CE100 Lab Report 2
Using Adders and Displays

Student Name: Artyom Martirosyan
Lab Sec: 1B
Date: 4/17/2020

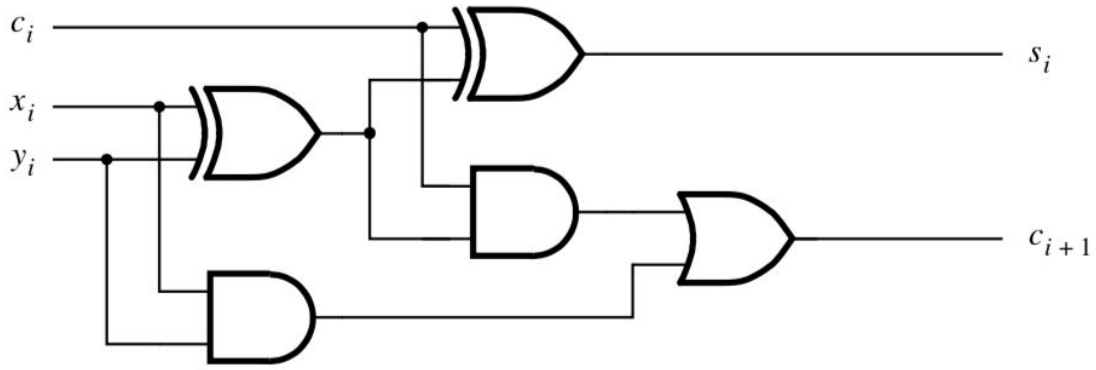
Description:

The purpose of this lab was for students to get familiar with how to implement hierarchy in out circuit designs as well as how to simulate our design by designing a 4-bit adder which would attach to a 7 segment display. The student would also need to experiment with the 7 segment display in order to understand how the different segments leading into the display worked such as the analogue and decimal point wires as well as the fact the the displays are driven high meaning that when a 1 is sent into a segment it would turn off a part of the display whereas a zero would turn it on. Students also began to experiment with more wires and how a wire can be used to represent boolean algebra rather than rewriting a bit of logic several times. For the simulation aspect of the experiment the student will begin to learn how to both use the simulation feature on vivado, as well as how to create a simulation file which would run off changing the switches at certain times.

Methods:

Full adder assembly:

A four bit adder consists of 3 3-bit adders(full adders) in order to assemble a 3 bit adder we must first draw our the schematic for the adder; fortunately the textbook we use had the circuit designed for us

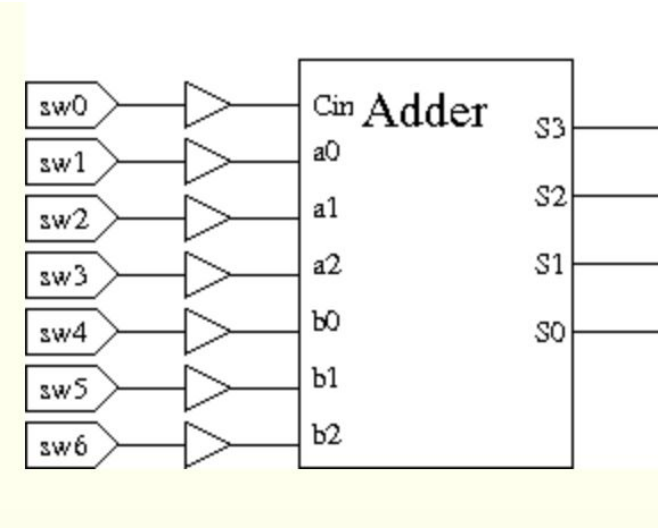


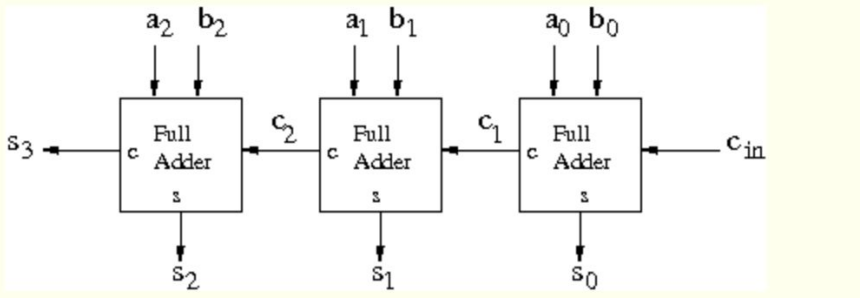
*the figure above(figure 1) is the logic design for a 3-bit adder, image taken from textbook in sources section

As seen above the four bit adder requires 3 inputs(carry, a, and b) and has 2 outputs(sum and carry) therefore we can simply create a source which takes in 3 inputs and two outputs and assign s= Cin ^ (a ^ b) as well as Cout = (a & b) | (Cin & (a ^ b))

Four-bit adder assembly:

A four bit adder will simply be calling 3 full adders:





*The figure to the left(figure 2)is the top view of the 4-bit adder where as the figure to the left(figure 3) is the internal view(both taken from CSE 100 Lab 2 manual)

As seen above the 4 bit adder will take in 7 inputs, below is a table of where each switch will be sent:

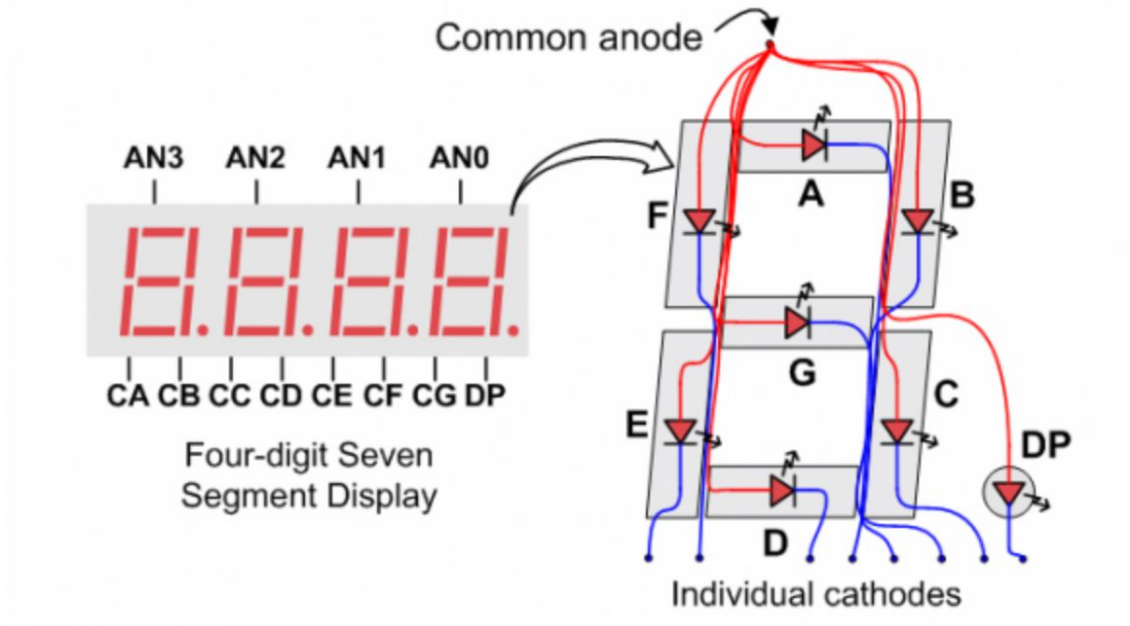
Switch	sw6	sw5	sw4	sw3	sw2	sw1	sw0
Input	b_2	b_1	b_0	a_2	a_1	a_0	c_{in}

