DLD LAB

ASSIGNMENT 3

Aim: The simulate behavior of 8-bit full adder and 8-bit full Subtractor

Tools used: Vivado software

Truth table:

8-bit full adder:

input	input	input	output	output
a	b	c_{in}	s	c_{out}
00000001	00000001	0	00000010	0
00000011	00000011	0	00000110	0
10000101	10000101	0	00001010	1
01001001	01001001	0	10010010	0
11001101	11001101	0	10011010	1
01101011	01101011	0	11010110	0

Boolean expression:

Sum = (~a*~b*c) + (~a*b*~c) + (a*~b*~c) + (a*b*c)

Carry = (a*b) + (a*c) + (b*c)

8-bit full subtractor:

input	input	input	output	output
a	b	cin	s	cout
00011001	00000001	0	00011000	0
00001011	00000011	0	00001000	0
10011101	10000101	0	00011000	0
01001001	01001001	0	00000000	0
11001101	11001101	0	00000000	0
01101011	01101011	0	00000000	0

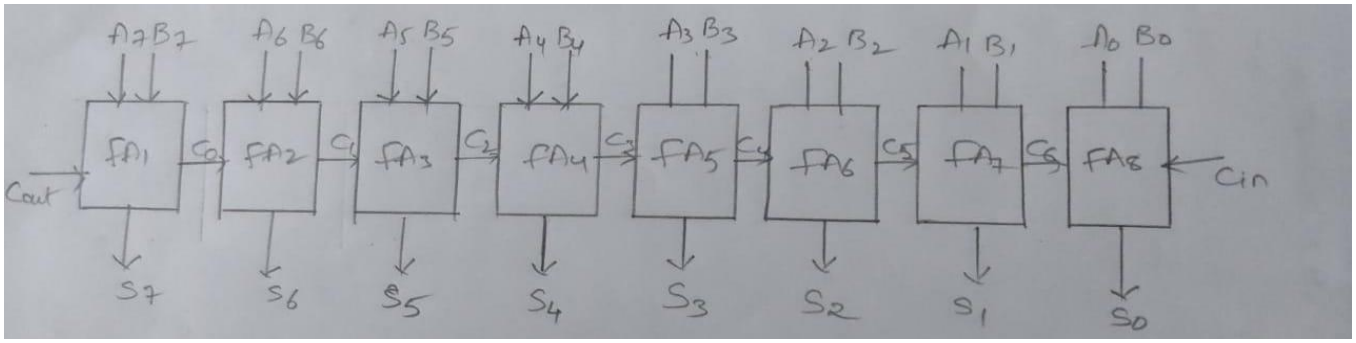
Boolean expression:

Sum = a (XOR) b c

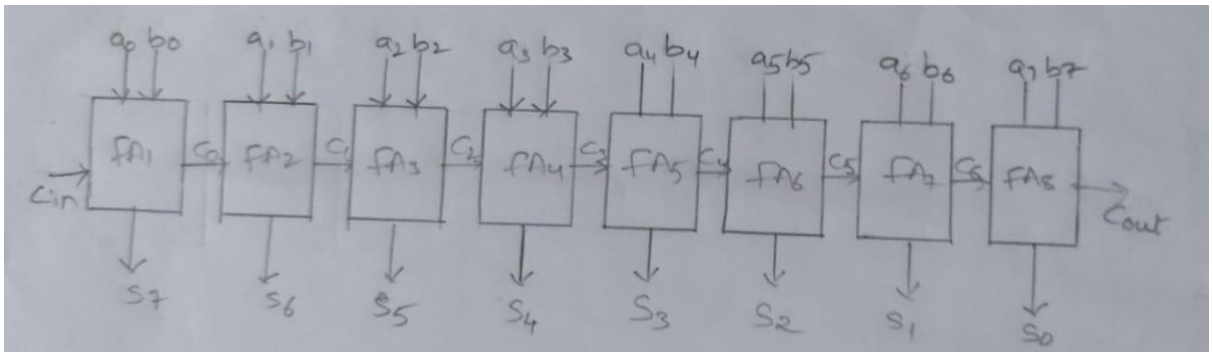
Carry = a′ b + a′c + b c

Circuit diagrams :

8-bit full adder:



8-bit full subtractor:



Codes:

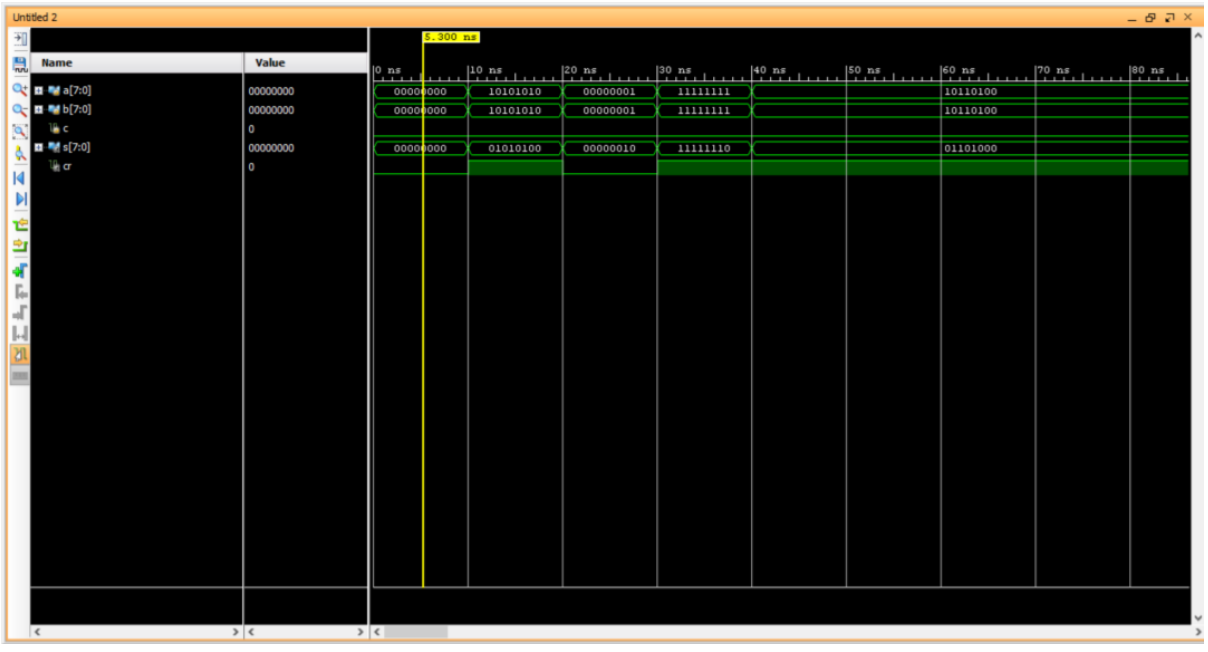
8-bit full adder:

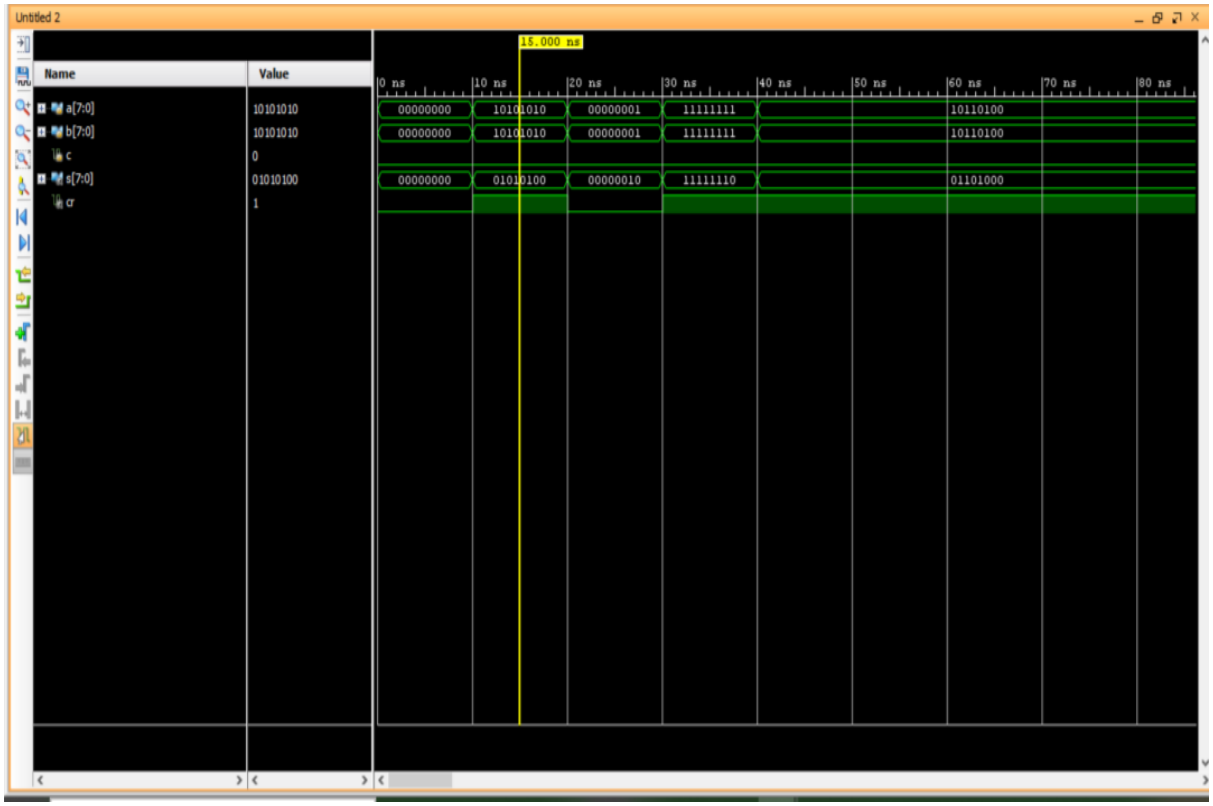
```
22 module eig_bit_adder(a,b,cin,s,count);
23 input [7:0] a;
24 input [7:0] b;
25 input cin;
26 output [7:0] s;
27 output count;
28 wire c0,c1,c2,c3,c4,c5,c6;
29 full_adder a1(a[0],b[0],0,s[0],c0);
30 full_adder a2(a[1],b[1],c0,s[1],c1);
31 full_adder a3(a[2],b[2],c1,s[2],c2);
32 full_adder a4(a[3],b[3],c2,s[3],c3);
33 full_adder a5(a[4],b[4],c3,s[4],c4);
34 full_adder a6(a[5],b[5],c4,s[5],c5);
35 full_adder a7(a[6],b[6],c5,s[6],c6);
36 full_adder a8(a[7],b[7],c6,s[7],count);
37 endmodule
```

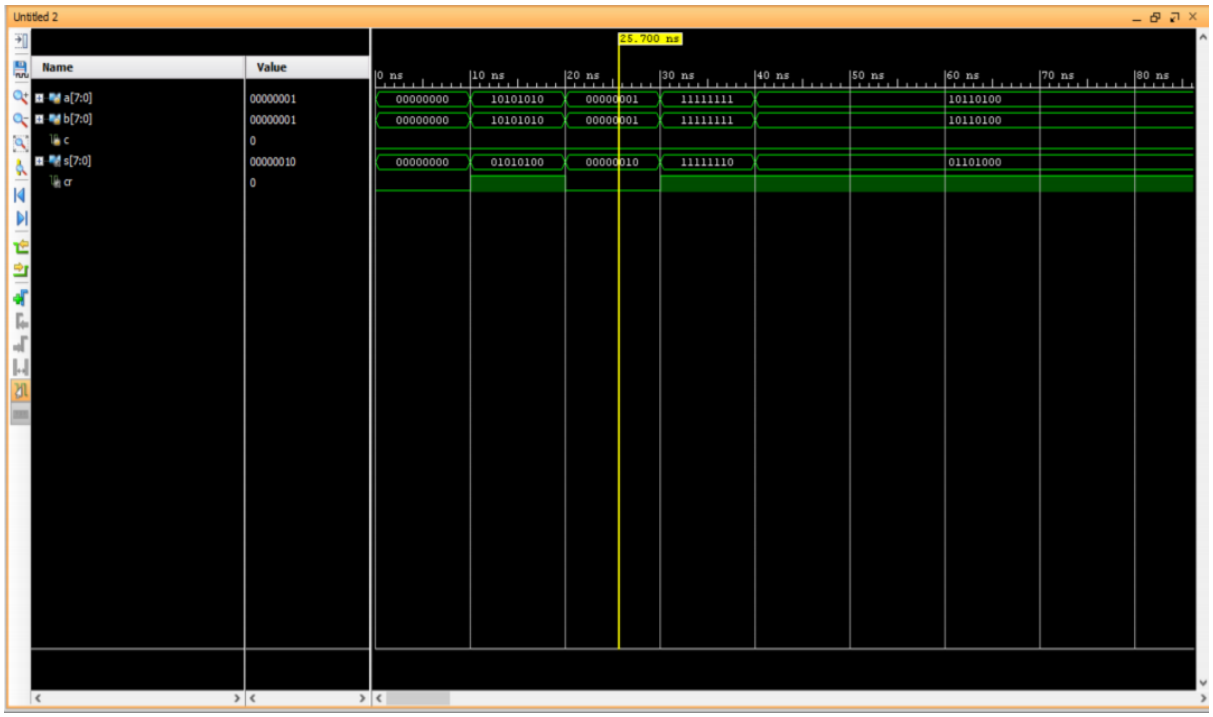
Test bench code:

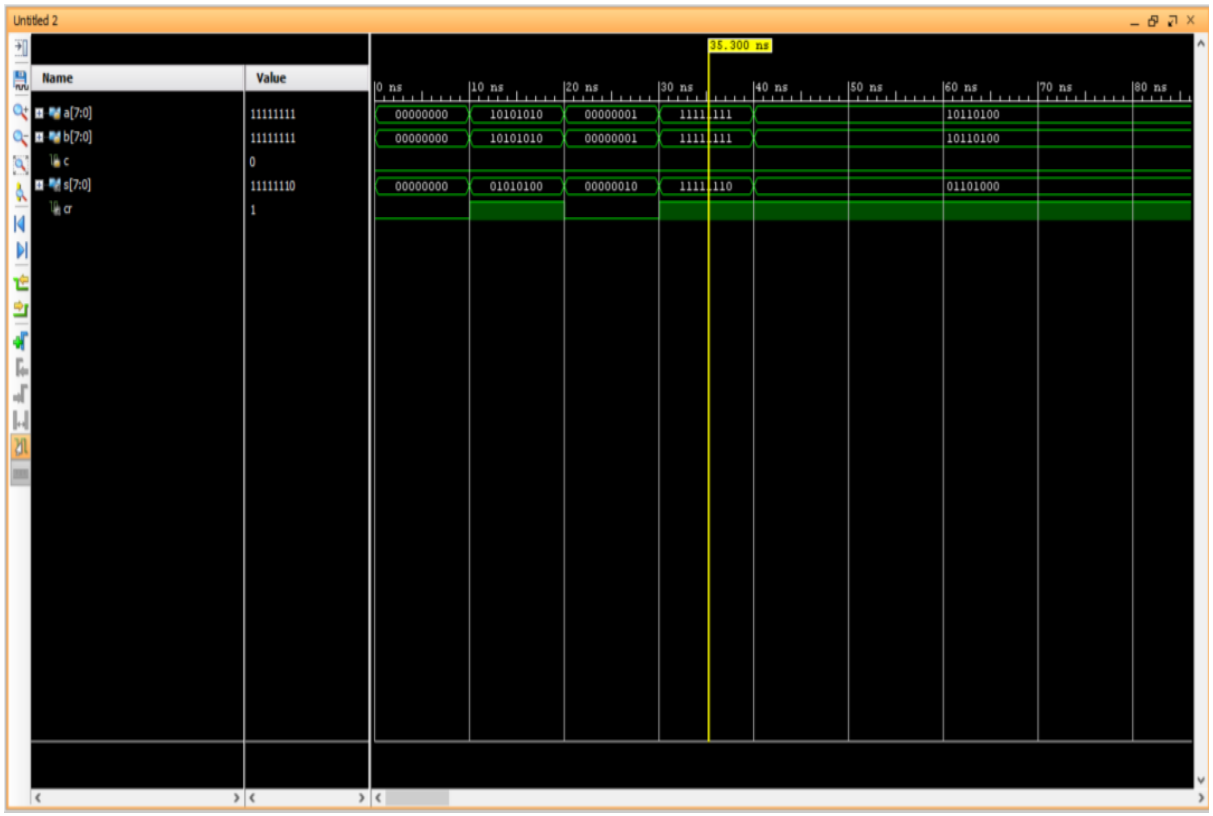
```
23 module tb_eig_bit_adder;
24 reg [7:0] a,b;
25 reg cin;
26 wire [7:0] s;
27 wire count;
28 eig_bit_adder a1(a,b,cin,s,count);
29 initial
30 begin
31  a = 8'b00000001;b = 8'b00000001;cin = 1'b0;
32 #10 a = 8'b00000011;b = 8'b00000011;cin = 1'b0;
33 #10 a = 8'b10000101;b = 8'b10000101;cin = 1'b0;
34 #10 a = 8'b01001001;b = 8'b01001001;cin = 1'b0;
35 #10 a = 8'b11001101;b = 8'b11001101;cin = 1'b0;
36 #10 a = 8'b01101011;b = 8'b01101011;cin = 1'b0;
37 end
38 endmodule
```

