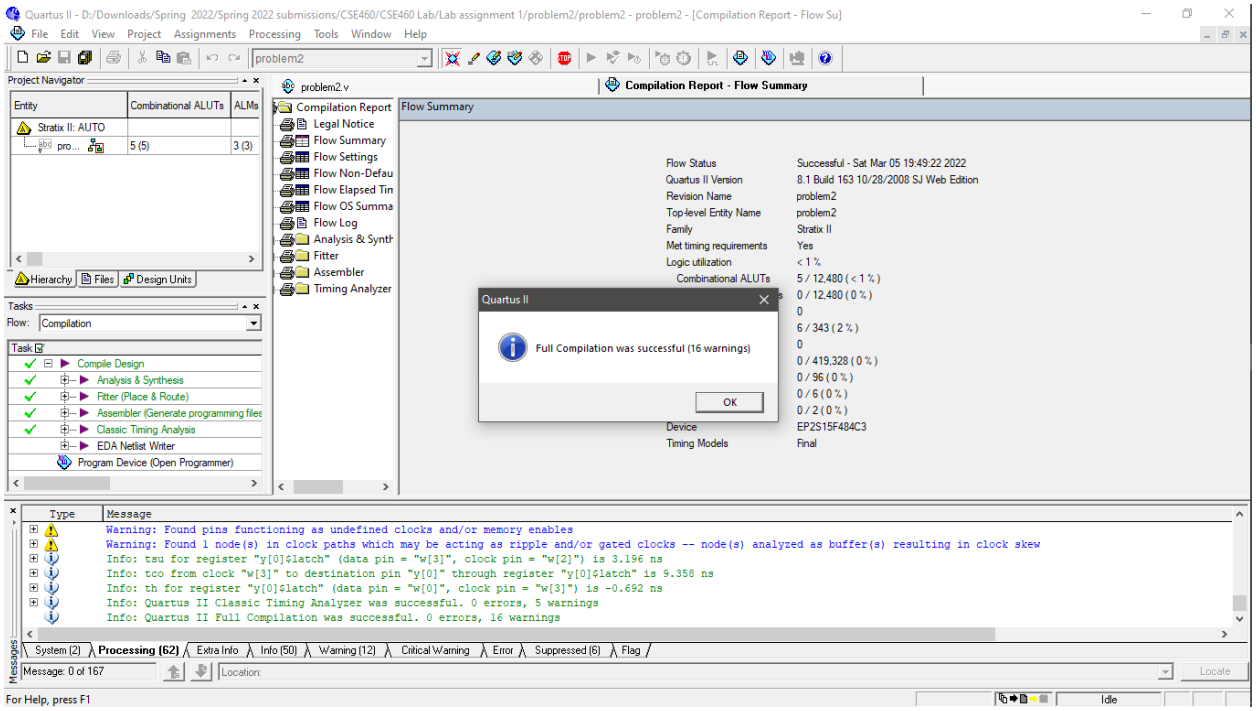
Problem 2: Design a 4 to 2 priority encoder (0>1>3>2) using Verilog HDL and verify using timing diagram.

Answer :

