Frank I Gomez
4/13/21
Lab 2, Section D.
ID: 1650550
figomez
CSE 100L

Lab 2 Write Up

1. Description

The purpose of this lab was to create an 8 bit full adder, which would then be displayed on a seven-segment display as a hex value from 0 to 15/F.

2. Design

- Top Level

Diagram 1

As can be seen in Diagram 1, the 8 switches enter full adder, providing a 4 bit output, which is then taken to the seven segment display converter, which converts the 4 bits into logic functions, which then determine which segments of the display are on or off. Something to note is that the display is driven by low values, which makes the logic equations easier to compute. Finally, as only the least significant digit was powered, the display's DP, AN1, 2, and 3 values were always set to 1, to keep them turned off.

- Full Adder and Mini Adders

The adder was controlled by the 8 switches, which would act as values of 0, 1, 2, or 4. The full adder itself consisted of 3 smaller adders, which carried the output into the next respective adder, and the final output of the system.

Each mini-adder consisted of 2 functions, setting values for the carry out variable, and the sum variable.

a	b	c_in		s	c_out
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

Table 1

It can be seen by Table 1 that the sum output is actually an XOR gate, while the c_out variable is a AND b, OR'd with a XOR b AND c_in.

- Seven Segment Display

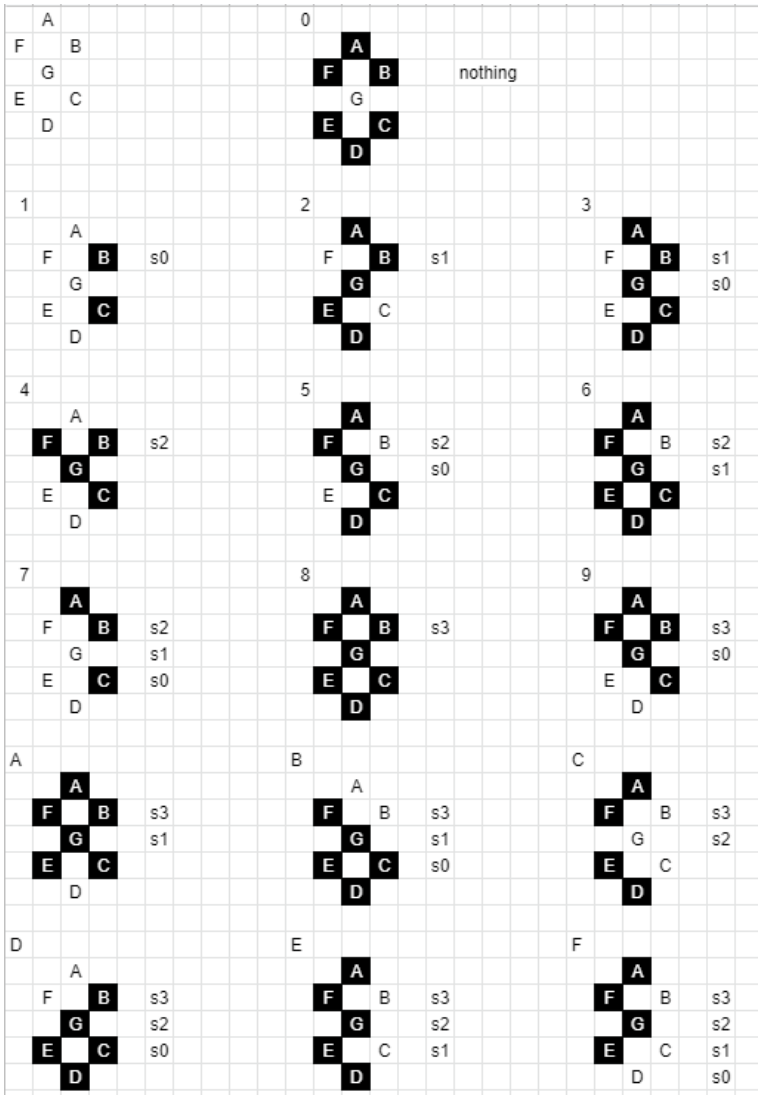


Diagram 2

The interesting segment of the display was creating logic functions for each possible value, to determine whether a light should be on or off. First, all 16 values were designed in a spreadsheet, displayed here as Diagram 2, with the black cells representing values that needed to be lit, or, in other words, set to low. As such, each of the 7 outputs consisted of a logic function OR'ing all the values for which said segment would be off, or otherwise, set to high. Take, for example, segment g in value 0. As g needed to be driven there, its logic function consisted of $s0\&s1'\&s2'\&s3'$. This would, in turn, power the g when given that input, disabling it on the display. Segment g then had this statement OR'd with the other 3 inputs for which it would need to be disabled. This method was used for all 7 outputs.

3. Testing and Simulation

Running a verilog simulation was all the testing that was required for this lab, as it provided all 16 switch combinations, and displayed their ASCII outputs, in the way that would be displayed on a physical screen. The only difficulties during testing was actually from the test harness itself, as there were a few errors (such as running for twice as long in one case, and not running the final case) that needed to be fixed prior to final testing. Initial testing revealed one of the logic statements for segment b of the seven segment display was entered incorrectly, and after fixing it, the program functioned as expected.

4. Results

- Document which pins of the FPGA you used and how they were connected to the switches and 7-segment displays.

The following FPGA pins were used: CA, CB, CC, CD, CE, CF, CG, DP, AN0, AN1, AN2, and AN3, renamed from Verilog's seg[0], seg[1], seg[2], seg[3], seg[4], seg[5], seg[6], dp, an[0], an[1], an[2],and an[3] pins. All the "C" pins were outputs from the seven-segment display module, each corresponding to a specific segment of the display. AN0 corresponded to the least significant display, and as such, was powered off. The remaining pins, DP, AN1, AN2, and AN3 were driven high, to keep those segments of the display powered off.

- Determine the longest path from any input to any output in your 3-bit adder. (i.e. the highest number of gates that any input goes through before reaching an output.)

The longest path from any input to any output in the 3 bit adder was 3, taken from either of the direct switches' inputs.

