Ce 100 Lab 2

Gavin Chen
Lab section 2
4/20/18

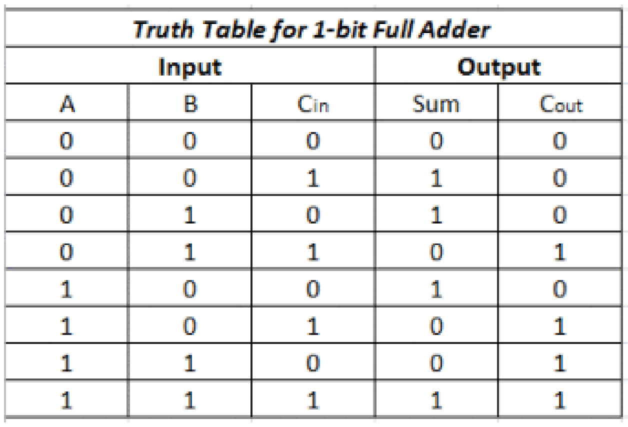
7-Segment display and module programming

Description:

In this lab we investigated the idea of programming a logical circuit using individualized modules. We did so through the implementation of a 3 bit calculator broken into two parts. One instance was the calculator section that performed full bit addition on 2 inputed bit numbers, facilitated through the on board switches. The second was a module which facilitated interaction with the on board 7-segment display. The end result was a fully functional 3 bit calculator which took input through switches and output the sum on the LED display.

Method:

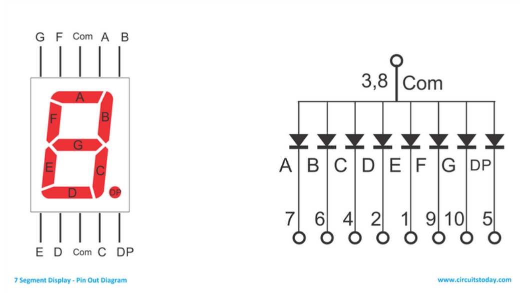
This lab contained 2 parts, the calculator implementation using logical gates, and the 7-segment display module. The 3 bit adder was implemented by first creating the truth table for a full bit adder, presented to the right [1]. We then calculated the sum of products formula from the table, and reduced it down using boolean algebra. The final formulas output the desired truth table:



Truth Table for 1-bit Full Adder				
Input			Output	
A	B	Cin	Sum	Cout
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

Sum: A xor B xor C
Carry bit: (A xor B) ^ C or (A ^ B)

With this formula we then created a full bit adder module that implemented this logic using 3 inputs A, B, and Cin; and 2 outputs the sum bit and the carry out bit. This base module allowed us to implement a 3 bit adder by "chaining" 3 adders together, feeding the sum bit to the display and the carry out bit into the carry in bit for the subsequent adder. We ended up with a higher level module that implemented 3 full adders to create a 3 bit full adder.



The second section of the lab was centered on interfacing with the 7 segment LED display using another module. Due to the nature of the digit display, each "segment" shared common cathodes but had individualized anodes, as shown in the figure to the left [2]. With this design the wiring of the cathodes could be toggled to create different digits/displays. By wiring the anode for a specific segment, that display/digit would appear on that digit. In other words, we choose the segment display using shared cathodes, and choose the digit to display by selecting an anode. Synchronizing the toggling of anodes for individual segments and the shared cathodes allows for the showing of a different number on each digit, despite the fact that all segments share the same cathodes. With this design the number of pins required to control the display is reduced dramatically.

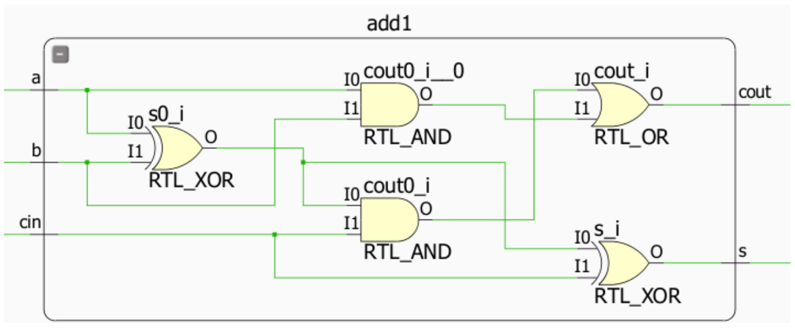
We implemented the display in the same fashion as we implemented full bit adder. Firstly the sequence of LEDs needed to display a number were calculated, i.e. to display the numerical digit one cathodes B and C were toggled to ground. With the cathode sequences configured, we then created a truth table with a 4 bit input and 7 bit output. The 4 bit input represented the hexadecimal range of 0-15, and the corresponding output represented the ground connections for the 7 pins (A-G) that displayed the correct number on the segment display. We then derived the sum of products formula for each output cathode and implemented that logic in our segment interface module. The truth table and boolean formulas are presented in the appendix section succeeding this lab report.

