Lab 3 - Arithmetic Logic Unit
CECS 341 – Computer Architecture Organization
Student Antonio Ojeda SID 026076048
Student Bridget Naylor SID 025531413
Professor:Jose Aceves

California State University, Long Beach
College of Engineering

1250 Bellflower Blvd, Long Beach, CA 90840

Lab 2/7/2021

Lab 3- ALU CECS 341 Spring 2021

Goal/Objective:

Our goal for this lab was to model an ALU using verilog. To accomplish this, we need to utilize a table to graph our data and double-check our results.

Technical Description/Steps:(include screenshots and description here)

For this lab assignment we needed to model an ALU's functionality. To model it we used an always block with different cases for the ALUctrl, each case being a different binary value with respect to the given functions. The ALUctrl dictated which operation we would perform on the inputs A and B. Each ALUctrl case utilizes the Negative, Overflow, and Carry-out differently, but they all output the same ALUout result in accordance with the ALUctrl value. The Zero flag is raised if the input value is all zeros (Z=1); if the input value is not all zeros, then Z=0.

Our testbench instantiates the ALU by transferring our ports into the testbench and displays our input and output values on the console and waveform accordingly.

Results:(what we would do differently next time)

Next lab I would like to try coding the lab first without looking at the example introduction videos, and then look at the resources provided.

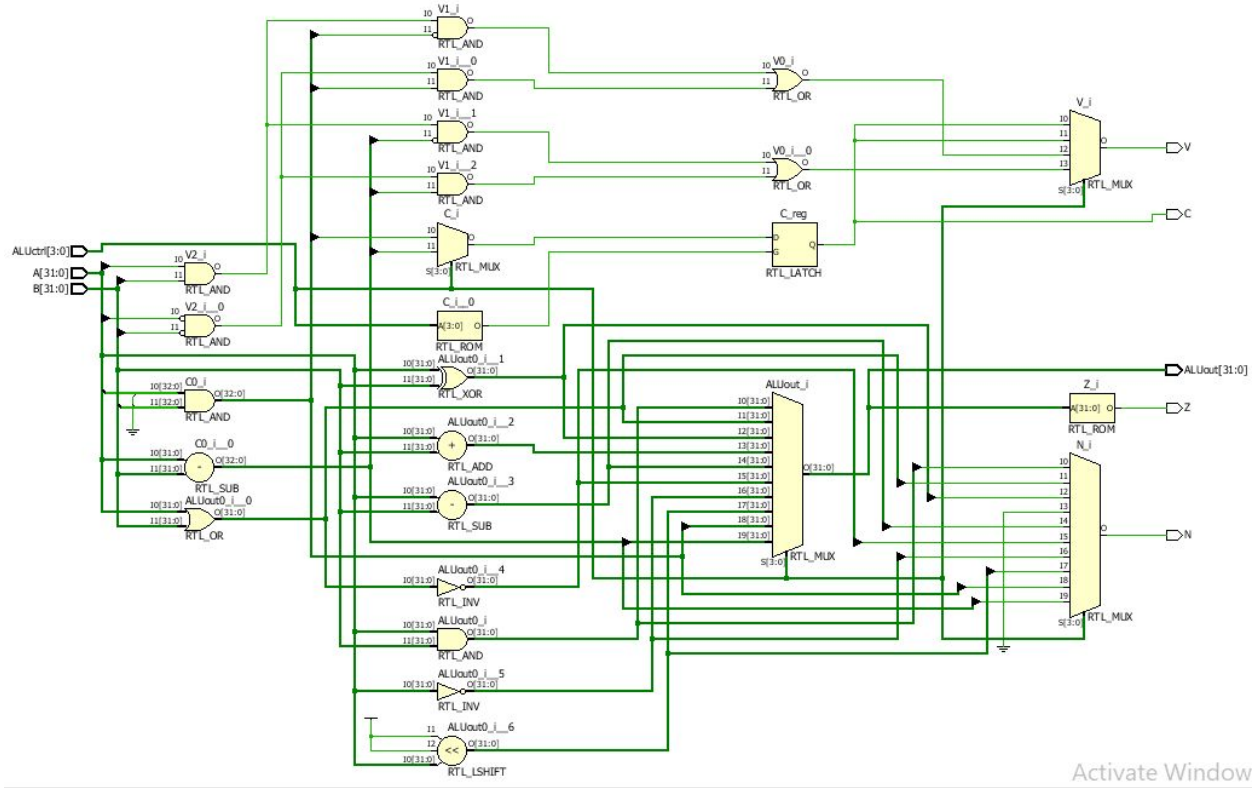
In addition to that, we should prioritize starting the project earlier so that we may have more time to figure out the problems at hand and reach our conclusions before the deadline.
Here is our console output:

```
A=f0f03c3c B=bf0ff5f5 ALUctrl=0000 || ALUout=b0003434 N=1 C=x Z=0 V=x
A=f0f03c3c B=bf0ff5f5 ALUctrl=0001 || ALUout=fffffdfd N=1 C=x Z=0 V=x
A=f0f03c3c B=bf0ff5f5 ALUctrl=0011 || ALUout=4fffc9c9 N=0 C=x Z=0 V=x
A=00000001 B=ffffffff ALUctrl=0010 || ALUout=00000000 N=0 C=1 Z=1 V=1
A=6389754f B=ad5624e6 ALUctrl=0010 || ALUout=10df9a35 N=0 C=1 Z=0 V=1
A=00000001 B=ffffffff ALUctrl=0010 || ALUout=00000000 N=0 C=1 Z=1 V=1
A=6389754f B=ad5624e6 ALUctrl=0010 || ALUout=10df9a35 N=0 C=1 Z=0 V=1
A=ffffffff B=ffffffff ALUctrl=0010 || ALUout=fffffffe N=0 C=1 Z=0 V=1
A=00000000 B=00000001 ALUctrl=0110 || ALUout=ffffffff N=0 C=1 Z=0 V=1
A=f9684783 B=f998d562 ALUctrl=0110 || ALUout=ffcf7221 N=0 C=1 Z=0 V=1
A=f0f03c3c B=bf0ff5f5 ALUctrl=1100 || ALUout=00000202 N=0 C=x Z=0 V=x
A=89bcde34 B=c53bd687 ALUctrl=0111 || ALUout=764321cb N=0 C=x Z=0 V=x
A=f0f03c3c B=bf0ff5f5 ALUctrl=1101 || ALUout=e1e07878 N=1 C=x Z=0 V=x
A=f0f03c3c B=bf0ff5f5 ALUctrl=1111 || ALUout=xxxxxxxx N=x C=x Z=x V=x
A=f0f03c3c B=bf0ff5f5 ALUctrl=1010 || ALUout=b0003231 N=1 C=1 Z=0 V=0
A=f0f03c3c B=bf0ff5f5 ALUctrl=1110 || ALUout=31e04647 N=0 C=0 Z=0 V=1
```