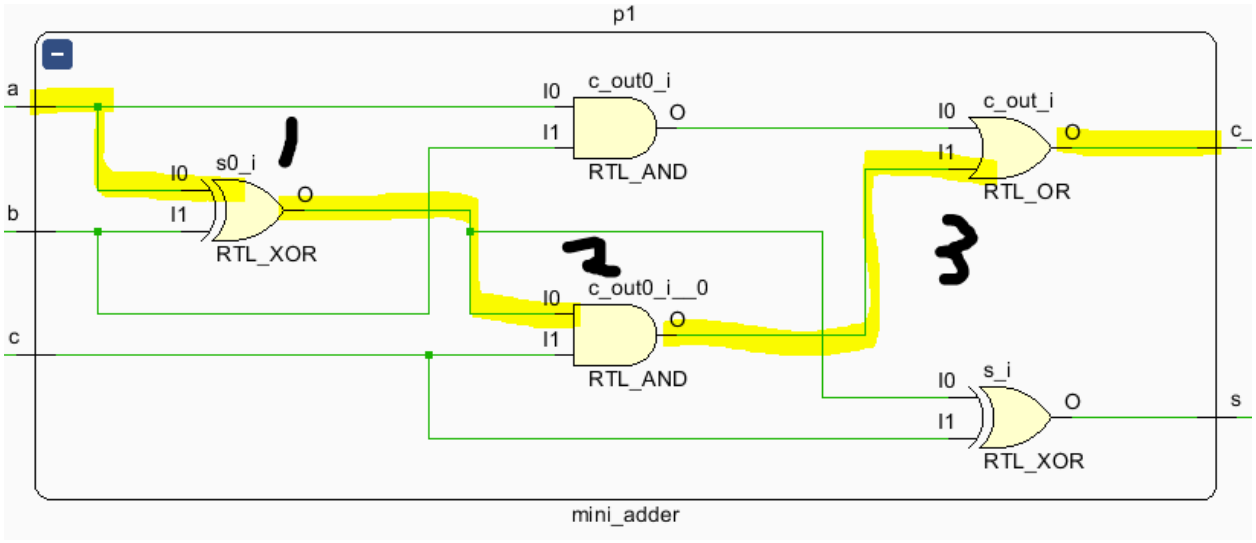


Diagram 3

Seen above in Diagram 3, input a travels through an XOR gate with input b, which is then AND'd with input c, which is then OR'd against the result of a AND b, resulting in c_out.

- For an n-bit adder determine the length of the longest path from any input to any output (in terms of n).

The longest path an input can take in an n-bit adder is n gates, traveling from the input to the xor of the carry-out.