1"1 Bit – Adders – CODE STALL."
2"Interface 7 Segment Display to AVR Atmega32 Using CD4511B."

The final section of the lab was to implement all individual modules together to create a single fully functional calculator. This was done by creating a top module with instances of a full 3 bit adder and a 7 segment display module. The sum bits and most significant carry out bit were wired into the 7 segment display as input with switches sw{0} to sw{6} comprising the two 3 bit numbers and first carry in bit. The output of the segment display module was connected to the cathode of the rightmost segment display on the Bayes3 board, and all anodes save the rightmost segment were disconnected. A schematic of the top module is presented post-lab. The pins of the Artix7 FPGA that correspond to the slide switch inputs are V15 – V17, and W13 – W17. Setting these pin properties in the configuration file allowed us to access these switches for input. The pins used in the 7 segment display are W6 – W7, U5, U7, U8, V5, V7, and V8 for the common cathode portions of the 7 segment display, and U2 corresponding to the anode pin of the rightmost digit on the display.

Results:

The final outcome of the lab produced a fully functional 3 bit adder that took in as input two 3 bit numbers and a carry in bit, and output the result on a 7-segment display. The elaborated design for the full bit adder we implemented is presented in the adjacent figure. The maximum path between input bit and output bit for our full adder is 3 gates, however the longest path for a carry in bit is 2 gates. In the case of a 3 bit adder, the configuration where a bit will experience the longest possible path involves the initial input bit for either A or B, i.e. an input bit for the rightmost digit. In the case where all other input bits for A and B are high, the initial input bit must ripple through the whole circuit as a carry bit before "landing" at its final output. In the case of a 3 bit adder the bit will have to ripple through 3 adders before landing at the overflow bit place. Since each adder subsequent to the initial will treat the bit as a carry in, the longest path for an input bit in a 3 bit adder is 3 + 2(2) = 7 gates. As the number of carry in gates depends on the number of full adders in our circuit, we can generalize the longest path as 3 + 2(n-1) gates, where n is the number of adders.

