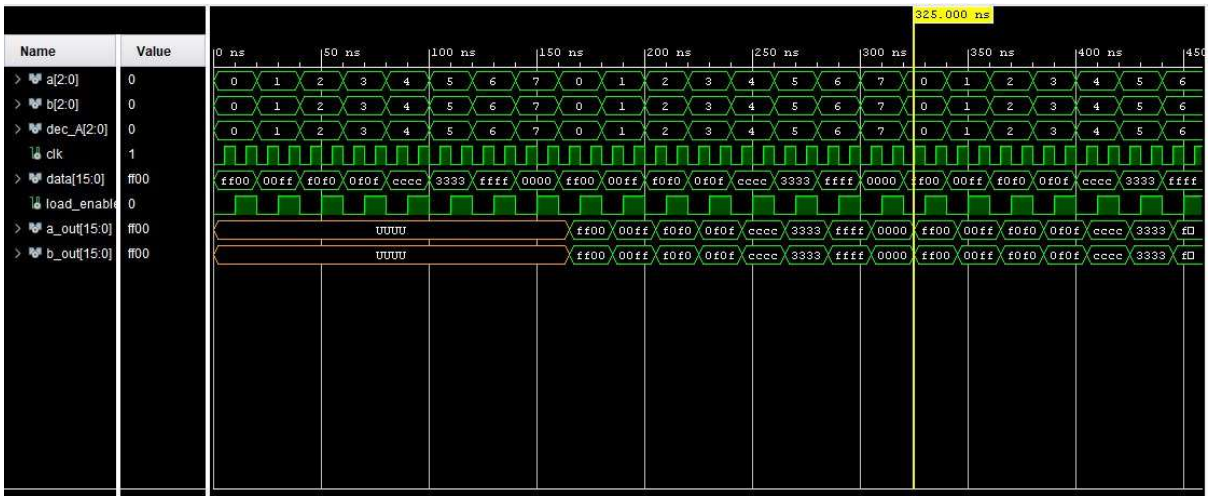
CS2022 Computer Architecture: Project 1B

Contents

Ellen Whelan 17324116

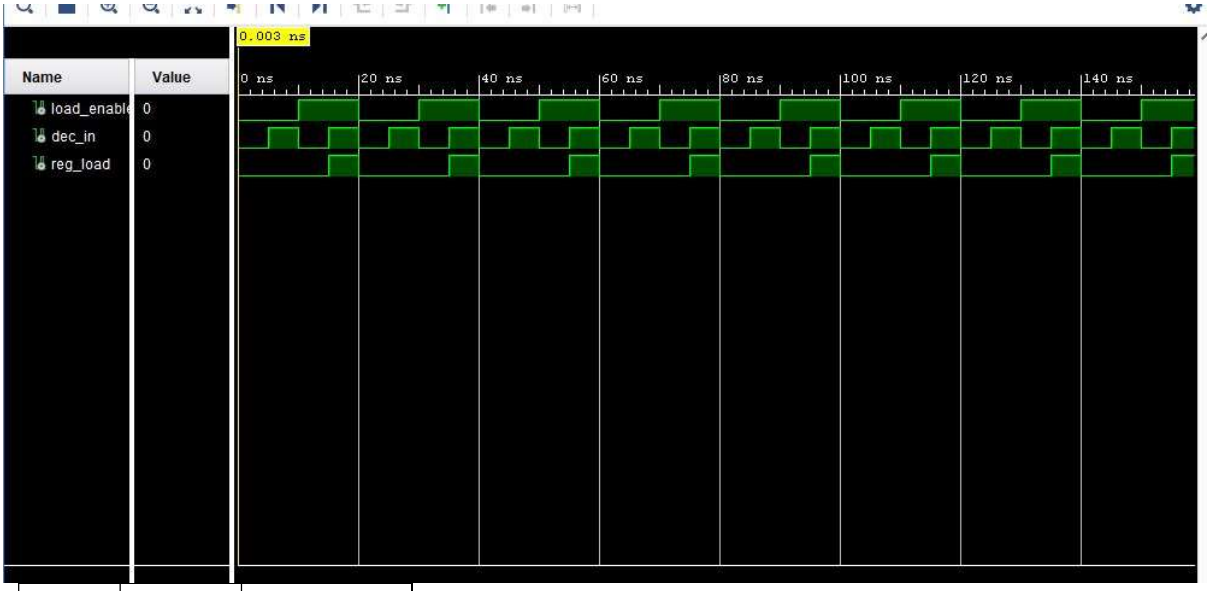
Register File

I altered the register file from the previous assignment (added second mux and B data bus etc) and rewrote the test bench accordingly.