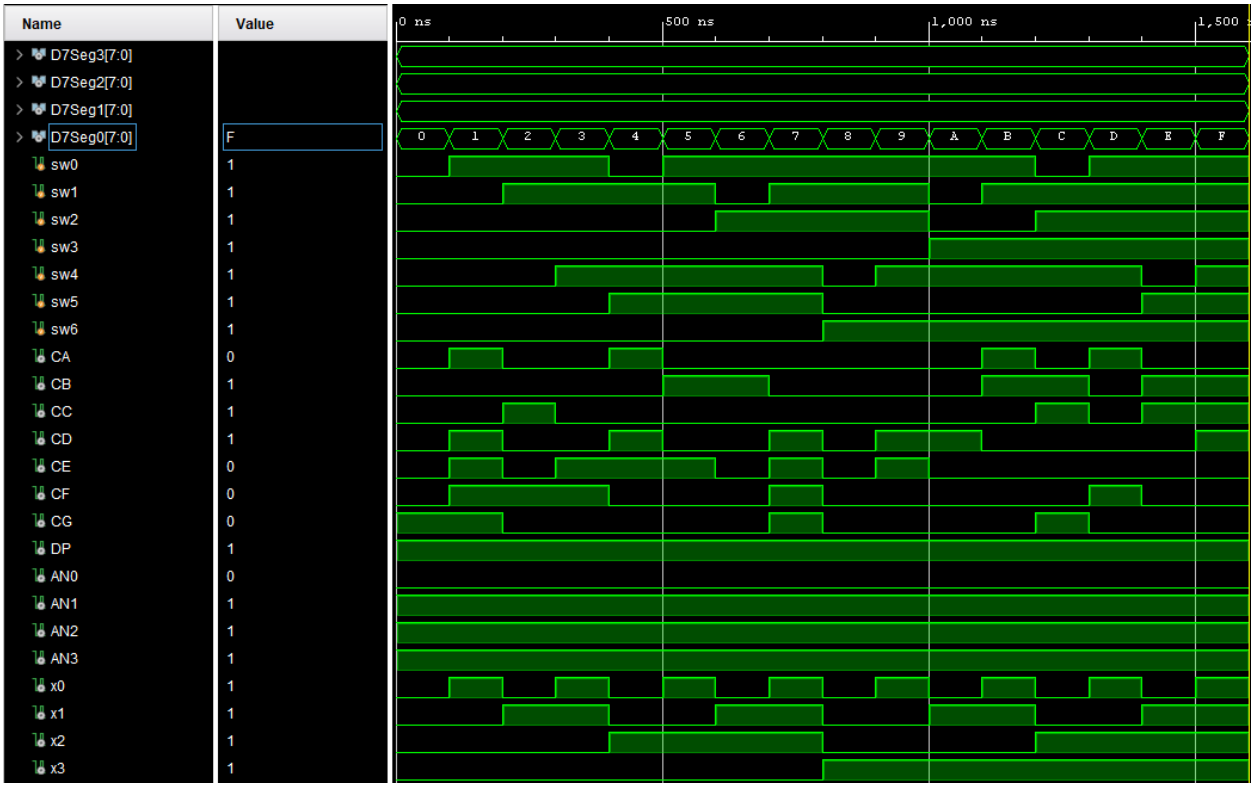
5. Conclusion

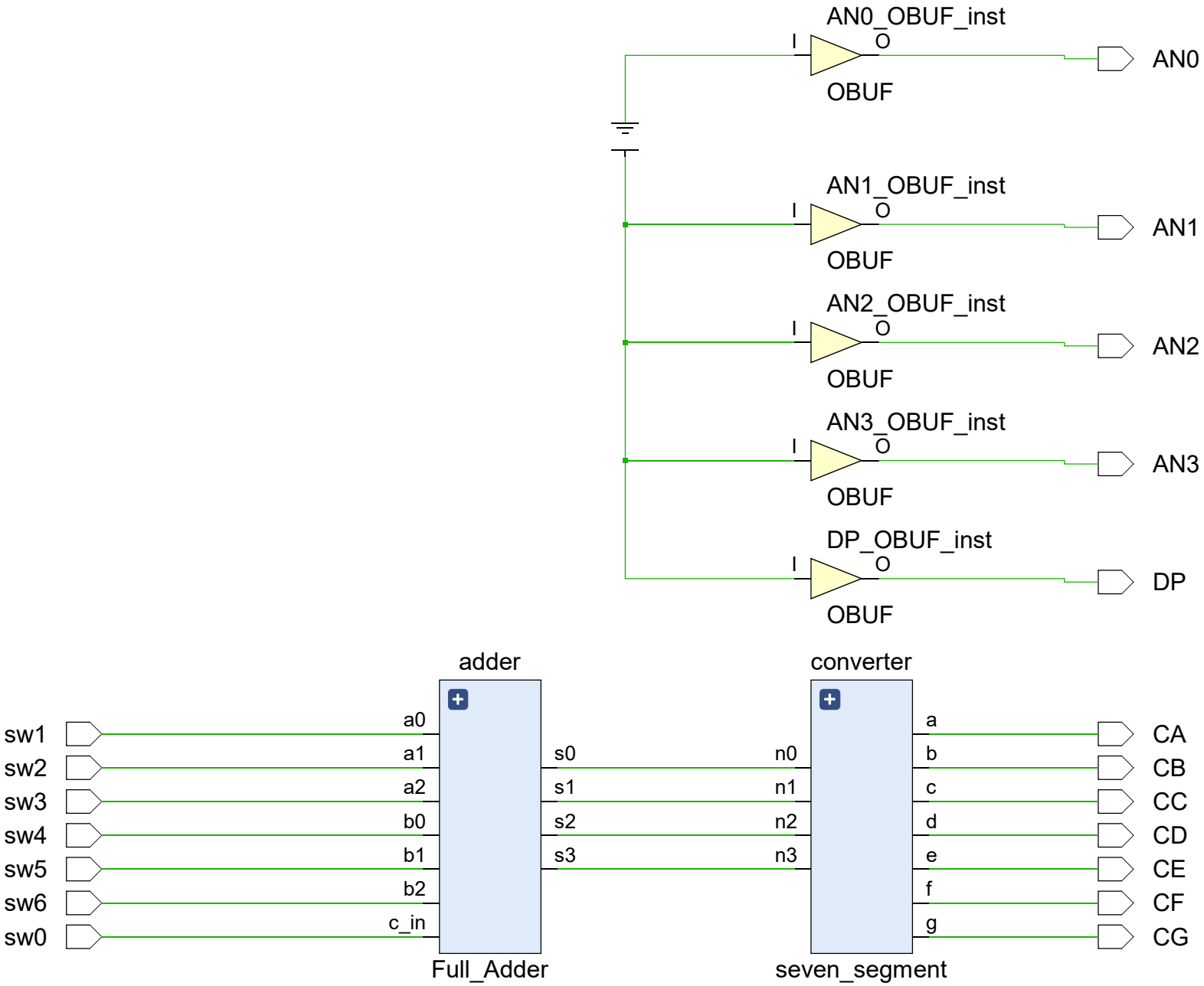
Through this lab, I learned how to transfer an input of 8 bits into a single, seven segment display, by the use of a full adder. The only real difficulties came in the form of Verilog syntax, as the logic behind the lab was relatively simple in comparison. Nothing would be changed on a subsequent attempt, as the lab provides sufficient reading to understand without needing to ask for assistance.