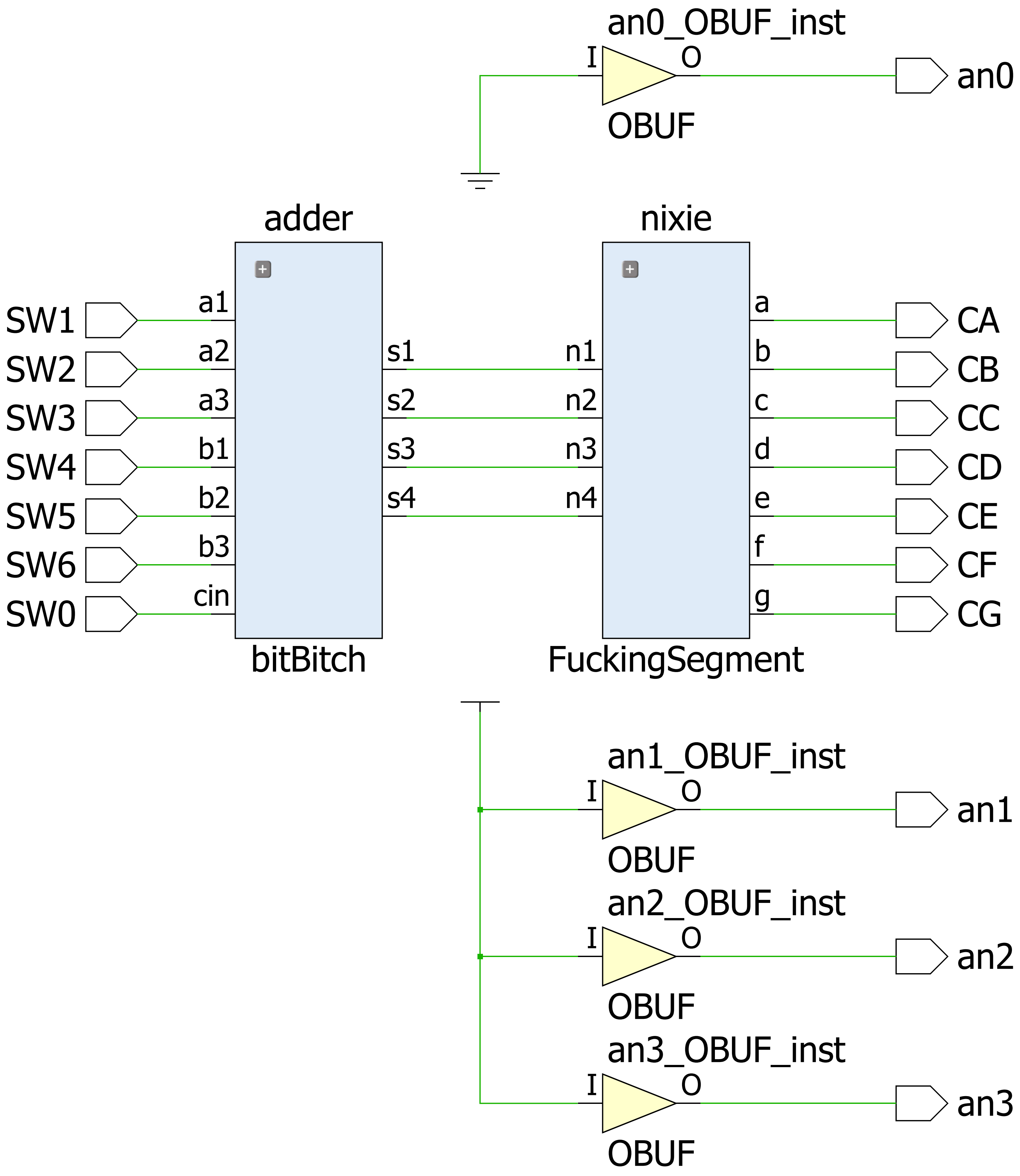


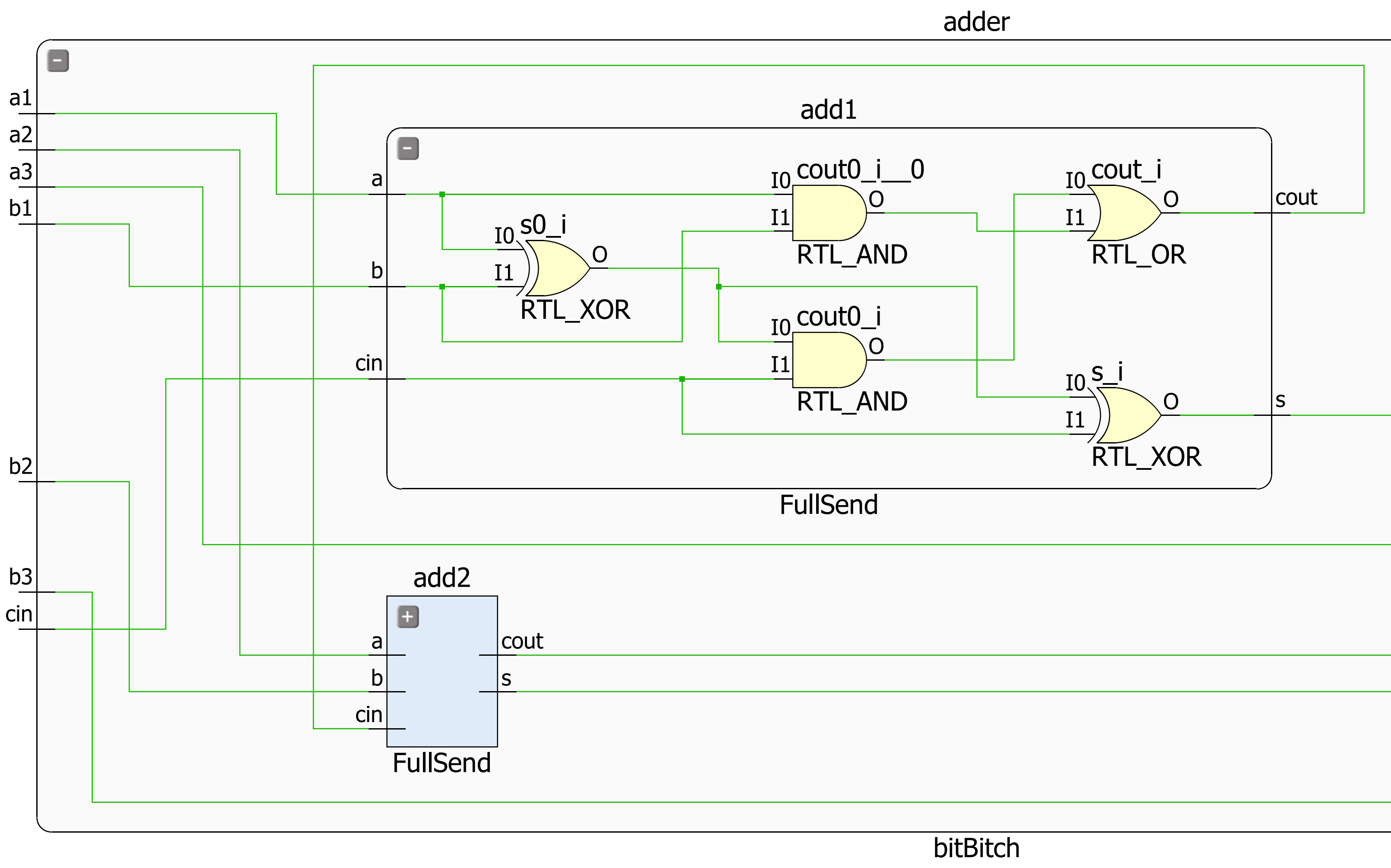
The corresponding output of the full bit adder is then piped into the 7 segment display converter to convert the input 4 bits into a predetermined 7 bit output. Since the connection to ground was set for a low input, the truth table for the display was inverted. Adjusting for this and converting the truth table into logic gates, the output performed as expected and all possible 16 hexadecimal digits 0-F were displayed on the 7 segment display. We then tested the implemented circuit using Verilog simulation settings and inputs. We modified a test script that toggled the switches such as to yield summations resulting in all numbers 0-F to test the output of the adder and 7 segment display module. The total number of possible 7 bit inputs for our adder is 2^7 = 64 possible permutations. Our test script was focused on the functionality of the adder and the 7 segment display. As such, we only tested 16 possible inputs, corresponding to summations yielding the numbers 0-F. Our simulation only tested 16/64 = 25% of the total inputs.

