Jorge Avila (#1001543128)

Dr. Bill Carroll

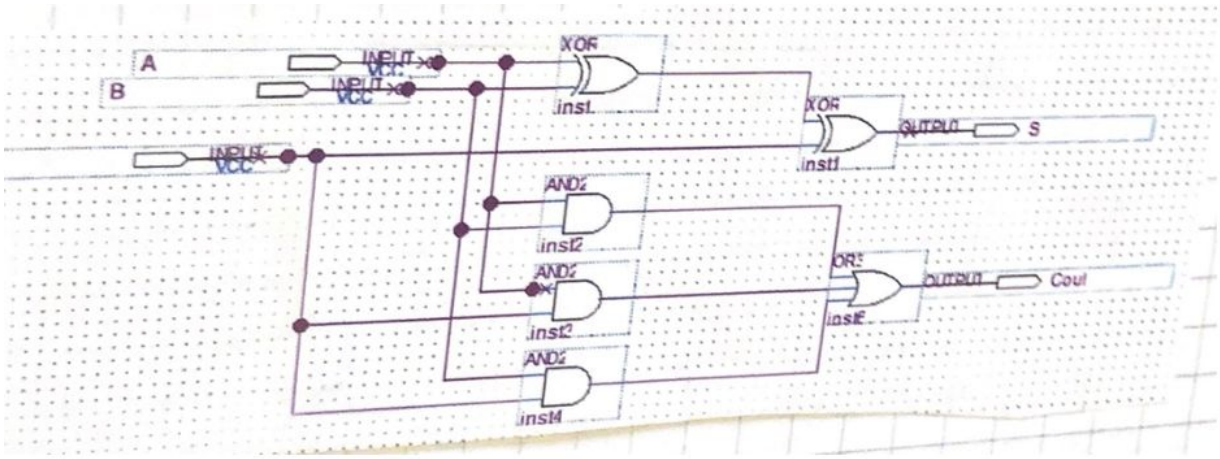
Advanced Digital Logic (CSE4357/5357)

Due Date – January 30, 2021 (11:59 PM)

Ripple-Carry Adder/Subtractor

Design Process

Designing the adder/subtractor using full-adders and XOR gates as components. Drawing the schematic diagram on quartus turned out as the diagram below:



The code is shown below:

```
module FAbehave (s,cout,a,b,cin);
      input a,b,cin;
      output reg s,cout;
      always @ (a,b,cin)
            {cout,s} = a + b + cin;
endmodule
```

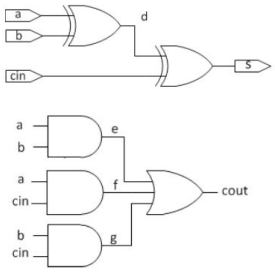
We know the full adder realization equations to the output logic as