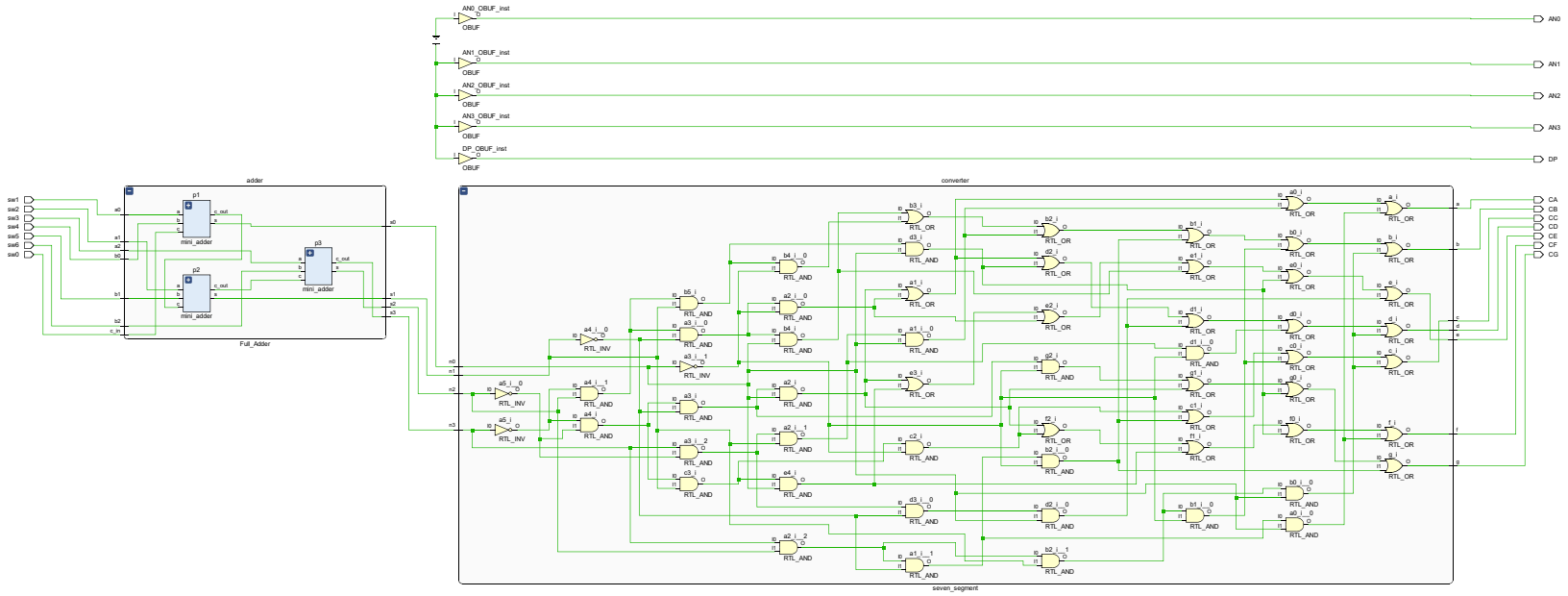
6. Appendix

(Note: Waveform Viewer cannot be printed to PDF, and as such, it is added here as a screenshot. Additionally, as the testbench was modified as stated above, it will also be attached.)

The bottom 4 values, the x variables, are the top level wires.