*The figure above(figure 4) is a table showing where each input will go to(taken from lab manual)

Therefore we will need to create 3 full adders inside of the 4-bit adder module sending these switches as inputs, you will also need 2 wires to send the carry from full adder 0 to full adder 1 and from full adder 1 to full adder 2

Seven Segment display:

As mentioned above we will be using a high driven display meaning that we will need a module to convert the 4-bit number into what we want to be displayed. In order to do this we will make a new source which will have 4 inputs(given to us by the 4-bit adder) we will have 8 outputs one for each line of the figure below:



*The figure above(figure 5) is the design for the segment display which shows what each variable controls(taken from basys 3 reference manual)

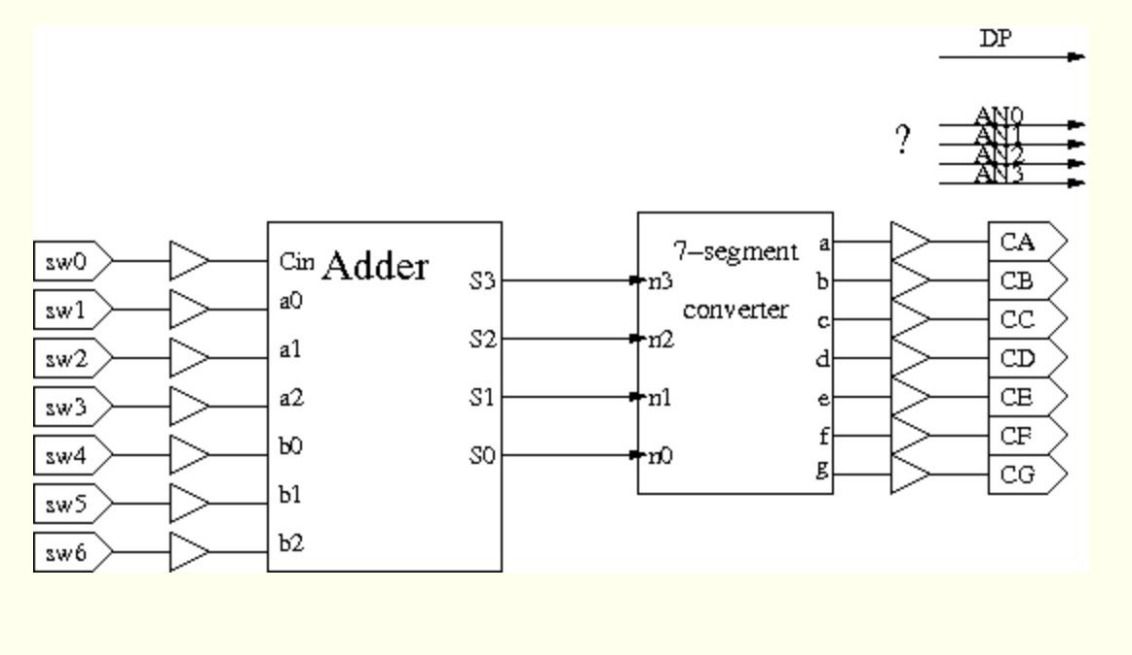
In order to do this we will first create wires which will represent the numbers coming from the adder(refer to appendices) from there we will assign boolean algebra for each variable as shown below(Note: you will need to invert the logic as this is a high driven display meaning 1 means off and 0 means off)

Assign a = ~(zero | two | three | five | six | seven | eight | nine | letter_a | letter_c | letter_e | letter_f)

You will do this assignment for all different lines in the display

Full assembly:

For the final assembly it is highly recommended to follow the schematic below(referring to wiring and naming of switches and ports):



*The figure above(figure six) is the schematic provided for students in the lab 2 manual

As seen above you will want to have wires running from the adders to the display to the led displays. In order to do this you will make a new source which will call upon the 4_bit adder and send in the switches as shown above. You will then create wires which will take the outputs from the adder and input them into the 7-segment converter(Note: follow the image above make sure sum3 goes to n3). You will then assign the converter outputs to the different leds in the segment display as seen above. You will also need to set AN1-3 to 1 and AN 0 to 0 as we are only displaying single digits.(Note: I will explain the constraints in the question section to avoid unnecessary repetition)

Simulating:

For the simulation we were given a file which we download and add into our Lab_2 vivado project, then the user will change the variables declared in the simulation file to match the variables used in their own design. Once all variables have been changed accordingly simply click run simulation. From here you will change the segment into ascii(making it depict the values we expect rather than only printing 38) and change the time span in the top left corner to about(1800ns). Then click restart and run and you should get something like the figure in the appendices section.

Results:

Personally the outputs were as expected running from 0 to f as designed, The only strange thing was when exporting the file wouldn't allow for me to print. I'm assuming that this is because the simulation was still ‘running’. After talking to the lab

assistant we decided that it would be best to simply screenshot the whole screen so that I can have some depiction of my results.

Below will be hand drawn schematics of the top level as well as the adders, I have a basic schematic of the seven segment display due to the fact that the display is very difficult to draw out due to complexity.