Conclusion:

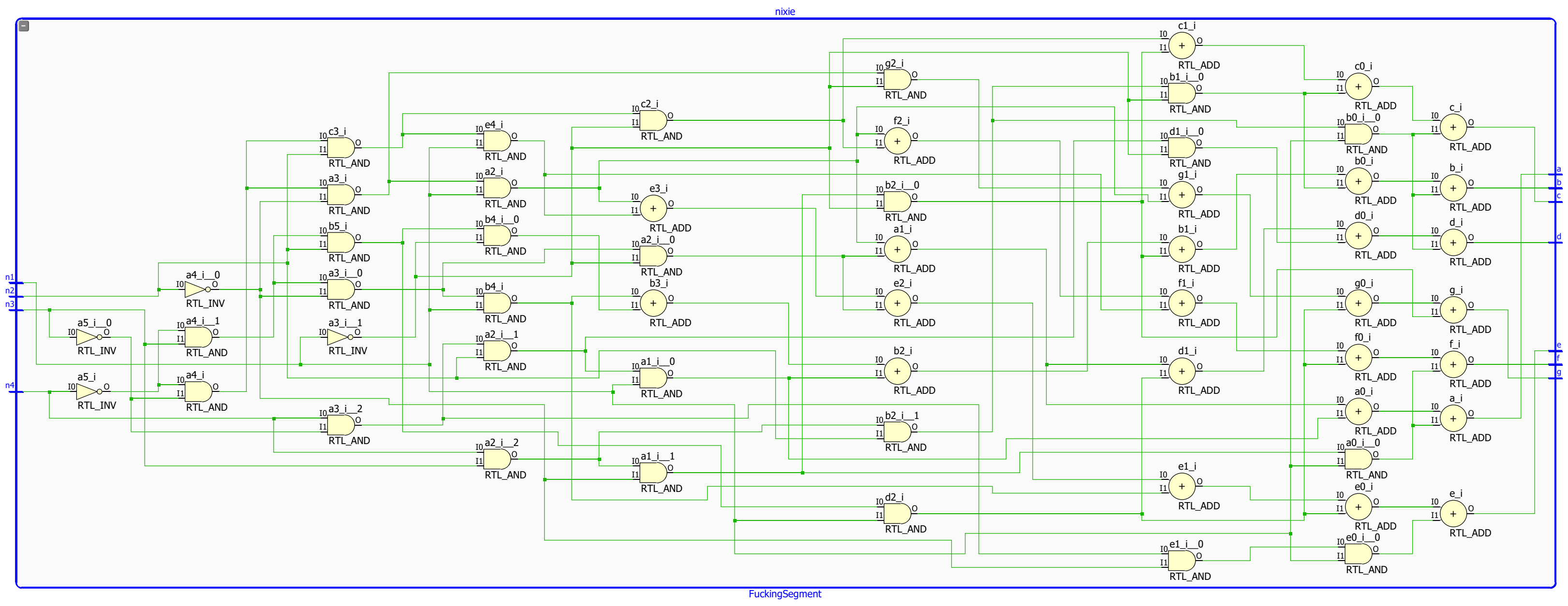
This lab taught us the concept of modules to create larger and more elaborate designs using successive smaller modules as building blocks. We implemented an optimized full adder module using logic gates, which we then expanded into a 3 bit ripple adder to perform full binary addition. We also investigated the idea of the 7 segment display, and became familiar with converting a truth table containing the sequence of highlighted segments corresponding to numbers, and translated it into a logic circuit that depended on a 4 bit input. The schematic for the design is included in the Appendix. Finally, we familiarized ourselves with the Verilog simulations, inputing our desired input as a waveform and observing the output. This process will help us analyze much more complex and embedded circuits in the future much more quickly then physically inputing the tests.