AN0_OBUF_inst
I
O
OBUF
AN0
AN1_OBUF_inst
I
O
OBUF
AN1
AN2_OBUF_inst
I
O
OBUF
AN2
AN3_OBUF_inst
I
O
OBUF
AN3
DP_OBUF_inst
I
O
OBUF
DP
adder
converter
sw1
sw2
sw3
sw4
sw5
sw6
sw0
a0
a1
a2
b0
b1
b2
c_in
s0
s1
s2
s3
n0
n1
n2
n3
a
b
c
d
e
f
g
CA
CB
CC
CD
CE
CF
CG
Full_Adder
seven_segment


AN0_OBUF_inst
OBUF
AN1_OBUF_inst
OBUF
AN2_OBUF_inst
OBUF
AN3_OBUF_inst
OBUF
DP_OBUF_inst
OBUF
AN0
AN1
AN2
AN3
DP
adder
p1
mini_adder
p2
mini_adder
p3
mini_adder
Full_Adder
converter
seven_segment
sw1
sw2
sw3
sw4
sw5
sw6
sw0
CA
CB
CC
CD
CE
CF
CG

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Frank Gomez
//
// Create Date: 04/11/2021 06:16:41 PM
// Design Name:
// Module Name: lab2_top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab2_top(
    input sw0,
    input sw1,
    input sw2,
    input sw3,
    input sw4,
    input sw5,
    input sw6,
    output CA,
    output CB,
    output CC,
    output CD,
    output CE,
    output CF,
    output CG,
    output AN0,
    output AN1,
    output AN2,
    output AN3,
    output DP
    );
    wire x0, x1, x2, x3;
    Full_Adder adder (.c_in(sw0), .a0(sw1), .a1(sw2), .a2(sw3), .b0(sw4), .b1(sw5),
.b2(sw6), .s0(x0), .s1(x1), .s2(x2), .s3(x3));
```

```
    seven_segment converter (.n3(x3), .n2(x2), .n1(x1), .n0(x0), .a(CA), .b(CB),
.c(CC), .d(CD), .e(CE), .f(CF), .g(CG));
    assign AN0 = 0;
    assign AN1 = 1;
    assign AN2 = 1;
    assign AN3 = 1;
    assign DP = 1;
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Frank Isidore Gomez
//
// Create Date: 04/11/2021 02:52:58 PM
// Design Name:
// Module Name: Full_Adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Full_Adder(
    input c_in,
    input a0,
    input a1,
    input a2,
    input b0,
    input b1,
    input b2,
    output s0,
    output s1,
    output s2,
    output s3
    );
    wire t0, t1;
    mini_adder p1 (.a(a0), .b(b0), .c(c_in), .s(s0), .c_out(t0)) ;
    mini_adder p2 (.a(a1), .b(b1), .c(t0), .s(s1), .c_out(t1)) ;
    mini_adder p3 (.a(a2), .b(b2), .c(t1), .s(s2), .c_out(s3)) ;
    //seven_segment sx (.n3(s3), .n2(s2), .n1(s1), .n0(s0));
endmodule
```

```
module mini_adder(
    input a,
    input b,
    input c,
    output s,
    output c_out
    );
    assign s = a ^ b ^ c;
    assign c_out = a & b | (a ^ b) & c;