To test the reg file I loaded different data into each register one by one and then set a out and b out to show that data in each register one by one. The a out signal and b out are undefined in the first iteration of the test bench as they are accessing empty registers.

Also in the register file I created and gates and tested them by using the following table.



Input	Load Enable	Output
0	0	0
0	1	0
1	0	0
1	1	1

Function Unit

The function unit contains the ALU and the shifter

ALU

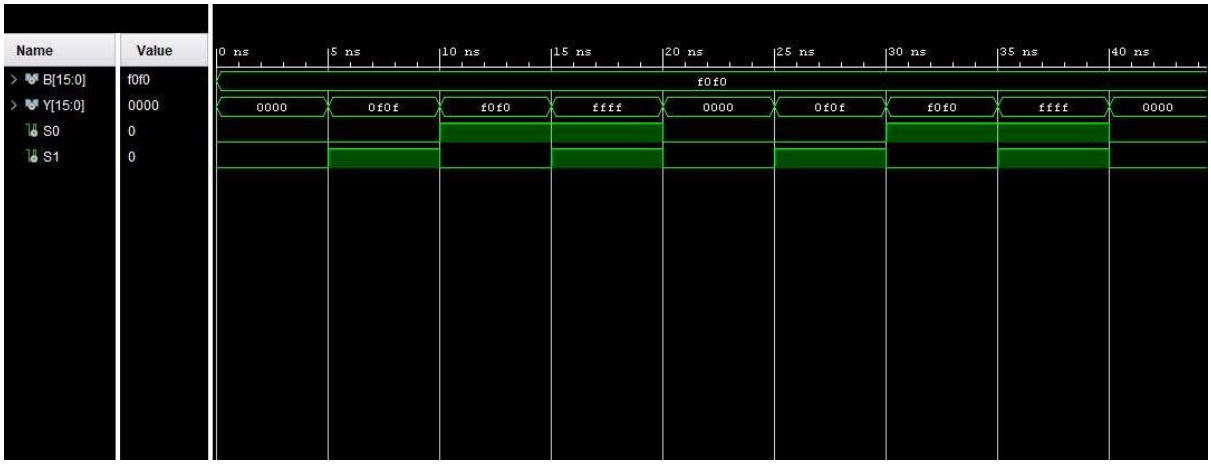
The ALU contains an arithmetic circuit and a logic circuit

Arithmetic Circuit

The ALU contains an arithmetic circuit and a logic circuit. The arithmetic circuit contains B_input_logic and the ripple adder.

B_input_logic

I tested the B input logic by iterating through all the combinations of S0 and S1.