Design:

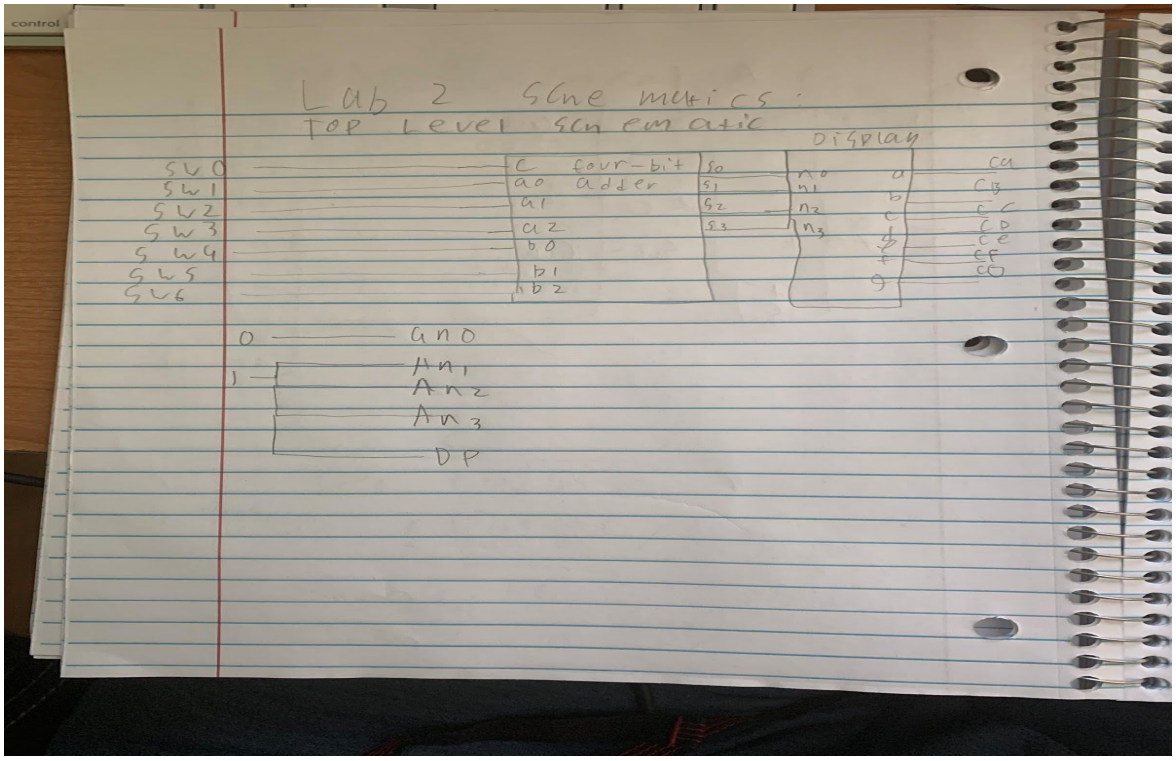
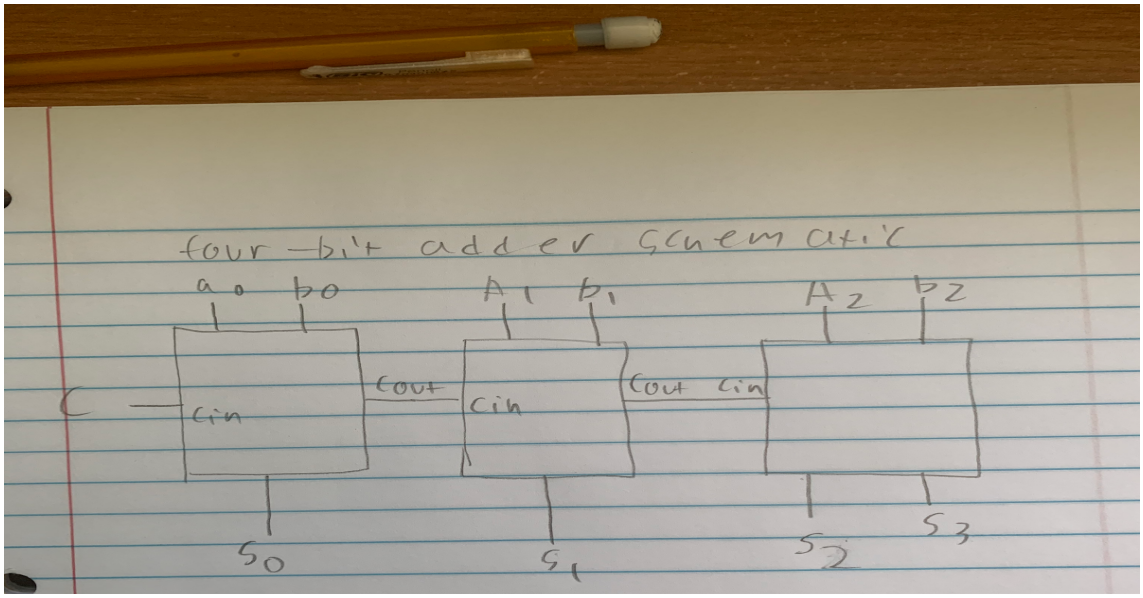