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/11/2021 04:19:15 PM
// Design Name:
// Module Name: seven_segment
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module seven_segment(
    input n3,
    input n2,
    input n1,
    input n0,
    output a,
    output b,
    output c,
    output d,
    output e,
    output f,
    output g
    );
    assign a = (~n3&~n2&~n1&n0) | (~n3&n2&~n1&~n0) | (n3&~n2&n1&n0) | (n3&n2&~n1&n0);
    assign f = (~n3&~n2&~n1&n0) | (~n3&~n2&n1&~n0) | (~n3&~n2&n1&n0) |
(~n3&n2&n1&n0) | (n3&n2&~n1&n0);
    assign b = (~n3&n2&~n1&n0) | (~n3&n2&n1&~n0) | (n3&~n2&n1&n0) | (n3&n2&~n1&~n0)
| (n3&n2&n1&~n0) | (n3&n2&n1&n0);
    assign g = (~n3&~n2&~n1&~n0) | (~n3&~n2&~n1&n0) | (~n3&n2&n1&n0) | (n3&n2&~n1&~n0
    assign e = (~n3&~n2&~n1&n0) | (~n3&~n2&n1&n0) | (~n3&n2&~n1&~n0) |
(~n3&n2&~n1&n0) | (~n3&n2&n1&n0) | (n3&~n2&~n1&n0);
    assign c = (~n3&~n2&n1&~n0) | (n3&n2&~n1&~n0) | (n3&n2&n1&~n0) | (n3&n2&n1&n0);
    assign d = (~n3&~n2&~n1&n0) | (~n3&n2&~n1&~n0) | (~n3&n2&n1&n0) |
(n3&~n2&~n1&n0) |  (n3&~n2&n1&~n0) | (n3&n2&n1&n0);
```

```
endmodule
```

```
// CSE 100/L Spring 2021
// This is a testbench for the entire Lab 2 Project.
// If the top level module in your Lab 2 project is named "top_lab2"
// and you used the suggested names for its inputs/outputs then
// then it will run without modification.  Otherwise follow the instructions
// in the comments marked "TODO" to modify this testbench to conform to your project.

//Author: Martine Schlag, updated by Sagnik Nath
`timescale 1ns/1ps
// ns is the unit of measurement of delays and simulation time
//ps is the precision unit(specifies how delay values are rounded before being used
in the simulation)

module lab2_7Seg_testbench();

   wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0;
   reg sw0, sw1, sw2, sw3, sw4, sw5, sw6;
   wire CA,CB,CC,CD,CE,CF,CG,DP,AN0,AN1,AN2,AN3;



//=======================Start interface your toplevel here
=========================
// TODO: replace "top_lab2" with the name of your top level Lab 2 module.

   lab2_top UUT (
        .sw0(sw0), .sw1(sw1), .sw2(sw2),.sw3(sw3), .sw4(sw4), .sw5(sw5), .sw6(sw6),
        .CA(CA),.CB(CB),.CC(CC),.CD(CD),.CE(CE),.CF(CF),.CG(CG),.DP(DP),
        .AN0(AN0),.AN1(AN1),.AN2(AN2),.AN3(AN3)
        );
// TODO: In the three lines above, make sure the pin names match the names
// used for the inputs/outputs of your top level module.   For example, if you
// used "cin" rather than "sw0" in yout top level module, then replace ".sw0(sw0)"
with ".cin(sw0)"

//=======================Stop interface your toplevel here ========================



 show_7segDisplay  showit
(.seg({CG,CF,CE,CD,CC,CB,CA}),.DP(DP),.AN0(AN0),.AN1(AN1),.AN2(AN2),.AN3(AN3),
               .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));
    initial
    begin

        sw0=1'b0;
```

```
sw1=1'b0;
sw2=1'b0;
sw3=1'b0;
sw4=1'b0;
sw5=1'b0;
sw6=1'b0;
// sum is 0
//-------------  Current Time:  0ns
#100;  //This advances time by 100 units (ns in this case)
sw0 = 1'b1;
// sum is 1
// -------------  Current Time:  100ns
#100;
sw1 = 1'b1;
// sum is 2
// -------------  Current Time:  200ns
#100;
sw4 = 1'b1;
// sum is 3
// -------------  Current Time:  300ns
#100;
sw0 = 1'b0;
sw5 = 1'b1;
// sum is 4
// -------------  Current Time:  400ns
#100;
sw0 = 1'b1;
// sum is 5
// -------------  Current Time:  500ns
#100;
sw1 = 1'b0;
sw2 = 1'b1;
// sum is 6
// -------------  Current Time:  600ns
#100;
sw1 = 1'b1;
// sum is 7
// -------------  Current Time:  700ns
#100;
sw4 = 1'b0;
sw5 = 1'b0;
sw6 = 1'b1;
// sum is 8
// -------------  Current Time:  800ns
#100;
sw4 = 1'b1;
```