Full Adder Realization

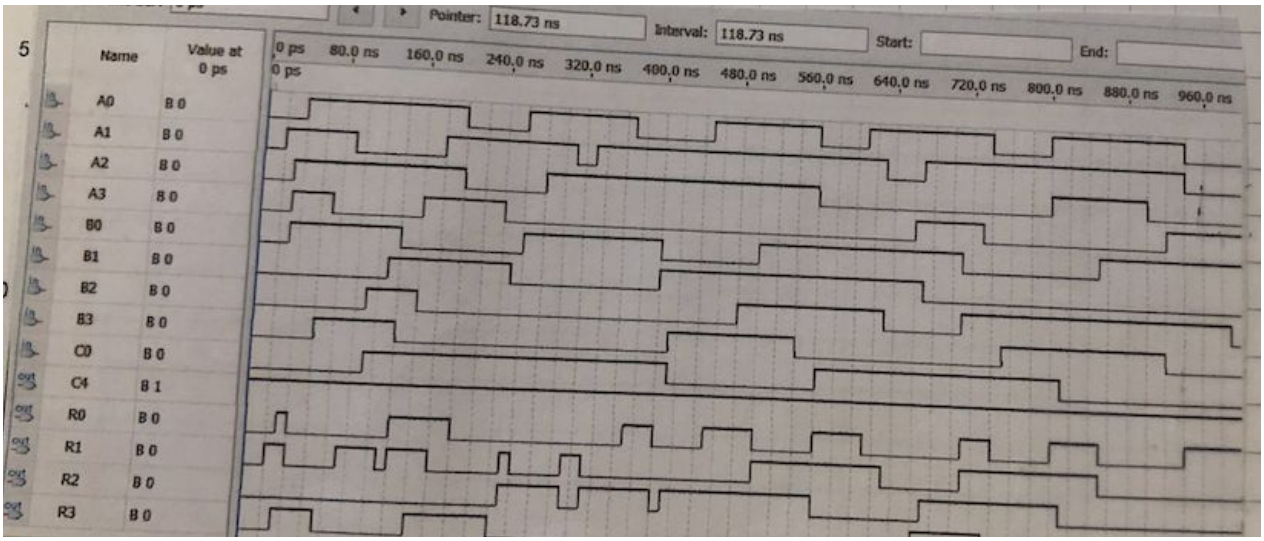
Output logic equations

$$s = a \oplus b \oplus c_{in}$$
$$= \bar{a}\bar{b}c_{in} + \bar{a}b\bar{c}_{in} + a\bar{b}\bar{c}_{in} + abc_{in}$$
$$c_{out} = \bar{a}bc_{in} + a\bar{b}c_{in} + ab\bar{c}_{in} + abc_{in}$$
$$= ab + ac_{in} + bc_{in}$$
$$= \textit{majority}\ (a,b,c_{in})$$

Logic Circuit Diagram



Once compilation success, The following waveforms were generated.



Design Verification

Simulating the design to verify its correctness. Use the following values of A and B in your simulation.

Addend	Addend	Subtrahend	Result 1	Result 2
0101	+ 0001	- 0001	**0110**	**0100**
0111	+ 0001	- 0001	**1000**	**0110**

0111	+ 1111	- 1111	0110	-1000
0110	+ 1101	- 1101	0011	-0111
1010	+ 0011	- 0011	**1101**	**0111**
1001	+ 1110	- 1110	0111	-0101
1010	+ 1110	- 1110	1000	-0100
1101	+ 1100	- 1100	**1001**	**0001**

This finalizes the table that was needed to create and test if the verilog code was working. The bolded are the * that are located in the pdf. The pictures are at the bottom of the DE1-SoC that I have taken.

Design Enhancement

The general code for the two sided seven segment decoder is composed as the sign (+/-) and magnitude. Once configured correctly and assumed low as for the activation. This will be your following code:

```
module BinToSevenSegment(
    input w,x,y,z,
    output reg a,b,c,d,e,f,g,h,i,j,k,l,m,n);
        always @ (w,x,y,z) begin
        case ({w,x,y,z})
        4'b0000: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010000001; //0
        4'b0001: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000011001111; //1
        4'b0010: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010010010; //2
        4'b0011: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010000110; //3
        4'b0100: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000011001100; //4
        4'b0101: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010100100; //5
        4'b0110: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010100000; //6
        4'b0111: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b00000010001111; //7
        4'b1000: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100000000; //-8
        4'b1001: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100001111; //-7
        4'b1010: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100100000; //-6
        4'b1011: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100100100; //-5
        4'b1100: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111101001100; //-4
        4'b1101: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100000110; //-3
        4'b1110: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111100010010; //-2
        4'b1111: {h,i,j,k,l,m,n,a,b,c,d,e,f,g} = 14'b11111101001111; //-1

        endcase
    end
endmodule
```

DE1-SoC Implementation

In this section I was able to retrieve all the necessary pins in the assignment editor and list them on the image as shown below.