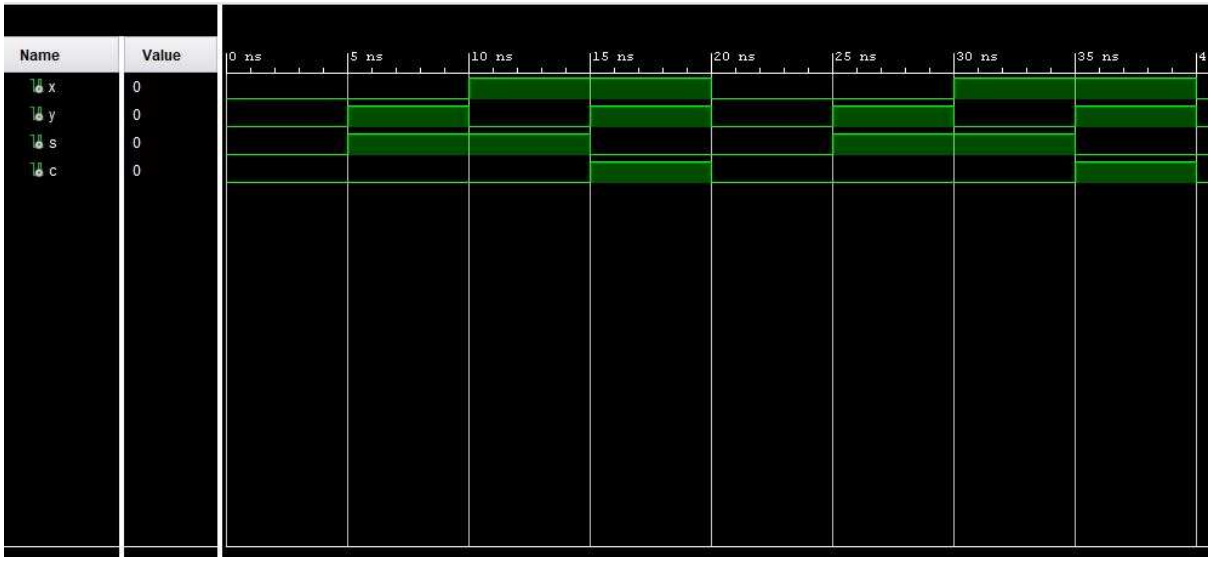


Parallel Adder

Testing the Full Adder

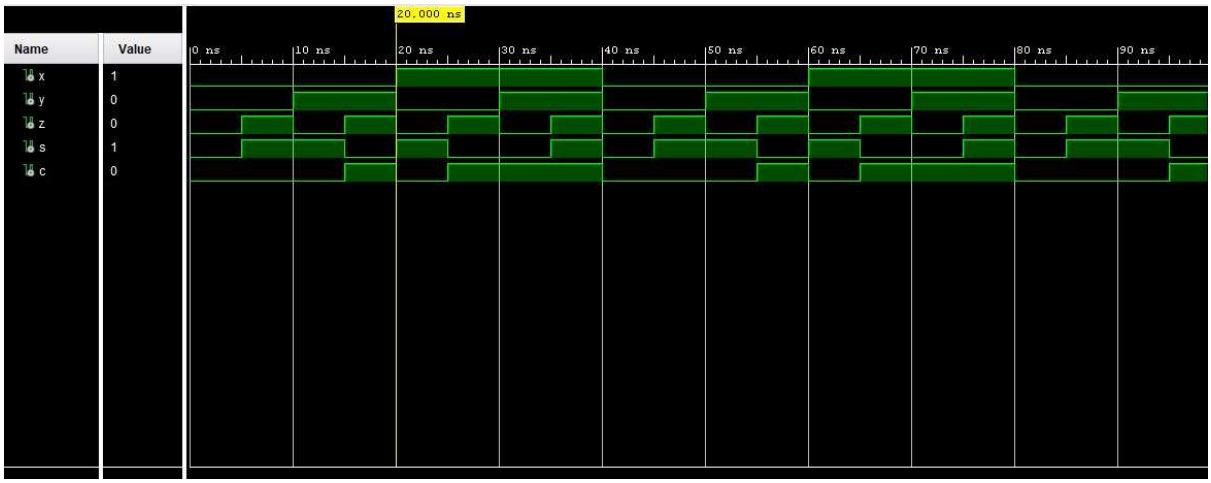
I first wrote and tested a half adder. To test it fully I iterated through all the combinations of X and Y 00,01,10,11, to show the table:

X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

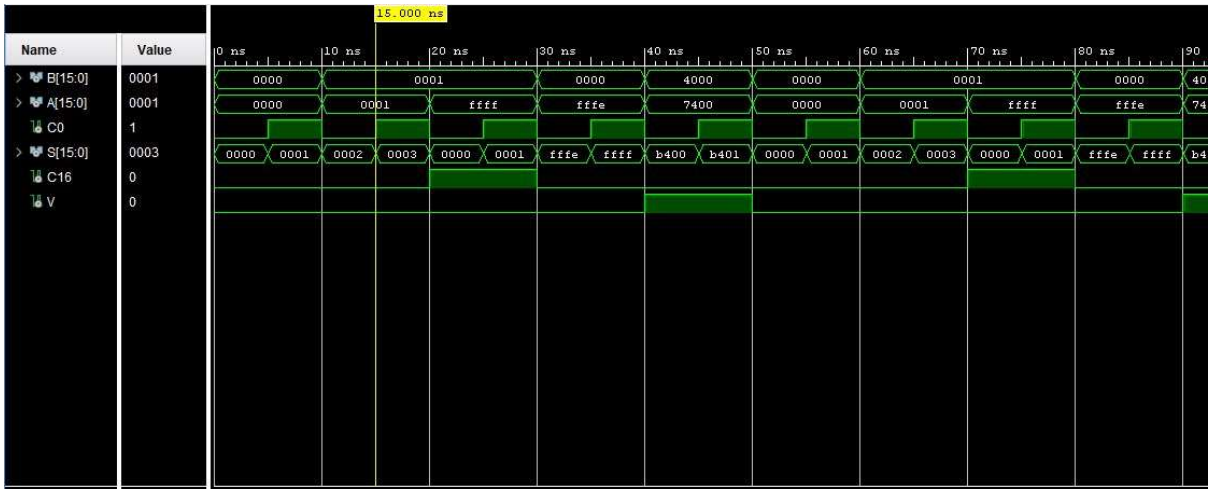


Then I instantiated a half adder twice to make a full adder. I then write the test bench to run through all the possible combinations of inputs x,y and z (000 to 111)

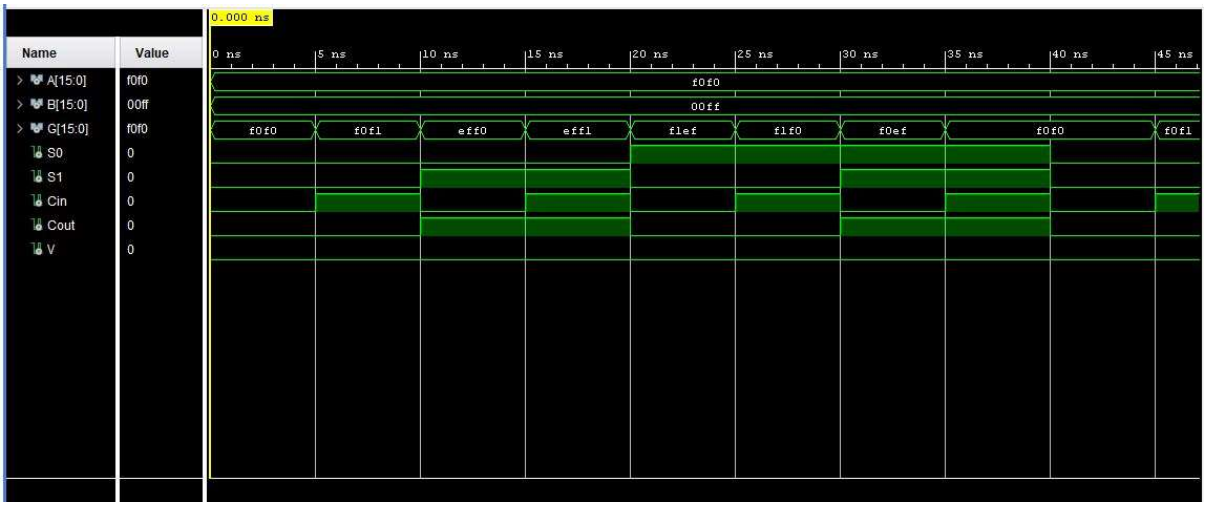
to ensure the correct answer was produced and the correct carry.



Finally I instantiated the full adder 16 times and made the ripple adder. In the test bench I simulated signals to show that carry and overflow were set when needed and not when not. I showed that the sum value was correct with both a carry in of 0 and 1.