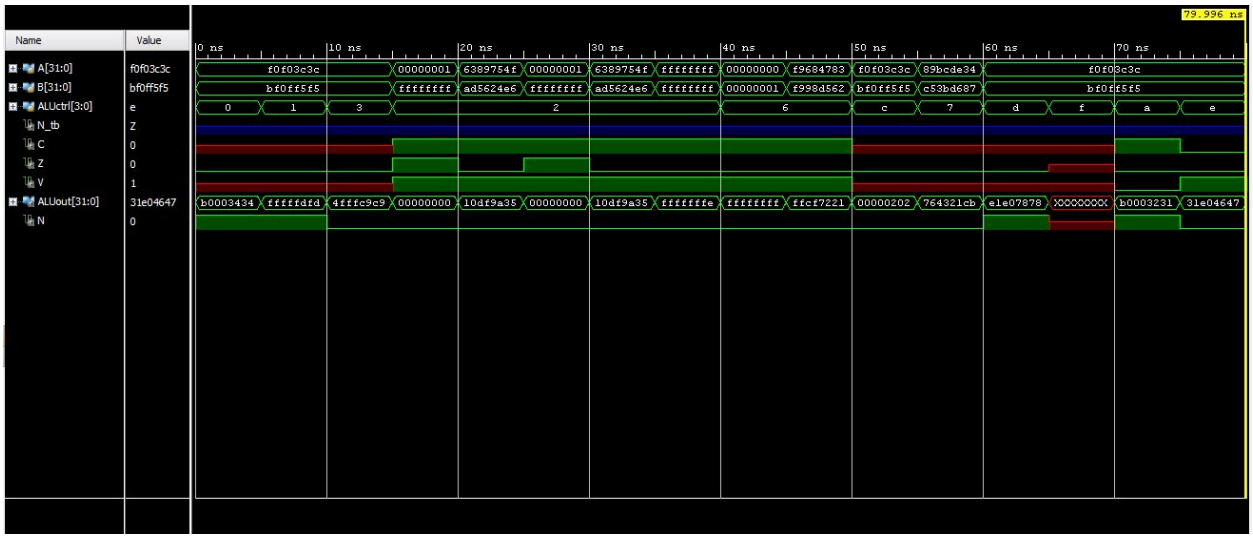
Conclusion:

Building an ALU was an interesting exercise to understand computers more and taught us specifically what is going on inside the computer when we add, subtract, or compare something.
If there are any more arithmetic or logical instructions that could be included in the ALU, then we can limit the usage of defaults so that it only works as a complete failsafe.

RTL Schematic, Captured WaveForm:



Both large muxes are wired to accommodate outputting ALUout and N values, while the smaller mux is accommodated to output the V value. Other parts include logic gates and carry-outs to transfer information according to the specific instruction given.



Some inputs are the same, so they are grouped together with their own ALUcrtl instructions. Each row presents all of their own input and output values, whether it be in 32-bit hexidecimal, 4-bit hexidecimal, or a single binary blip on the waveform for each of the other output flags beside ALUout (which is also present in 32-bit hexidecimal).

Table:

#	A_{hex}	B_{hex}	ALU_{ctrl}	ALU_{outHex}	Z	V	C	N
1	F0F03C3C	BF0FF5F5	0000	B0003434	0	x	x	1
2	F0F03C3C	BF0FF5F5	0001	FFFFFDFD	0	x	x	1
3	F0F03C3C	BF0FF5F5	0011	4FFFC9C9	0	x	x	0
4	00000001	FFFFFFFF	0010	00000000	1	1	1	0
5	6389754F	AD5624E6	0010	10DF9A35	0	1	1	0
6	00000001	FFFFFFFF	0010	00000000	1	1	1	0
7	6389754F	AD5624E6	0010	10DF9A35	0	1	1	0
8	FFFFFFFF	FFFFFFFF	0010	FFFFFFFE	0	1	1	0

9	00000000	00000001	0110	FFFFFFFF	0	1	1	0
10	F9684783	F998D562	0110	FFCF7221	0	1	1	0
11	F0F03C3C	BF0FF5F5	1100	00000202	0	x	x	0
12	89BCDE34	C53BD687	0111	764321CB	0	x	x	0
13	F0F03C3C	BF0FF5F5	1101	E1E07878	0	x	x	1
14	F0F03C3C	BF0FF5F5	1111	xxxxxxxx	x	x	x	x
15	F0F03C3C	BF0FF5F5	1010	B0003231	0	0	1	1
16	F0F03C3C	BF0FF5F5	1110	31E04647	0	1	0	0