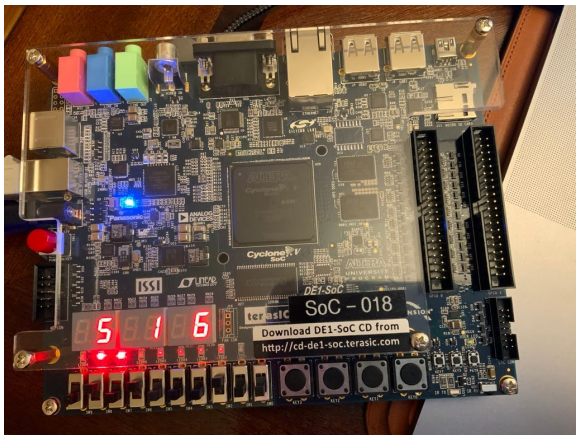
	Status	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	Ok		A[0]	Location	PIN_AE11	Yes			
2	Ok		A[1]	Location	PIN_AC9	Yes			
3	Ok		A[2]	Location	PIN_AD10	Yes			
4	Ok		A[3]	Location	PIN_AE12	Yes			
5	Ok		B[0]	Location	PIN_AF9	Yes			
6	Ok		B[1]	Location	PIN_AF10	Yes			
7	Ok		B[2]	Location	PIN_AD11	Yes			
8	Ok		B[3]	Location	PIN_AD12	Yes			
9	Ok		S[0]	Location	PIN_Y19	Yes			
10	Ok		S[1]	Location	PIN_W20	Yes			
11	Ok		S[2]	Location	PIN_W21	Yes			
12	Ok		S[3]	Location	PIN_Y21	Yes			
13	Ok		Cout	Location	PIN_W16	Yes			
14	Ok		AddSub	Location	PIN_AB12	Yes			
15	Ok		Hex0[1]	Location	PIN_AE27	Yes			
16	Ok		Hex0[2]	Location	PIN_AE28	Yes			
17	Ok		Hex0[3]	Location	PIN_AG27	Yes			
18	Ok		Hex0[4]	Location	PIN_AF28	Yes			
19	Ok		Hex0[5]	Location	PIN_AG28	Yes			
20	Ok		Hex0[6]	Location	PIN_AH28	Yes			
21	Ok		Hex1[0]	Location	PIN_AJ29	Yes			
22	Ok		Hex1[1]	Location	PIN_AH29	Yes			
23	Ok		Hex1[2]	Location	PIN_AH30	Yes			
24	Ok		Hex1[3]	Location	PIN_AG30	Yes			
25	Ok		Hex1[4]	Location	PIN_AF29	Yes			
26	Ok		Hex1[5]	Location	PIN_AF30	Yes			
27	Ok		Hex1[6]	Location	PIN_AD27	Yes			
28	Ok		Hex2[0]	Location	PIN_AB23	Yes			
29	Ok		Hex2[1]	Location	PIN_AE29	Yes			
30	Ok		Hex2[2]	Location	PIN_AD29	Yes			
31	Ok		Hex2[3]	Location	PIN_AC28	Yes			
32	Ok		Hex2[4]	Location	PIN_AD30	Yes			
33	Ok		Hex2[5]	Location	PIN_AC29	Yes			
34	Ok		Hex2[6]	Location	PIN_AC30	Yes			
35	Ok		Hex3[0]	Location	PIN_AD26	Yes			
36	Ok		Hex3[1]	Location	PIN_AC27	Yes			
37	Ok		Hex3[2]	Location	PIN_AD25	Yes			
38	Ok		Hex3[3]	Location	PIN_AC25	Yes			
39	Ok		Hex3[4]	Location	PIN_AB28	Yes			