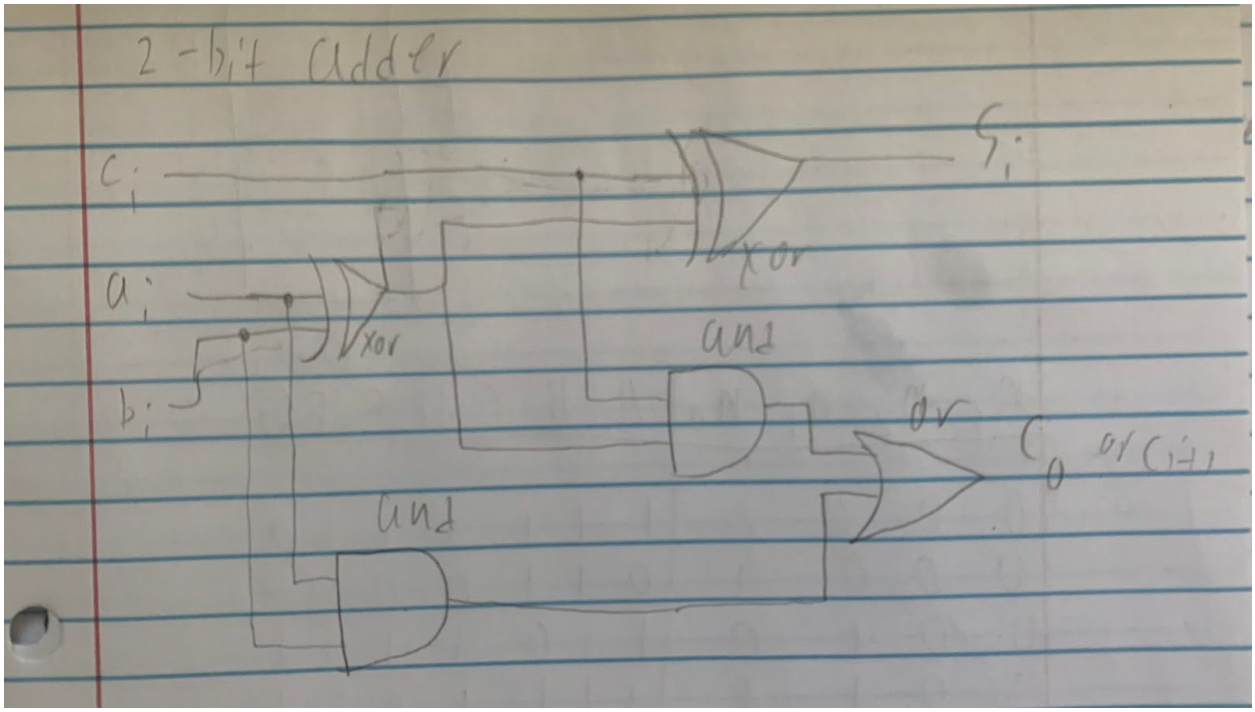
*Figure seven is a hand drawn top level schematic of the program I ended up designing

As seen above in the top level of the program I simply created a few wires called the adder function as well as the display function. Most of the extra details are mentioned in the methods section above.

*The figure below(figure eight) is my personal design for the four-bit adder

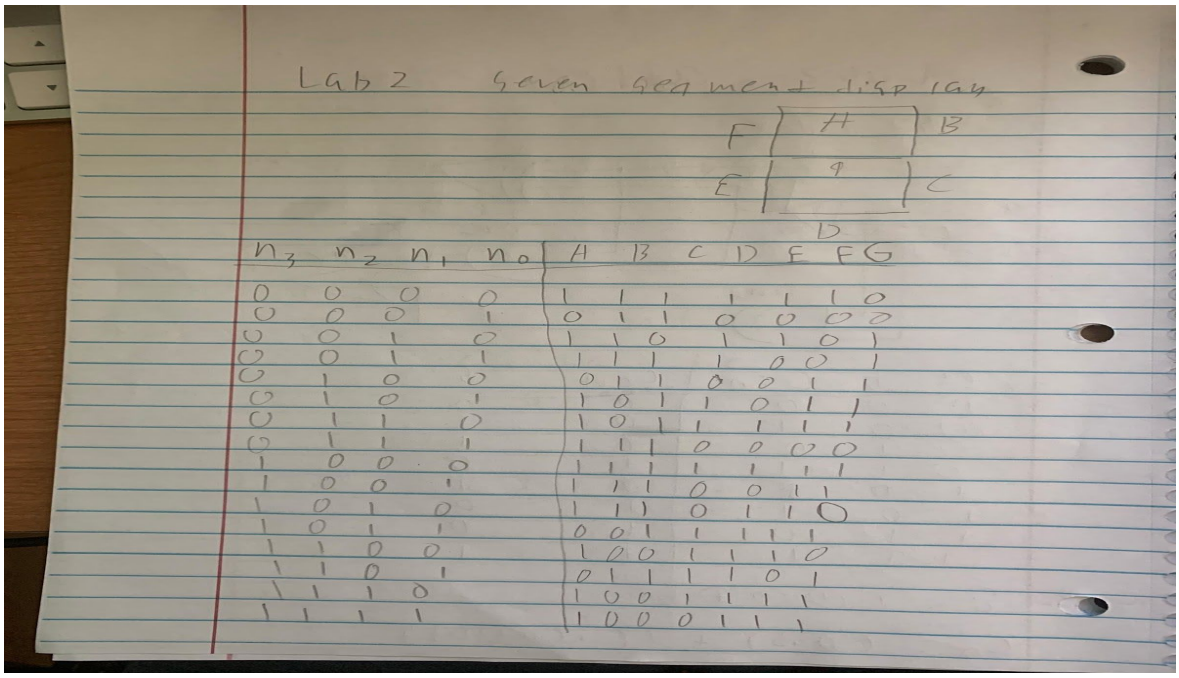


For the four-bit adder I simply created a module which would create three 2 bit adders and plug in the values as seen above.



*The image above(figure nine) is a schematic for the 2-bit adder

For this schematic I started off by creating a truth table for the adder as well as doing some research in the textbook where I stumbled upon the schematic for a 2-bit adder and used the results as mentioned above.



*The figure above(figure 10) is the truth table created for the seven segment display

Rather than drawing a schematic I decided to simply include the truth table and model of the analogue display. I ended up doing this because I realised that the logic design which was implemented is very long. This can be seen below in the computer generated schematic section in which this truth table requires roughly 60 logic gates to create. **For constraints I simply adjusted the constraint file provided.**

Testing and simulation:

For testing the students were provided with a simulation file from the professor which would test each value of the display by turning on the appropriate switches(tested from 0x1 to 0xf). I personally didnt have many issues with simulating my program as I personally had followed the manual very closely in the sense that I didn't have to alter much of the input or output gates. I ran the simulation and didn't have any issues.

Questions:

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

I personally used sw[0] - sw[6] changing the names into sw0-sw6 as depicted in the schematic provided. I also had to

use AN[0]-AN[1] again renaming into an0-an1, I also created the 7 segment displays individual leds(CA-CG). In terms of their actual connection I started off by setting AN 1-3 to 1 and set AN0 to 0 as for the switches to the four-bit adder via creating the adder and inputting them into their respected variable locations:

```
Wires w0, w1, w2, w3
Four_bit_adder adder (.C(sw0), .a0(sw1), .a1(sw2), .a2(sw3), .b0(sw4), .b1(sw5), .b2(sw6), .s0(w0), .s1(w1), .s2(w2), .s3(w3));
```

I then used the same logic below, but with the 7-segment converter sending in the inputs as wires seen above. I then had the outputs run to their expected outputs as seen below:

```
Seven_segment_display display (.n0(w0), .n1(w1), .n2(w2), .n3(w3), .a(CA), .b(CB), .c(CC), .d(CD), .e(CE), .f(CF), .g(CG));
```

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.):

If you refer to the figure off the full adder(3-bit adder) as seen there the path which requires the most logic gates it goes from either a or b into an xor gate, then into an and gate and finally into an or gate(exiting out of carry) this is the highest number of gates(3 gates).