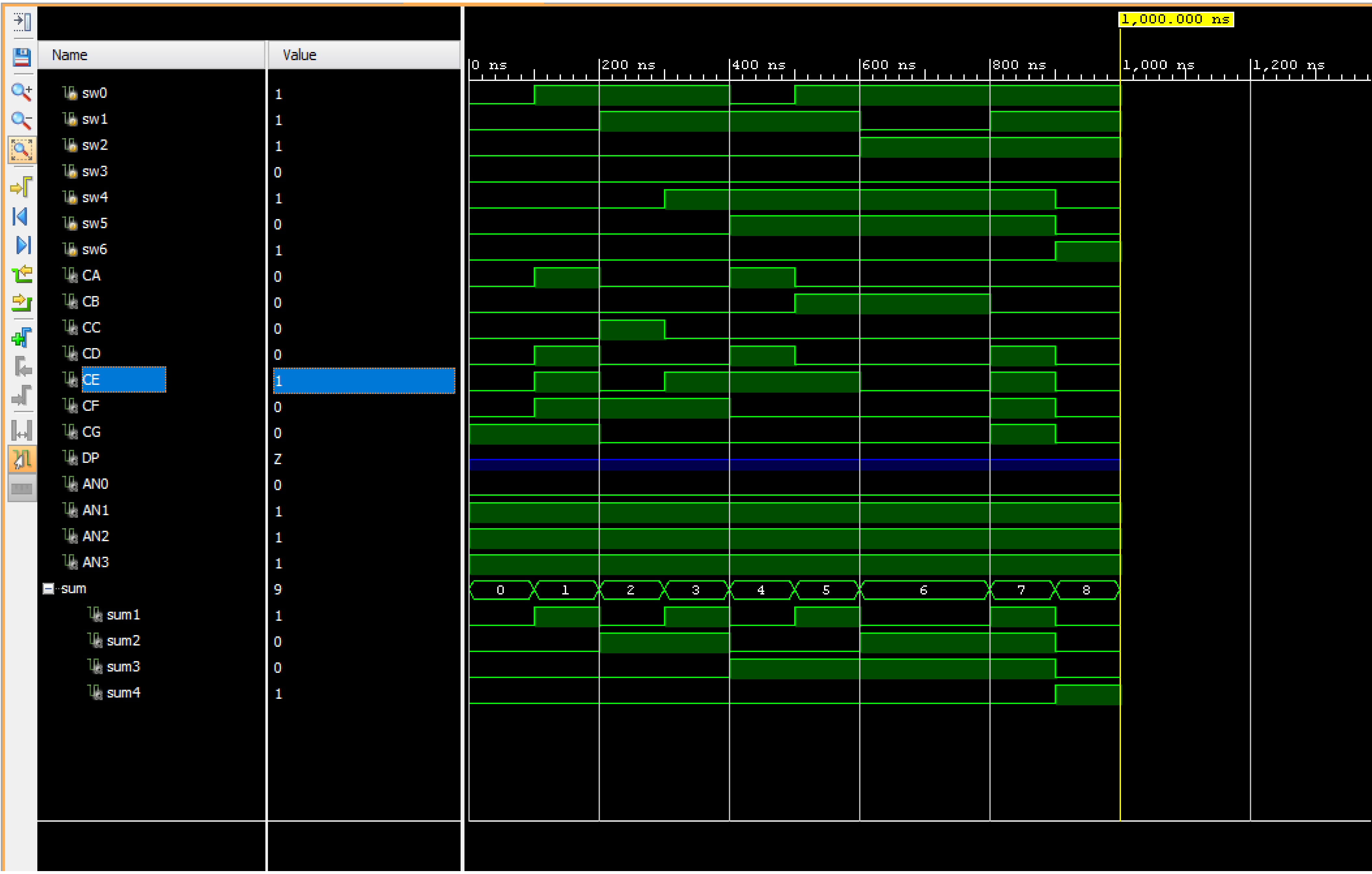
Appendix Section


an0_OBUF_inst
I
O
an0
OBUF
adder
nixie
SW1
SW2
SW3
SW4
SW5
SW6
SW0
a1
a2
a3
b1
b2
b3
cin
s1
s2
s3
s4
n1
n2
n3
n4
a
b
c
d
e
f
g
CA
CB
CC
CD
CE
CF
CG
bitBitch
FuckingSegment
an1_OBUF_inst
an1
an2_OBUF_inst
an2
an3_OBUF_inst
an3



adder
add1
a1
a2
a3
b1
a
b
cin
I0
I1
O
s0_i
RTL_XOR
cout0_i__0
RTL_AND
cout0_i
RTL_AND
cout_i
RTL_OR
s_i
RTL_XOR
cout
s
FullSend
b2
b3
cin
add2
a
b
cin
cout
s
FullSend
bitBitch


nixie
n1
n2
n3
n4
a5_i__0
RTL_INV
a5_i
RTL_INV
a4_i__0
RTL_INV
a4_i__1
RTL_AND
a4_i
RTL_AND
c3_i
RTL_AND
a3_i
RTL_AND
b5_i
RTL_AND
a3_i__0
RTL_AND
a3_i__1
RTL_INV
a3_i__2
RTL_AND
e4_i
RTL_AND
a2_i
RTL_AND
b4_i__0
RTL_AND
b4_i
RTL_AND
a2_i__1
RTL_AND
a2_i__2
RTL_AND
c2_i
RTL_AND
e3_i
RTL_ADD
a2_i__0
RTL_AND
b3_i
RTL_ADD
a1_i__0
RTL_AND
a1_i__1
RTL_AND
g2_i
RTL_AND
f2_i
RTL_ADD
b2_i__0
RTL_AND
a1_i
RTL_ADD
e2_i
RTL_ADD
b2_i
RTL_ADD
b2_i__1
RTL_AND
d2_i
RTL_AND
c1_i
RTL_ADD
b1_i__0
RTL_AND
d1_i__0
RTL_AND
g1_i
RTL_ADD
b1_i
RTL_ADD
f1_i
RTL_ADD
d1_i
RTL_ADD
e1_i
RTL_ADD
e1_i__0
RTL_AND
c0_i
RTL_ADD
b0_i__0
RTL_AND
b0_i
RTL_ADD
d0_i
RTL_ADD
g0_i
RTL_ADD
f0_i
RTL_ADD
a0_i
RTL_ADD
a0_i__0
RTL_AND
e0_i
RTL_ADD
e0_i__0
RTL_AND
c_i
RTL_ADD
b_i
RTL_ADD
d_i
RTL_ADD
g_i
RTL_ADD
f_i
RTL_ADD
a_i
RTL_ADD
e_i
RTL_ADD
a
b
c
d
e
f
g
FuckingSegment



Name	Value
sw0	1
sw1	1
sw2	1
sw3	0
sw4	1
sw5	0
sw6	1
CA	0
CB	0
CC	0
CD	0
CE	1
CF	0
CG	0
DP	Z
AN0	0
AN1	1
AN2	1
AN3	1
sum	9
sum1	1
sum2	0
sum3	0
sum4	1