```
    // sum is 9
    // -------------  Current Time:  900ns
    #100;
    sw1 = 1'b0;
    sw2 = 1'b0;
    sw3 = 1'b1;
    // sum is 10
    // -------------  Current Time:  1000ns
    #100;
    sw0=1'b1;
    sw1=1'b1;
    sw2=1'b0;
    sw3=1'b1;
    sw4=1'b1;
    sw5=1'b0;
    sw6=1'b1;
    //sum is 11
    //---------------- Current time: 1100ns
    #100;
    sw0=1'b0;
    sw1=1'b1;
    sw2=1'b1;
    sw3=1'b1;
    sw4=1'b1;
    sw5=1'b0;
    sw6=1'b1;
//sum is 12
//------------------- Current time: 1200ns;
   #100;
   sw0=1'b1;
   sw1=1'b1;
   sw2=1'b1;
   sw3=1'b1;
   sw4=1'b1;
   sw5=1'b0;
   sw6=1'b1;
 //sum is 13
 //------------------- Current time: 1300ns
  #100;
  sw0=1'b1;
  sw1=1'b1;
  sw2=1'b1;
  sw3=1'b1;
  sw4=1'b0;
  sw5=1'b1;
  sw6=1'b1;
```

```
    //sum is 14
    //-------------------- Current time: 1400ns
      #100;
      sw0=1'b1;
      sw1=1'b1;
      sw2=1'b1;
      sw3=1'b1;
      sw4=1'b1;
      sw5=1'b1;
      sw6=1'b1;
      #100;
    //sum is 15
    //--------------------- Current time: 1500ns
    // complete this testbentch  so that all
    // 16 hex values are generated
    end
endmodule


//==============Do not edit below this line ========================================
module show_7segDisplay (
 input [6:0] seg,
 input DP,AN0,AN1,AN2,AN3,
 output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

 reg [7:0] val;

  always @(AN0 or val)
   begin
        if (AN0 == 0) D7Seg0 <= val;
        else if (AN0 == 1) D7Seg1 <= " ";
        else D7Seg0 <= 8'bX;   //  non-blocking assignment
   end

  always @(AN1 or val)
   begin
        if (AN1 == 0) D7Seg1 <= val;
        else if (AN1 == 1) D7Seg1 <= " ";
        else D7Seg1 <= 8'bX;   //  non-blocking assignment
   end

  always @(AN2 or val)
   begin
        if (AN2 == 0) D7Seg2 <= val;
        else if (AN2 == 1) D7Seg2 <= " ";
        else D7Seg2 <= 8'bX;   //  non-blocking assignment
```

```
 end

always @(AN3 or val)
 begin
      if (AN3 == 0) D7Seg3 <= val;
      else if (AN3 == 1) D7Seg3 <= " ";
      else D7Seg3 <= 8'bX;   //  non-blocking assignment
 end

  always @(seg)
  case (seg)
  7'b01111111:
       val = "-";
  7'b1111111:
       val = " ";
  7'b1000000:
       val = "0";
  7'b1111001:
       val = "1";
  7'b0100100:
       val = "2";
  7'b0110000:
       val = "3";
  7'b0011001:
       val = "4";
  7'b0010010:
       val = "5";
  7'b0000010:
       val = "6";
  7'b1111000:
       val = "7";
  7'b0000000:
       val = "8";
  7'b0011000:
       val = "9";
  7'b0001000:
       val = "A";
  7'b0000011:
       val = "B";
  7'b1000110:
       val = "C";
  7'b0100001:
       val = "D";
  7'b0000110:
       val = "E";
  7'b0001110:
```

```
        val = "F";
    default:
        val = 8'bX;
    endcase
endmodule
```