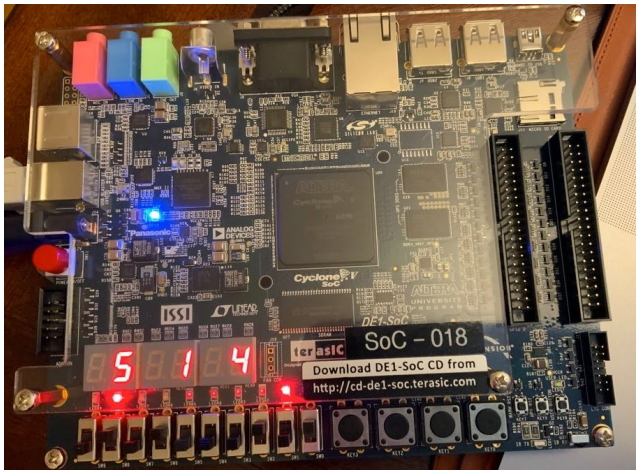
1

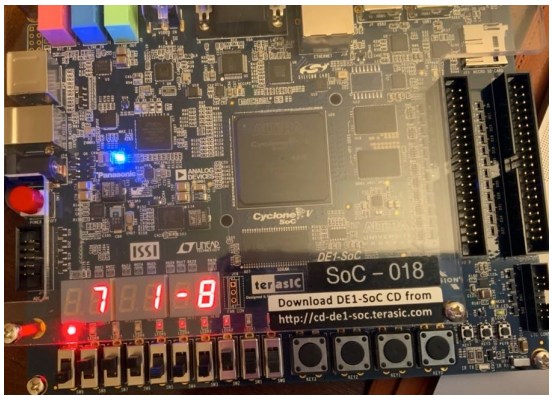
	Status	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
40	Ok		Hex3[5]	Location	PIN_AB25	Yes			
41	Ok		Hex3[6]	Location	PIN_AB22	Yes			
42	Ok		Hex4[0]	Location	PIN_AA24	Yes			
43	Ok		Hex4[1]	Location	PIN_Y23	Yes			
44	Ok		Hex4[2]	Location	PIN_Y24	Yes			
45	Ok		Hex4[3]	Location	PIN_W22	Yes			
46	Ok		Hex4[4]	Location	PIN_W24	Yes			
47	Ok		Hex4[5]	Location	PIN_V23	Yes			
48	Ok		Hex4[6]	Location	PIN_W25	Yes			
49	Ok		Hex5[0]	Location	PIN_V25	Yes			
50	Ok		Hex5[1]	Location	PIN_AA28	Yes			
51	Ok		Hex5[2]	Location	PIN_Y27	Yes			
52	Ok		Hex5[3]	Location	PIN_AB27	Yes			
53	Ok		Hex5[4]	Location	PIN_AB26	Yes			
54	Ok		Hex5[5]	Location	PIN_AA26	Yes			
55	Ok		Hex5[6]	Location	PIN_AA25	Yes			
56	Ok		Hex0[0]	Location	PIN_AE26	Yes			
57	Ok		OVR	Location	PIN_V16	Yes			
58		<<new>>	<<new>>	<<new>>					

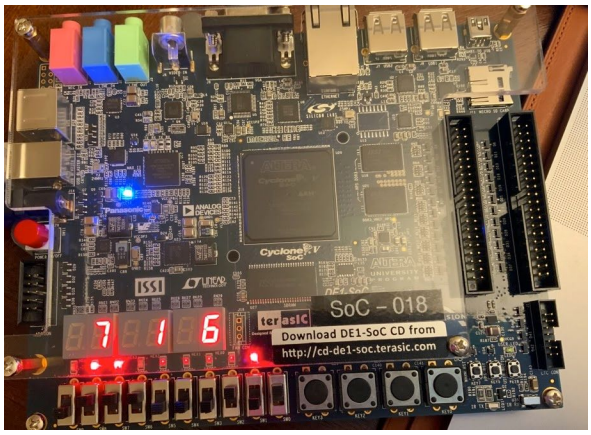
Once I was able to re-run and have a successful compilation. I programmed the DE1-SoC board and created the desired output as shown in previous images above of the DE1 board.