Results:




Untitled 2
15.000 ns
Name
Value
a[7:0]
10101010
b[7:0]
10101010
c
0
s[7:0]
01010100
cr
1
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
00000000
10101010
00000001
11111111
10110100
00000000
10101010
00000001
11111111
10110100
00000000
01010100
00000010
11111110
01101000


Untitled 2
25.700 ns
Name
Value
a[7:0]
00000001
b[7:0]
00000001
c
0
s[7:0]
00000010
cr
0
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
00000000
10101010
00000001
11111111
10110100
00000000
10101010
00000001
11111111
10110100
00000000
01010100
00000010
11111110
01101000



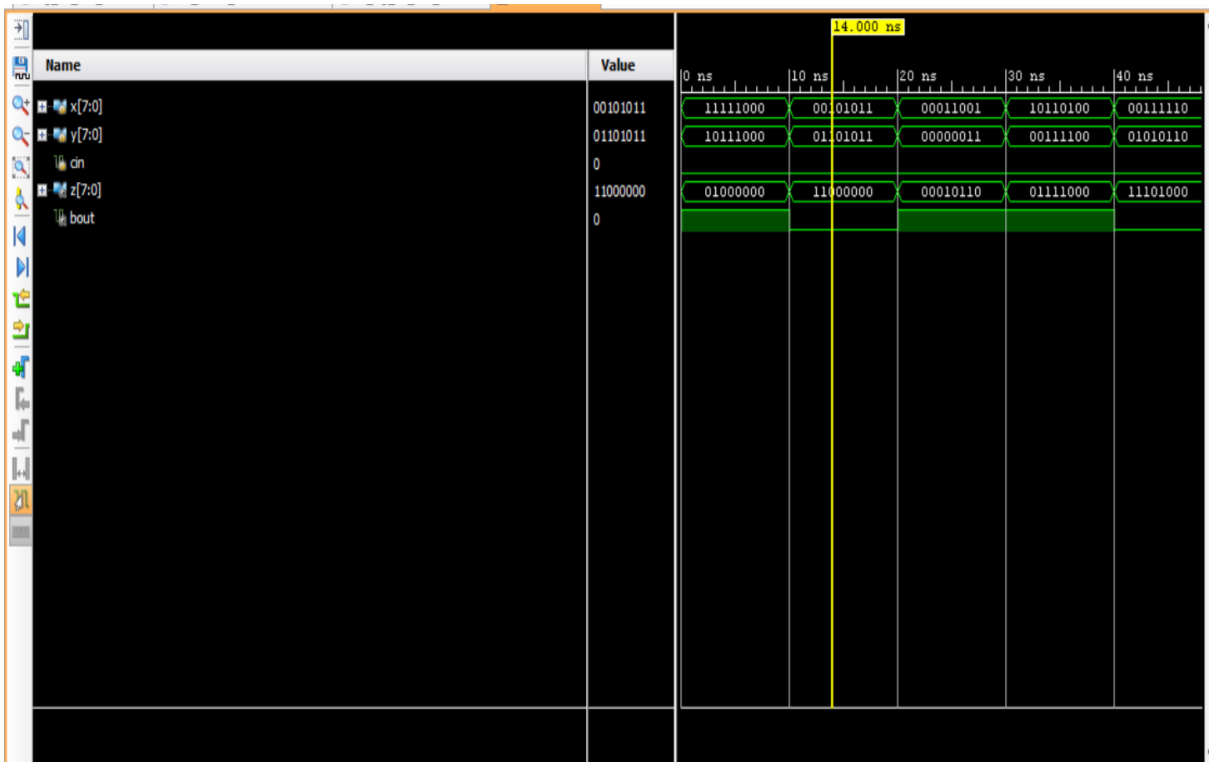
8-bit full subtractor:

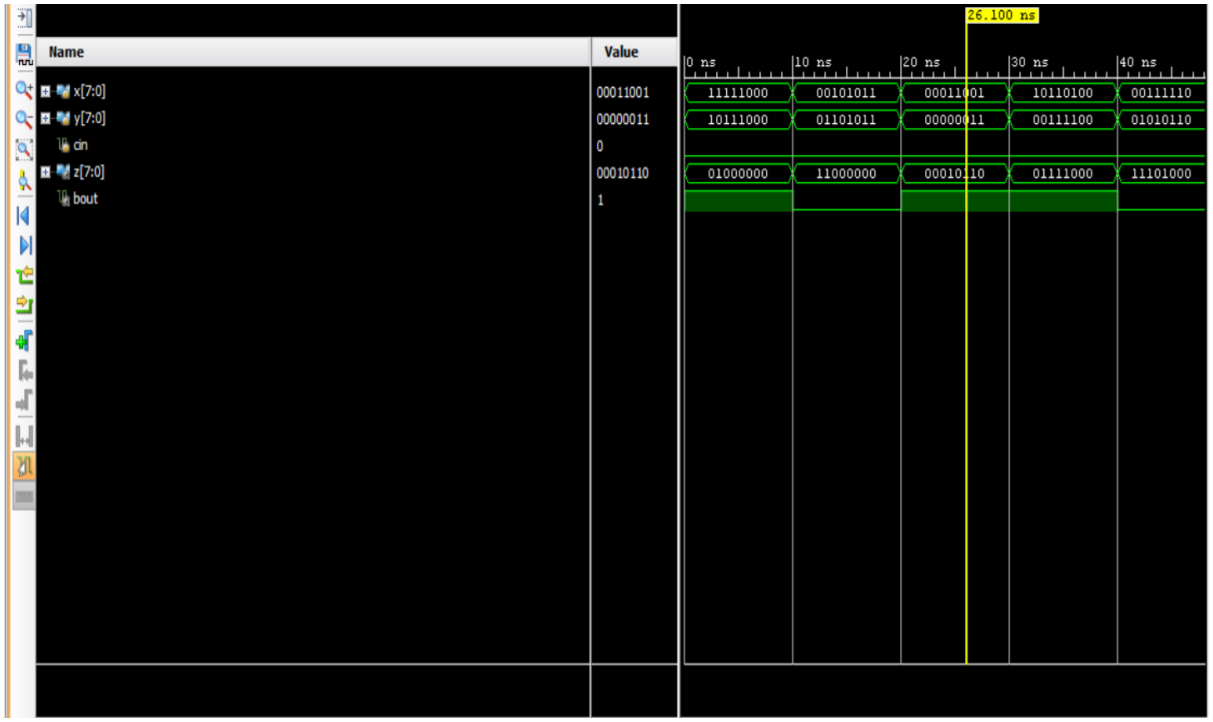
```
23 module eigbitsub(x,y,cin,z,bout);
24 input [7:0] x,y;
25 input cin;
26 output [7:0] z;
27 output bout;
28 wire c0,c1,c2,c3,c4,c5,c6;
29 full_adder fa1(x[0],~y[0],1,z[0],c0);
30 full_adder fa2(x[1],~y[1],c0,z[1],c1);
31 full_adder fa3(x[2],~y[2],c1,z[2],c2);
32 full_adder fa4(x[3],~y[3],c2,z[3],c3);
33 full_adder fa5(x[4],~y[4],c3,z[4],c4);
34 full_adder fa6(x[5],~y[5],c4,z[5],c5);
35 full_adder fa7(x[6],~y[6],c5,z[6],c6);
36 full_adder fa8(x[7],~y[7],c6,z[7],bout);
37 endmodule
```