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

For an n-bit adder the longest path is the path above(from a^b to carry) taking 3 logic gates to reach an output. This means that no matter how many bits an adder is, it will always occur therefore it's n(path from a^b to Cout) or simply 3n.

Calculate the number of possible input values (7-bit vectors) for your adder? Give the percentage of these that you tested in your simulation.

The total number of possible input values from 7 bits is 2^7 or 128, this is because a bit can only represent a 0 or 1 meaning that the total number of

combinations is 2^(number of bits). We only tested 16 numbers(0-f) meaning that the percentage is (16/128)*100 = 12.69%. **Out of the 128 possible values we only tested about 12.69% in the simulation.**

Conclusion:

In conclusion this assignment taught the student how hierarchy works in vivado as well as logic design. This assignment also introduced the usage of simulations in order to test the outcomes from our intricate designs. This also gave the students an understanding of what a high yield device looks like as well as showed the students the capabilities of wires(I personally believe this was one of the more important aspects as defining a wire makes everything look significantly cleaner). Another topic the students gained some experience with is what an adder is and how one would implement an adder into their design as well had the students develop modules which can be reused in future assignments.

Sources:

Cse 100 Lab 2 manual
https://classes.soe.ucsc.edu/cse100/Spring20/lab/lab2S20/lab2.html

'Fundamentals of Digital Logic with Verilog design', Stephen Brown and Zvonko Vranesic

Basys 3 reference manual:
https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual

Appendices:

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top
level signal names in the project

## Clock signal
#set_property PACKAGE_PIN W5 [get_ports clk]
    #set_property IOSTANDARD LVCMOS33 [get_ports clk]
    #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk

## Switches
set_property PACKAGE_PIN V17 [get_ports {sw0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw0}]
set_property PACKAGE_PIN V16 [get_ports {sw1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw1}]
set_property PACKAGE_PIN W16 [get_ports {sw2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw2}]
set_property PACKAGE_PIN W17 [get_ports {sw3}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw3}]
set_property PACKAGE_PIN W15 [get_ports {sw4}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw4}]
set_property PACKAGE_PIN V15 [get_ports {sw5}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw5}]
set_property PACKAGE_PIN W14 [get_ports {sw6}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw6}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]
```

```
## LEDs
#set_property PACKAGE_PIN U16 [get_ports {led[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {led[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {CA}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CA}]
set_property PACKAGE_PIN W6 [get_ports {CB}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CB}]
set_property PACKAGE_PIN U8 [get_ports {CC}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CC}]
set_property PACKAGE_PIN V8 [get_ports {CD}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CD}]
set_property PACKAGE_PIN U5 [get_ports {CE}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CE}]
```

```
set_property PACKAGE_PIN V5 [get_ports {CF}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CF}]
set_property PACKAGE_PIN U7 [get_ports {CG}]
    set_property IOSTANDARD LVCMOS33 [get_ports {CG}]

set_property PACKAGE_PIN V7 [get_ports {DP}]
    set_property IOSTANDARD LVCMOS33 [get_ports {DP}]

set_property PACKAGE_PIN U2 [get_ports {AN0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {AN0}]
set_property PACKAGE_PIN U4 [get_ports {AN1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {AN1}]
set_property PACKAGE_PIN V4 [get_ports {AN2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {AN2}]
set_property PACKAGE_PIN W4 [get_ports {AN3}]
    set_property IOSTANDARD LVCMOS33 [get_ports {AN3}]


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btnC]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnC]
#set_property PACKAGE_PIN T18 [get_ports btnU]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnU]
#set_property PACKAGE_PIN W19 [get_ports btnL]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnL]
#set_property PACKAGE_PIN T17 [get_ports btnR]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnR]
#set_property PACKAGE_PIN U17 [get_ports btnD]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnD]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
```

```
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]



##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
```

```
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]


##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
```

```
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]



##VGA Connector
#set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
#set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
#set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
#set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
#set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
#set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
#set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
#set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
#set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
#set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]
#set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]
#set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]
#set_property PACKAGE_PIN P19 [get_ports Hsync]
    #set_property IOSTANDARD LVCMOS33 [get_ports Hsync]
#set_property PACKAGE_PIN R19 [get_ports Vsync]
    #set_property IOSTANDARD LVCMOS33 [get_ports Vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]
    #set_property IOSTANDARD LVCMOS33 [get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]
    #set_property IOSTANDARD LVCMOS33 [get_ports RsTx]


##USB HID (PS/2)
#set_property PACKAGE_PIN C17 [get_ports PS2Clk]
    #set_property IOSTANDARD LVCMOS33 [get_ports PS2Clk]
```

```
    #set_property PULLUP true [get_ports PS2Clk]
#set_property PACKAGE_PIN B17 [get_ports PS2Data]
    #set_property IOSTANDARD LVCMOS33 [get_ports PS2Data]
    #set_property PULLUP true [get_ports PS2Data]


##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
    #set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/07/2020 10:06:18 AM
// Design Name:
// Module Name: Lab_2_Assembly
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Lab_2_Assembly(
    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 DP,
    output AN0,
    output AN1,
    output AN2,
    output AN3
    );
    wire w0, w1, w2, w3;
    assign DP = 1;
    assign AN0 = 0;
```