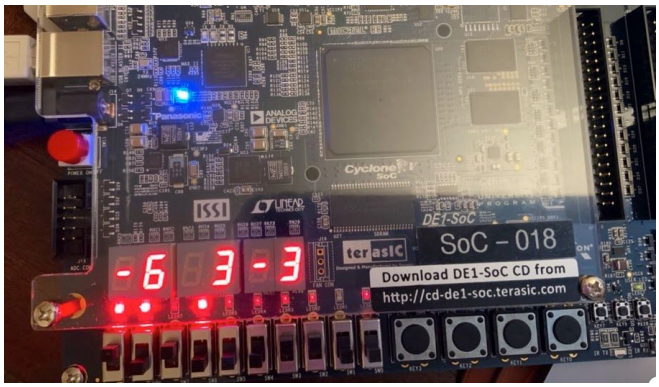
DE1-SoC Testing

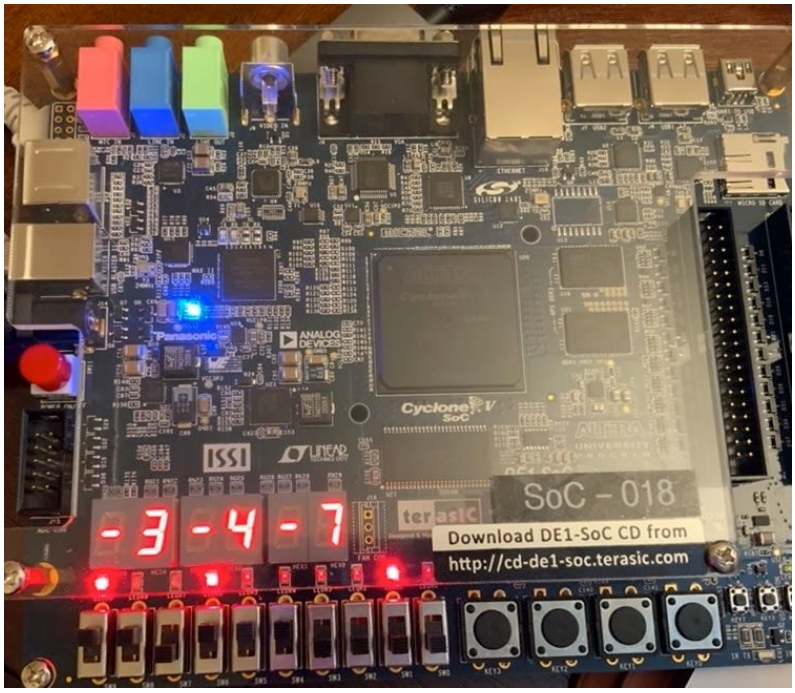


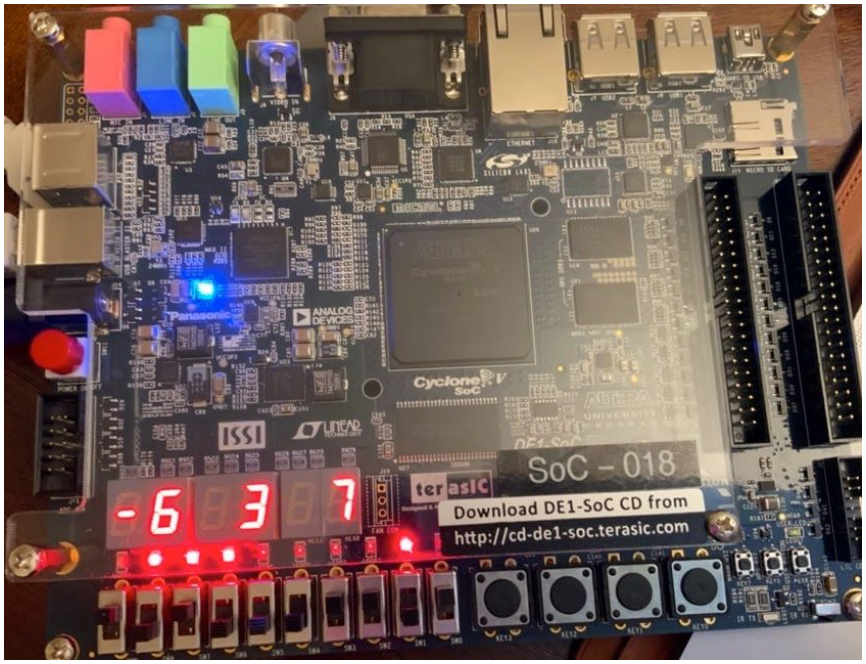


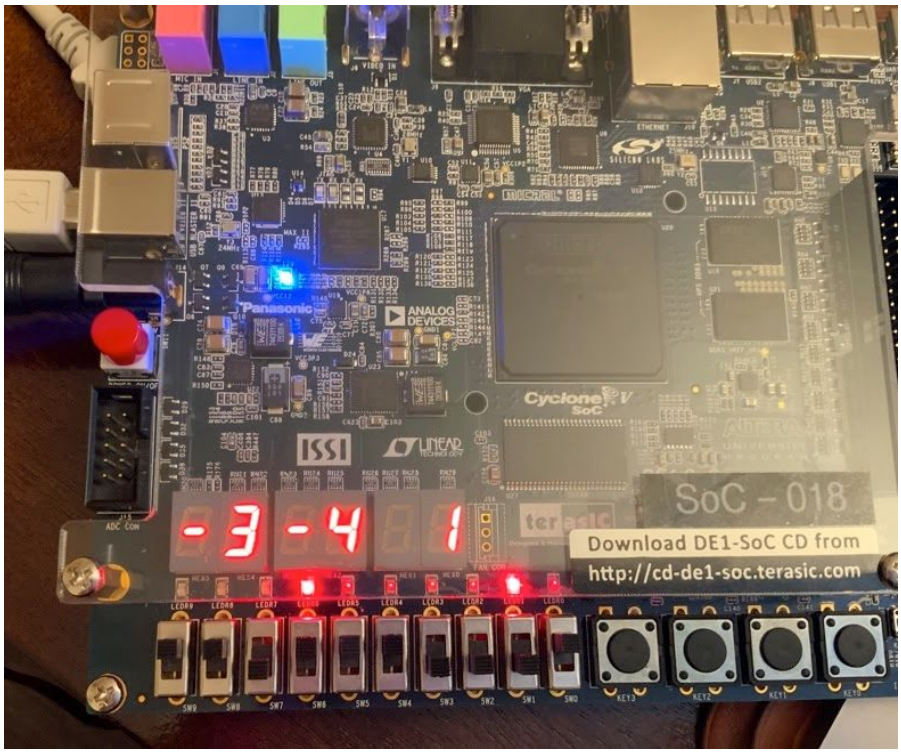

SoC – 018
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com


SoC – 018
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com


SoC – 018
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com


SoC – 018
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com





These are the pictures that are in * that have been taken. In addition, the whole code produced is shown below.

```
module RCAddSub(
```

```
        input AddSub, //this is the operation if we want to add or subtract
        input [3:0] A,B,
        output [3:0] S,
        output [6:0] Hex5, Hex4, Hex3, Hex2, Hex1, Hex0,
        output OVR,
        output Cout);
        wire [3:0] Bb;
        wire [4:0] C;


        assign Bb[3:0] = {AddSub^B[3],AddSub^B[2],AddSub^B[1],AddSub^B[0]};
        assign C[0] = AddSub; //Add 0 for addition, 1 for subtraction assign
        assign Cout = C[4]; //rename carry out port
        assign OVR = C[4] ^ C[3];
//instantiate the full adder module for each stage of the ripple carry adder
//select add (AddSub=0) or subtract (AddSub=1) operation //declare input ports



        FAbehave s0 (S[0], C[1], A[0], Bb[0], C[0]); //stage 0
        FAbehave s1 (S[1], C[2], A[1], Bb[1], C[1]); //stage 1
        FAbehave s2 (S[2], C[3], A[2], Bb[2], C[2]); //stage 2
        FAbehave s3 (S[3], C[4], A[3], Bb[3], C[3]); //stage 3

        BinToSevenSegment d0 (A,Hex5,Hex4); //print A
        BinToSevenSegment d1 (B,Hex3,Hex2); //print B