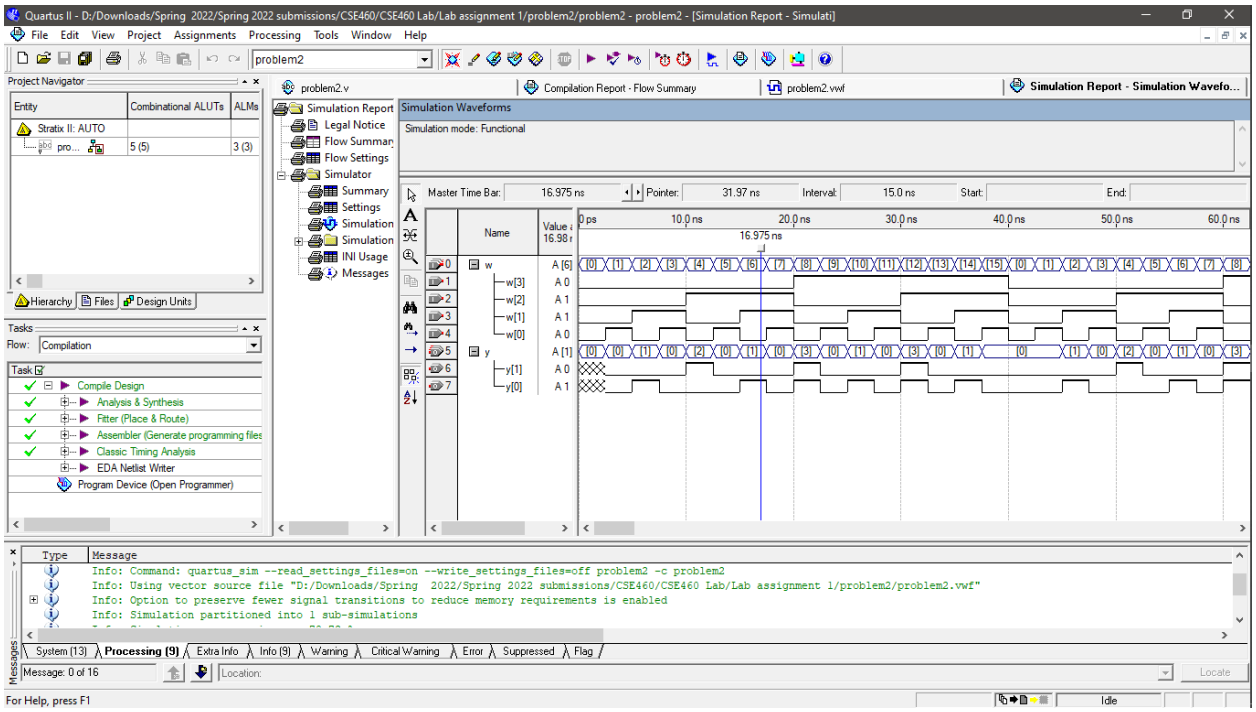
Code :

```
module problem2(w,y);

input [3:0]w;
output reg[1:0]y;

always @(w)
      casex (w)

      // priority 0>1>3>2

            4'bxxx1: y=0;
            4'bxx10: y=1;
            4'b1x00: y=3;
            4'b0100: y=2;

      endcase
endmodule
```

Compilation report :



Simulation report :



Brief description :

Problem – 2

In this problem the priority of the encoder is 0>1>3>2

which means when $W_0 = 1$ then we won't care about other W values. Same goes for when $W_1 = 1$ then W_0 have to be 0 and then we don't care about other W values and output would be 1

As it's a 4 bit priority encoder so, we will take 4 bit input and will generate 2 bit output.

The priority is 0>1>3>2 ∴ $W_0 > W_1 > W_3 > W_2$

which means,

W_3	W_2	W_1	W_0	
x	x	x	1	→ y = 00
x	x	1	0	→ y = 01
1	x	0	0	→ y = 11
0	1	0	0	→ y = 10

Let's observe a scenario when, multiple input bits are high.

time range: 30ns to 32.5ns

w[3] = 1

w[2] = 1

w[1] = 0

w[0] = 0

in this scenario w[2] = 0

So, output wouldn't be (1) cause w[0] position is occupied by 0

Now, we will look for the next priority which is w[1] and here w[1] = 0 That's why we have to look for next priority which is 3.

Now, w[3] = 1 so, we will show output = 11

Because the previous priority positions w[0] and w[1] are 0.

Most important thing is here w[2] = 1 but we didn't care about that position. Because 2 has the lower priority than 3. That's why we didn't

look at the position W[2] cause we got a 1 in W[3] position which has a higher priority than W[2]

So, the input : 1100

output : y = 11

This input shows that W[3] has higher priority than W[2] though the reason why W[2] = 1 but the output came from W[3] which is y = 3

RTL schematic :