```
    assign AN1 = 1;
    assign AN2 = 1;
    assign AN3 = 1;
    Four_bit_adder adder (.C(sw0), .a0(sw1), .a1(sw2), .a2(sw3), .b0(sw4), .b1(sw5),
.b2(sw6),.S0(w0), .S1(w1), .S2(w2), .S3(w3));
    seven_segment_display display (.n0(w0), .n1(w1), .n2(w2), .n3(w3), .a(CA),
.b(CB), .c(CC), .d(CD), .e(CE), .f(CF), .g(CG));

endmodule
```

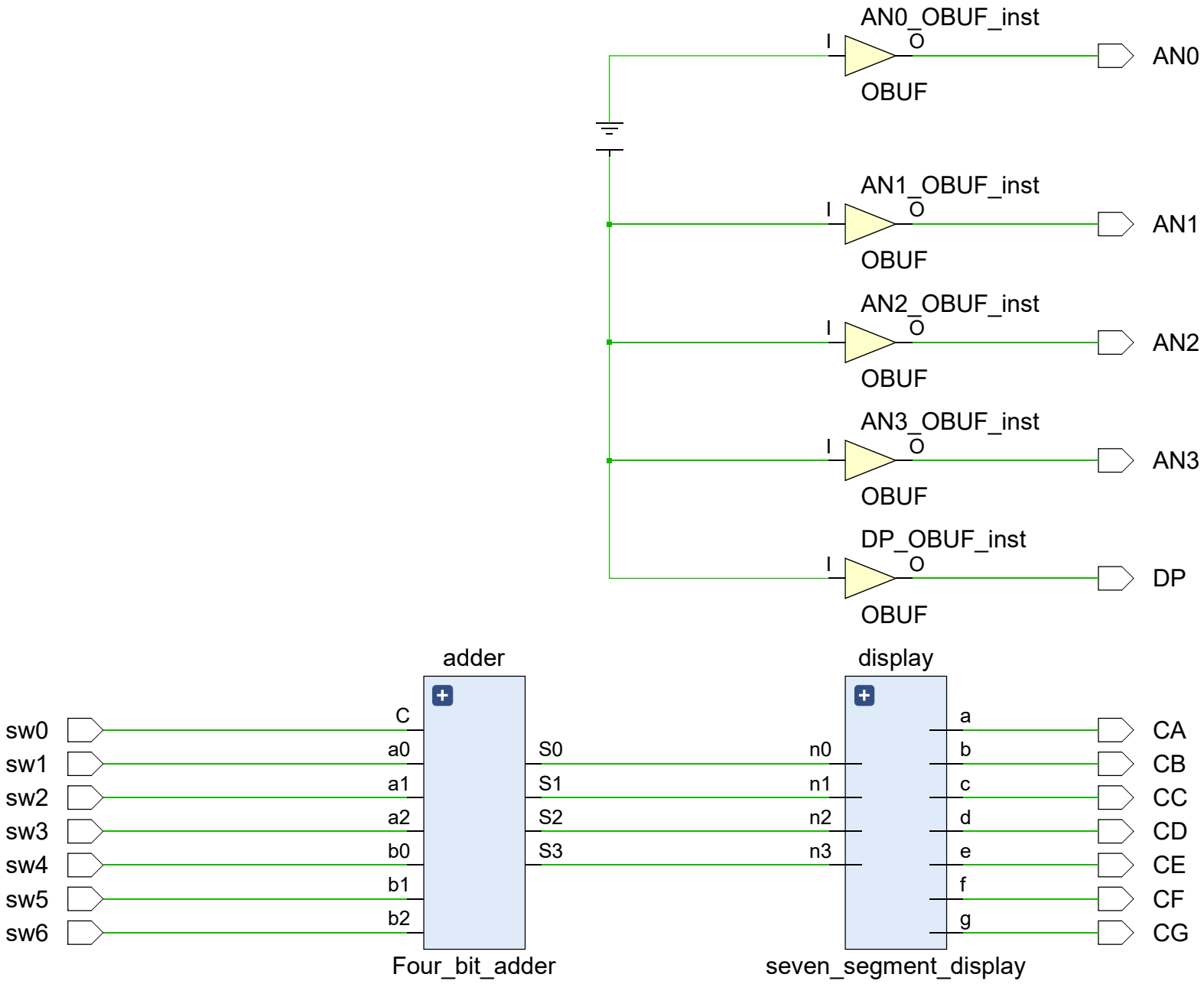
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/06/2020 11:19:06 PM
// Design Name:
// Module Name: seven_segment_display
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module seven_segment_display(
    input n0,
    input n1,
    input n2,
    input n3,
    output a,
    output b,
    output c,
    output d,
    output e,
    output f,
    output g
    );
    wire zero, one, two, three, four, five, six, seven, eight, nine, let_a, let_b,
let_c, let_d, let_e, let_f;
    assign zero = (~n3 & ~n2 & ~n1 & ~n0);
    assign one = (~n3 & ~n2 & ~n1 & n0);
    assign two = (~n3 & ~n2 & n1 & ~n0);
    assign three = (~n3 & ~n2 & n1 & n0);
    assign four = (~n3 & n2 & ~n1 & ~n0);
    assign five = (~n3 & n2 & ~n1 & n0);
    assign six = (~n3 & n2 & n1 & ~n0);
    assign seven = (~n3 & n2 & n1 & n0);
    assign eight = (n3 & ~n2 & ~n1 & ~n0);
```