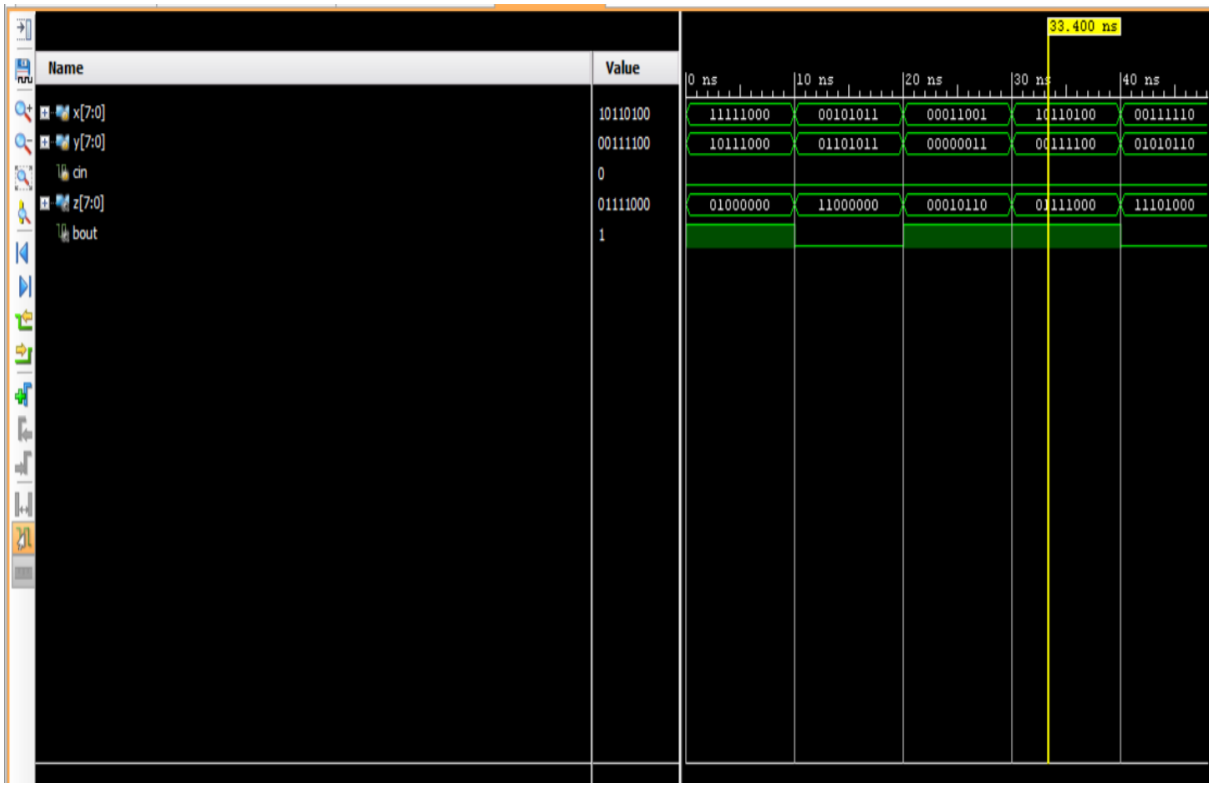
Test bench code:

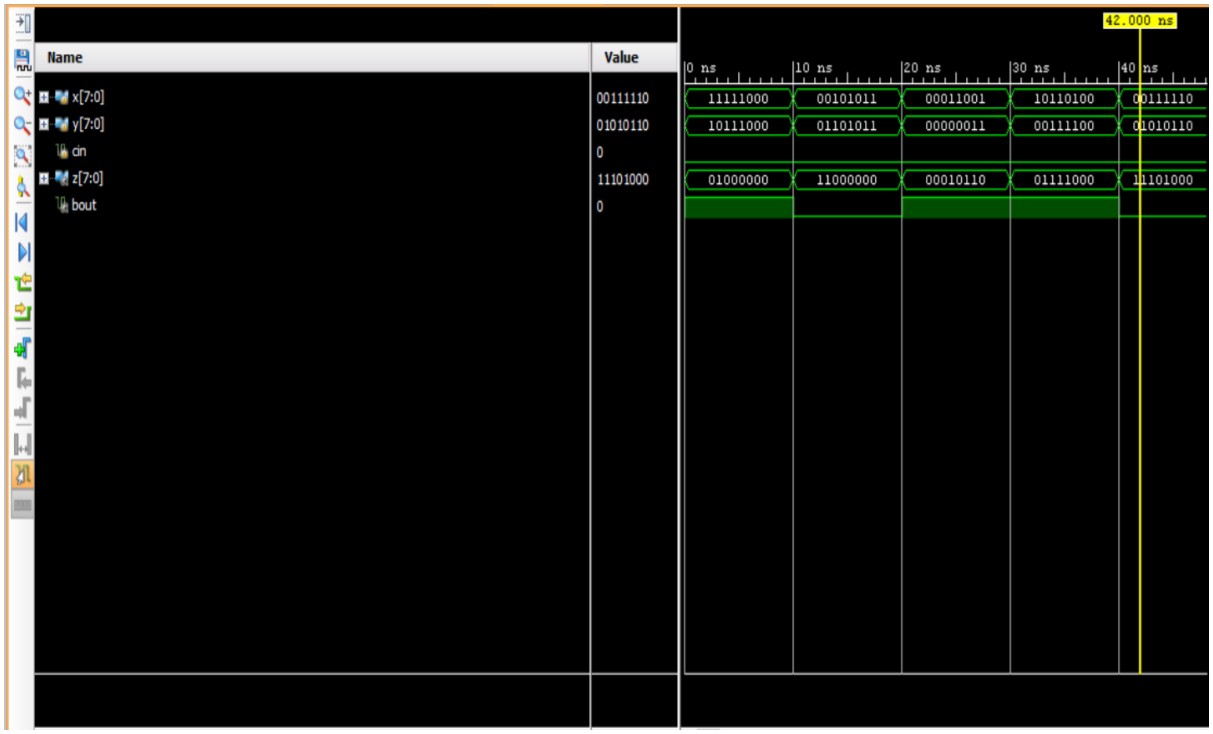
```
23 module tb_eig_bit_sub_26;
24 reg [7:0] x,y;
25 reg cin;
26 wire [7:0] z;
27 wire bout;
28 eig_bit_sub_26 f5(x,y,cin,z,bout);
29 initial
30 begin
31  x = 8'b11111000;y = 8'b10111000;cin = 1'b0;
32 #10 x = 8'b00101011;y = 8'b01101011;cin = 1'b0;
33 #10 x = 8'b00011001;y = 8'b00000011;cin = 1'b0;
34 #10 x = 8'b10110100;y = 8'b00111100;cin = 1'b0;
35 #10 x = 8'b00111110;y = 8'b01010110;cin = 1'b0;
36 #10 x = 8'b01010010;y = 8'b00010010;cin = 1'b0;
37 #10 x = 8'b00011010;y = 8'b00011000;cin = 1'b0;
38 #10 x = 8'b00111011;y = 8'b00001011;cin = 1'b0;
39 end
40 endmodule
```

Results:


Name
Value
x[7:0]
00101011
y[7:0]
01101011
cin
0
z[7:0]
11000000
bout
0
14.000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
11111000
00101011
00011001
10110100
00111110
10111000
01101011
00000011
00111100
01010110
01000000
11000000
00010110
01111000
11101000


Name
Value
x[7:0]
00011001
y[7:0]
00000011
cin
0
z[7:0]
00010110
bout
1
26.100 ns
0 ns
10 ns
20 ns
30 ns
40 ns
11111000
00101011
00011001
10110100
00111110
10111000
01101011
00000011
00111100
01010110
01000000
11000000
00010110
01111000
11101000


Name
Value
x[7:0]
10110100
y[7:0]
00111100
cin
0
z[7:0]
01111000
bout
1
33.400 ns
0 ns
10 ns
20 ns
30 ns
40 ns
11111000
00101011
00011001
10110100
00111110
10111000
01101011
00000011
00111100
01010110
01000000
11000000
00010110
01111000
11101000


Name
Value
x[7:0]
00111110
y[7:0]
01010110
cin
0
z[7:0]
11101000
bout
0
42.000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
11111000
00101011
00011001
10110100
00111110
10111000
01101011
00000011
00111100
01010110
01000000
11000000
00010110
01111000
11101000

Conclusion:

Now I'm able to design the 8-bit full adder and subtractor, its truth table, its timing diagram and figure of the logic gates