        BinToSevenSegment d2 (S,Hex1,Hex0); //print Result
endmodule


module FAbehave (
        output reg s,cout,
        input a,b,cin);
        always @ (a,b,cin)
                {cout,s} = a + b + cin;
endmodule

module BinToSevenSegment (
        input [3:0] value,
        output reg [6:0] left, right);
                always @ (value) begin
                case (value)

                                4'b0000: begin left = 7'b1111111; right = 7'b1000000; end  //0

            4'b0001: begin left = 7'b1111111; right = 7'b1111001; end  //1

            4'b0010: begin left = 7'b1111111; right = 7'b0100100; end  //2

            4'b0011: begin left = 7'b1111111; right = 7'b0110000; end  //3

            4'b0100: begin left = 7'b1111111; right = 7'b0011001; end  //4

            4'b0101: begin left = 7'b1111111; right = 7'b0010010; end  //5

            4'b0110: begin left = 7'b1111111; right = 7'b0000010; end  //6
```

```
            4'b0111: begin left = 7'b1111111; right = 7'b1111000; end  //7
            4'b1000: begin left = 7'b0111111; right = 7'b0000000; end  //-8
            4'b1001: begin left = 7'b0111111; right = 7'b1111000; end  //-7
            4'b1010: begin left = 7'b0111111; right = 7'b0000010; end  //-6
            4'b1011: begin left = 7'b0111111; right = 7'b1110010; end  //-5
            4'b1100: begin left = 7'b0111111; right = 7'b0011001; end  //-4
            4'b1101: begin left = 7'b0111111; right = 7'b0110000; end  //-3
            4'b1110: begin left = 7'b0111111; right = 7'b0100100; end  //-2
            4'b1111: begin left = 7'b0111111; right = 7'b1111001; end  //-1
                endcase
        end
endmodule
```