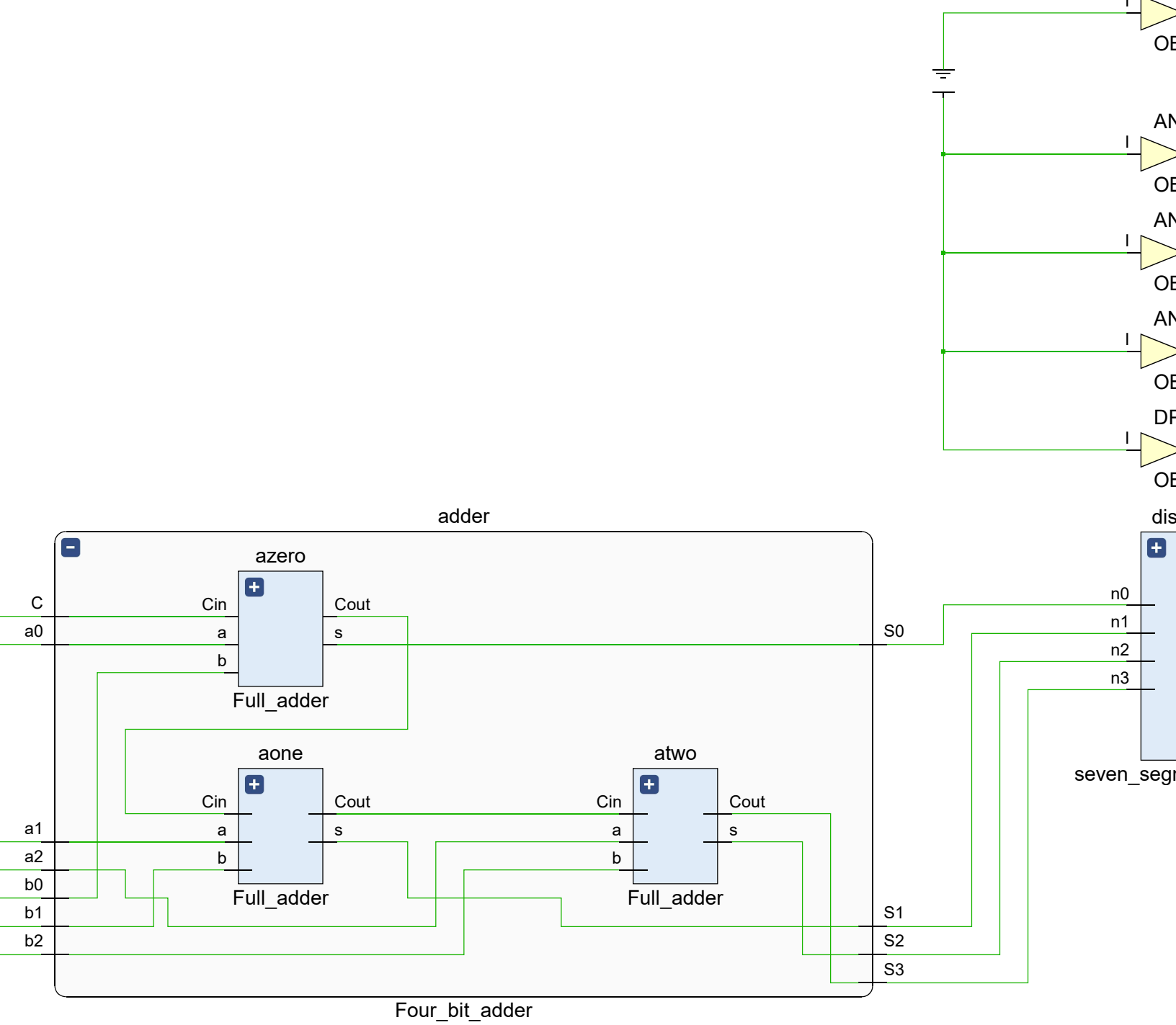
```
    assign nine = (n3 & ~n2 & ~n1 & n0);
    assign let_a = (n3 & ~n2 & n1 & ~n0);
    assign let_b = (n3 & ~n2 & n1 & n0);
    assign let_c = (n3 & n2 & ~n1 & ~n0);
    assign let_d = (n3 & n2 & ~n1 & n0);
    assign let_e = (n3 & n2 & n1 & ~n0);
    assign let_f = (n3 & n2 & n1 & n0);

    assign a = ~(zero | two | three | five | six | seven | eight | nine | let_a |
let_c | let_e | let_f);
    assign b = ~(zero | one | two | three | four | seven | eight | nine | let_a |
let_d);
    assign c = ~(zero | one | three | four | five | six | seven | eight | nine |
let_a | let_b | let_d);
    assign d = ~(zero | two | three | five | six | eight | let_b | let_c | let_d |
let_e);
    assign e = ~(zero | two | six | eight | let_a | let_b | let_c | let_d | let_e
|let_f);
    assign f = ~(zero | four | five | six | eight | nine | let_a | let_b | let_c |
let_e | let_f);
    assign g = ~(two | three | four | five | six | eight | nine | let_a | let_b |
let_d | let_e | let_f);
endmodule
```

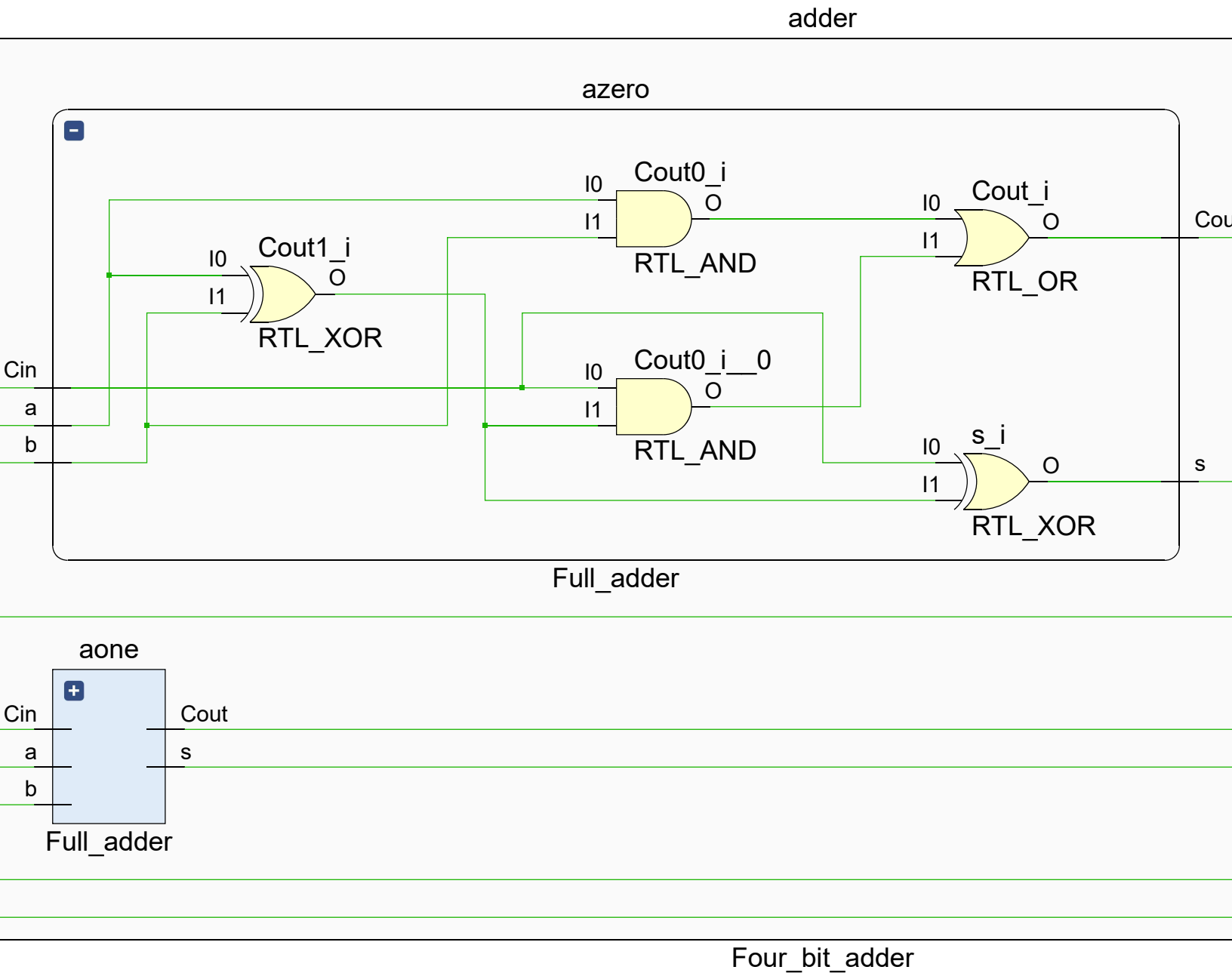
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/06/2020 06:39:54 PM
// Design Name:
// Module Name: Four_bit_adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Four_bit_adder(
    input C,
    input a0,
    input a1,
    input a2,
    input b0,
    input b1,
    input b2,
    output S0,
    output S1,
    output S2,
    output S3
    );
    wire t0, t1;

