Ryan Sullivan U73687595

EC311 Lab Report

Objective: Design and test a four bit adder and subtractor using half adders and simple gates, then design and create an Arithmetic Logic Unit using behavioral Verilog code.

Methodology:

1. Adder:
   a. I used a simple truth table to design and verify a valid half adder design also previously discussed in class.
   b. Using this design I put two together to create a simple one bit full adder with carry in option.
   c. By putting four of these one bit full adders in a row and ripple pushing the carry out component to the subsequent carry in component of the next adder, I was able to create a simple four bit adder.
   d. Adding on xor gates to the B input of the four bit adder means the adder can now do one's complement subtraction by flipping each bit of B by using a new input M, signifying minus.
2. ALU:
   a. Creating each subsection of the unit using behavioral Verilog was simply describing the task that needed to be done, for the Multiplier Y = A*B, the Adder Y = A+B, the Bit Shifter Y = A << B, and the Concatenator Y = {A,B}.

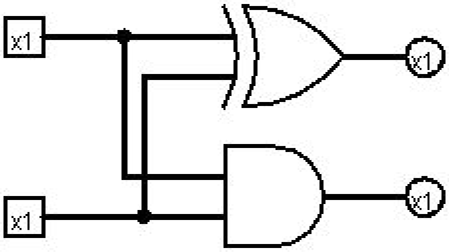
b. Connecting all of these I used a simple multiplexer that accepted a two bit input S, with 00,01,10, and 11 each corresponding to a different ALU function

c. This was accomplished using a logical case statement that assigns the output to any of the outputs of the constituent ALU functions given a certain S value.

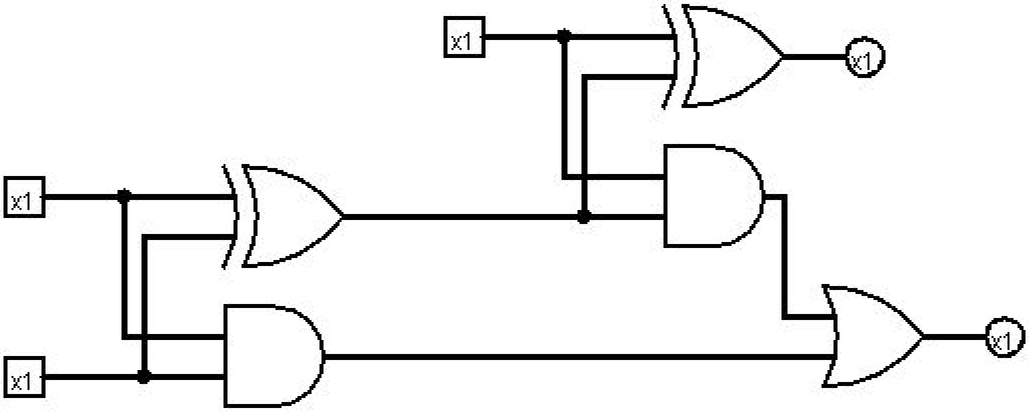
Observations:

- Schematics and Testbenches on final products

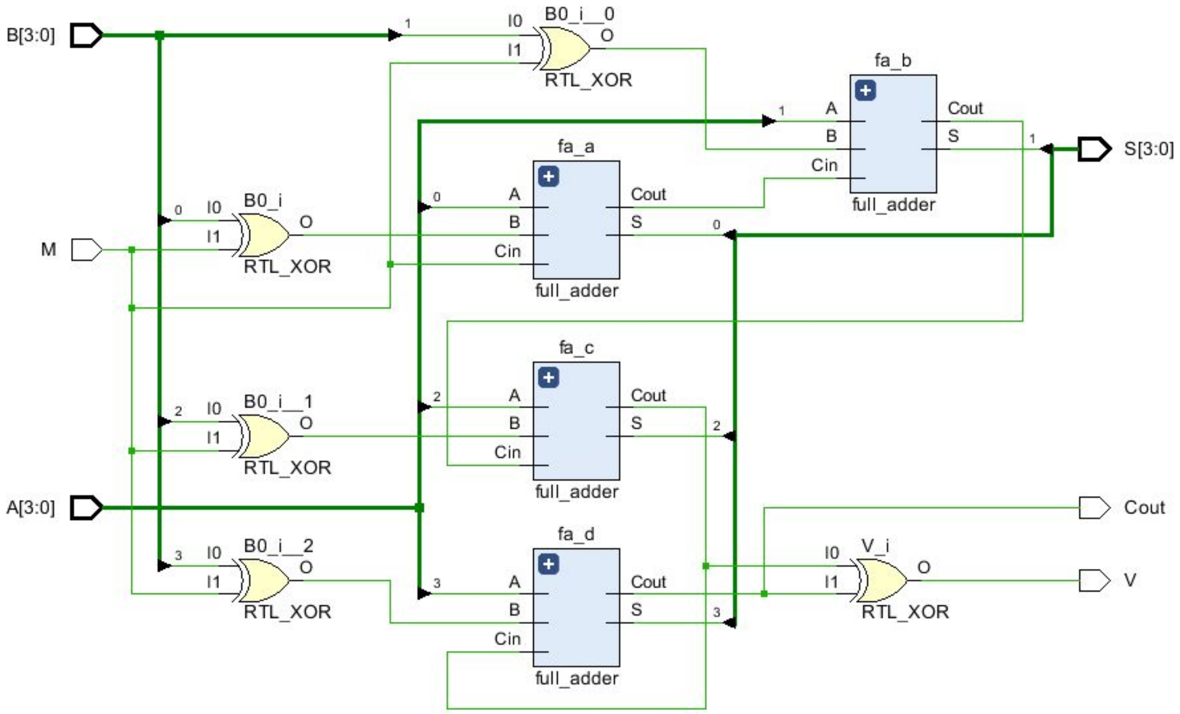
Half Adder:

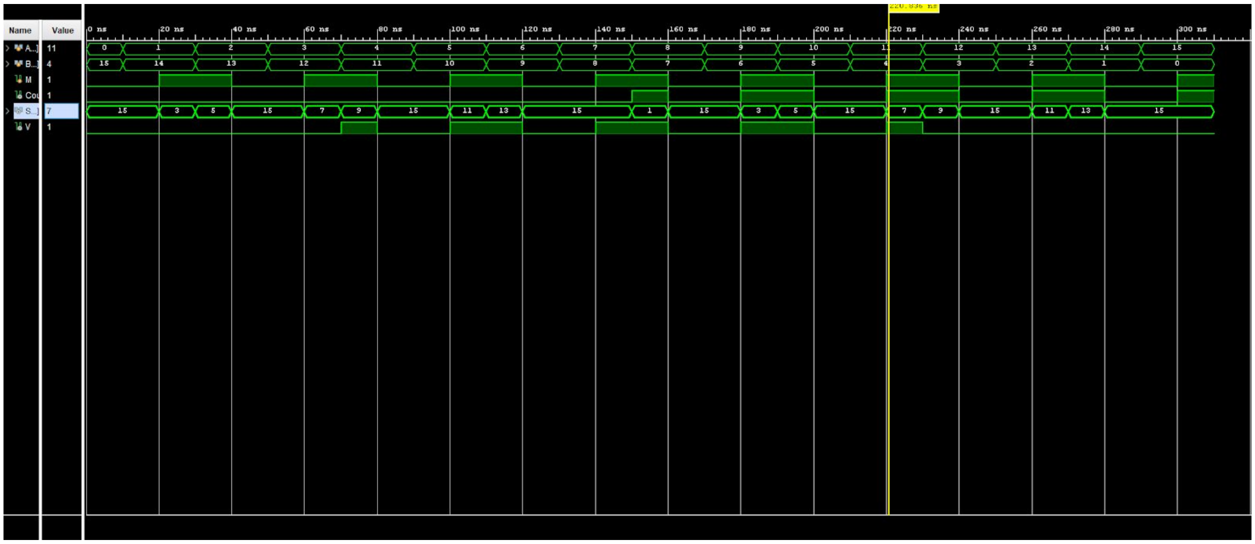


Full Adder:

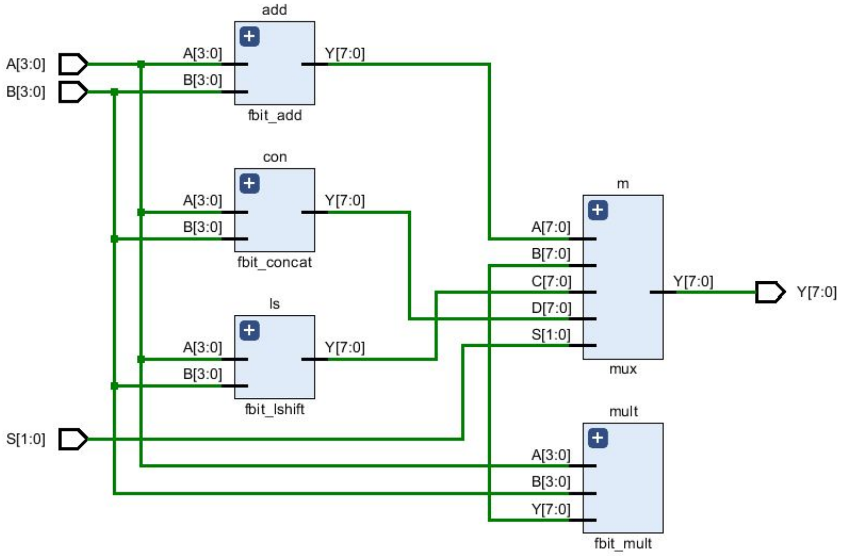


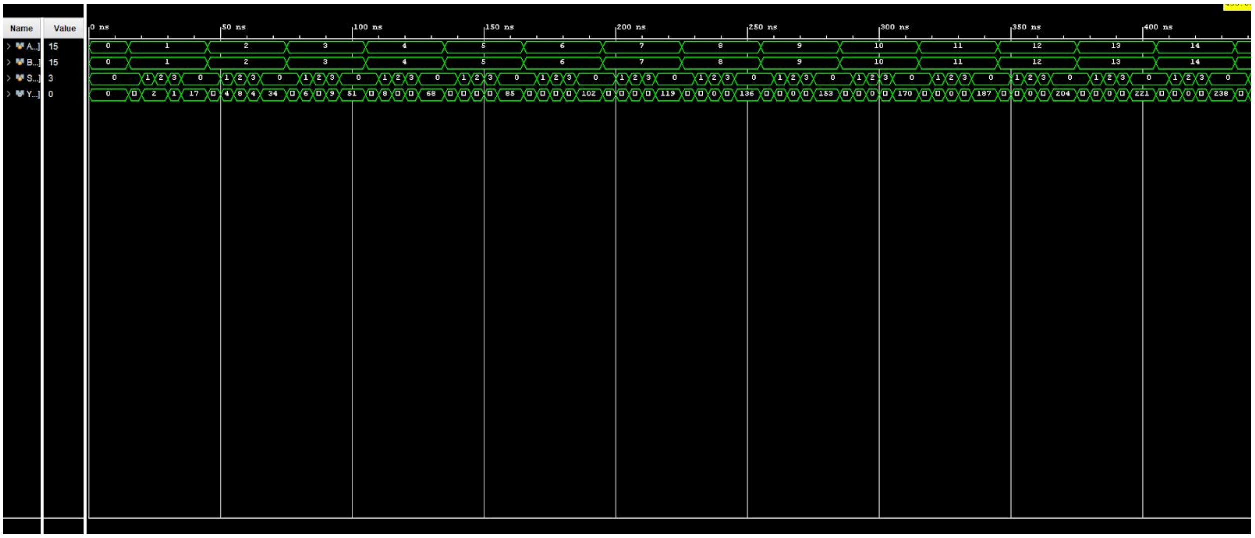
Four-Bit Full Adder-Subtractor:





Arithmetic Logic Unit:


add
A[3:0]
B[3:0]
Y[7:0]
fbit_add
A[3:0]
B[3:0]
con
A[3:0]
B[3:0]
Y[7:0]
fbit_concat
ls
A[3:0]
B[3:0]
Y[7:0]
fbit_lshift
S[1:0]
m
A[7:0]
B[7:0]
C[7:0]
D[7:0]
S[1:0]
Y[7:0]
mux
Y[7:0]
mult
A[3:0]
B[3:0]
Y[7:0]
fbit_mult


Name
Value
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
A..] 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
B..] 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
S..] 3
0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0
Y..] 0
0 □ 2 1 17 □ 4 8 4 34 □ 6 □ 9 51 □ 8 □ □ 68 □ □ □ □ 85 □ □ □ □ 102 □ □ □ □ 119 □ □ 0 □ 136 □ □ 0 □ 153 □ □ 0 □ 170 □ □ 0 □ 187 □ □ 0 □ 204 □ □ 0 □ 221 □ □ 0 □ 238 □