    Full_adder azero (.Cin(C), .a(a0), .b(b0), .s(S0), .Cout(t0));
    Full_adder aone (.Cin(t0), .a(a1), .b(b1), .s(S1), .Cout(t1));
    Full_adder atwo (.Cin(t1), .a(a2), .b(b2), .s(S2), .Cout(S3));

endmodule
```

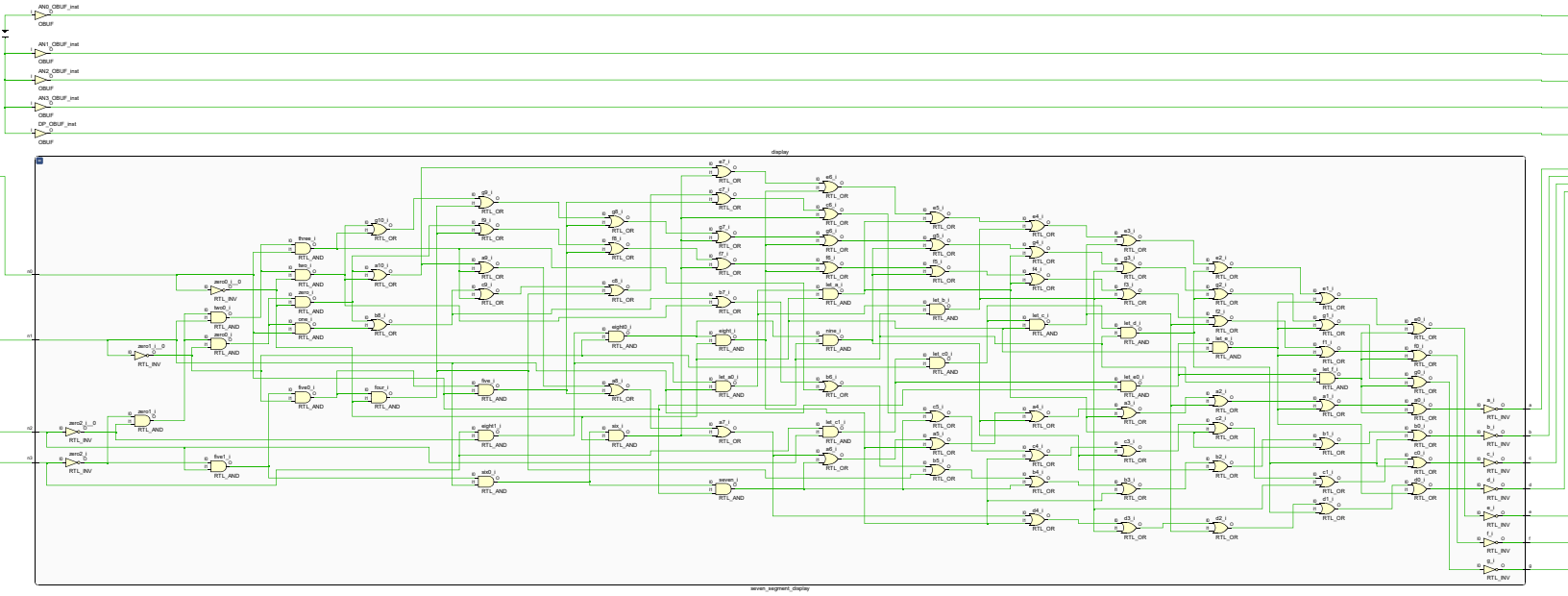
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/06/2020 06:23:41 PM
// Design Name:
// Module Name: Full_adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Full_adder(
    input Cin,
    output Cout,
    output s,
    input a,
    input b
    );

    assign s = Cin ^ (a ^ b);
    assign Cout = (a & b) | (Cin & (a ^ b));
endmodule
```


lab2_7Seg_testbench_behav.wcfg
1,800.000 ns
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
1,000 ns
1,200 ns
1,400 ns
1,600 ns
1,800 ns
D7Seg3[7:0]
20
D7Seg2[7:0]
20
D7Seg1[7:0]
20
D7Seg0[7:0]
F
0 1 2 3 4 5 6 7 8 9 A B C D E F
sw0
1
sw1
1
sw2
1
sw3
1
sw4
1
sw5
1
sw6
1
CA
0
CB
1
CC
1
CD
1
CE
0
CF
0
CG
0
DP
1
AN0
0
AN1
1
AN2
1


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
C
a0
a1
a2
b0
b1
b2
S0
S1
S2
S3
Four_bit_adder
display
n0
n1
n2
n3
a
b
c
d
e
f
g
seven_segment_display
sw0
sw1
sw2
sw3
sw4
sw5
sw6
CA
CB
CC
CD
CE
CF
CG


adder
azero
Cin
Cout
a
s
b
Full_adder
C
a0
aone
atwo
a1
a2
b0
b1
b2
S0
S1
S2
S3
Four_bit_adder
n0
n1
n2
n3
seven_segr
I
O
AN
DP


adder
azero
Cout0_i
I0
I1
O
RTL_AND
Cout_i
RTL_OR
Cout1_i
RTL_XOR
Cin
a
b
Cout0_i__0
s_i
s
Full_adder
aone
Cout
Four_bit_adder


AN0_OBUF_inst
OBUF
AN1_OBUF_inst
OBUF
AN2_OBUF_inst
OBUF
AN3_OBUF_inst
OBUF
DP_OBUF_inst
OBUF
display
seven_segment_display
RTL_AND
RTL_OR
RTL_INV