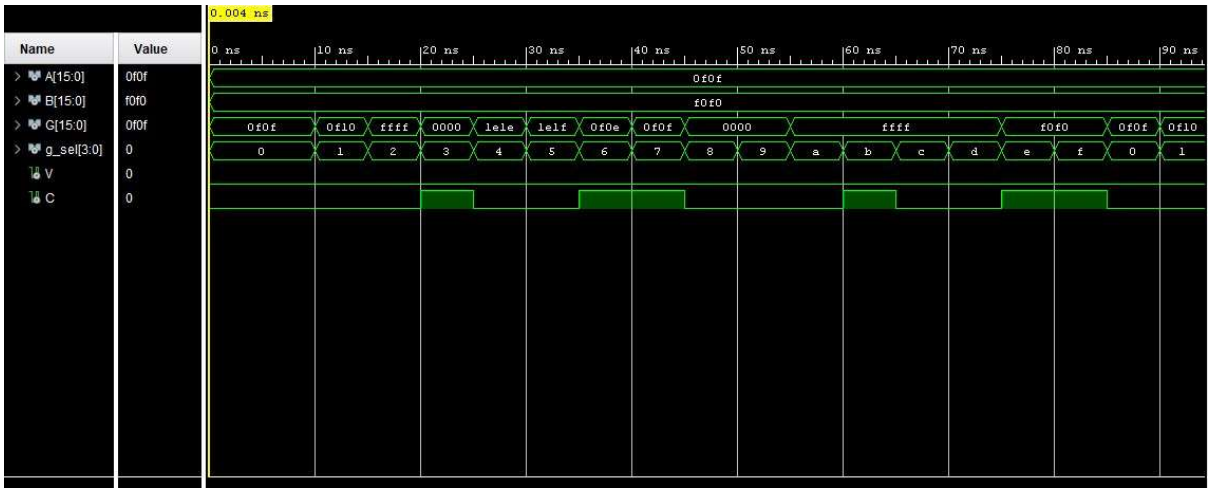
I tested the entire arithmetic circuit by testing each combination of S0 and S1 with a cin of 0 and 1.



Logic Circuit

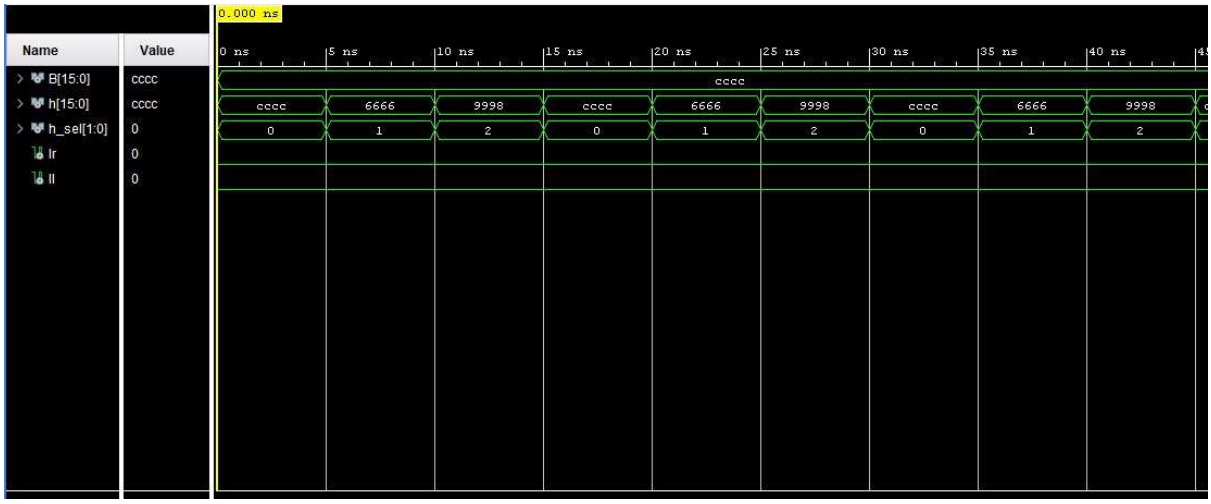
To test the logic circuit I iterated through all the combinations of S0 and S1 to get all the possible logical operations to occur on the values in A and B.

I tested the entire ALU by iterating through all the g_sel options to ensure each operation carries out correctly.

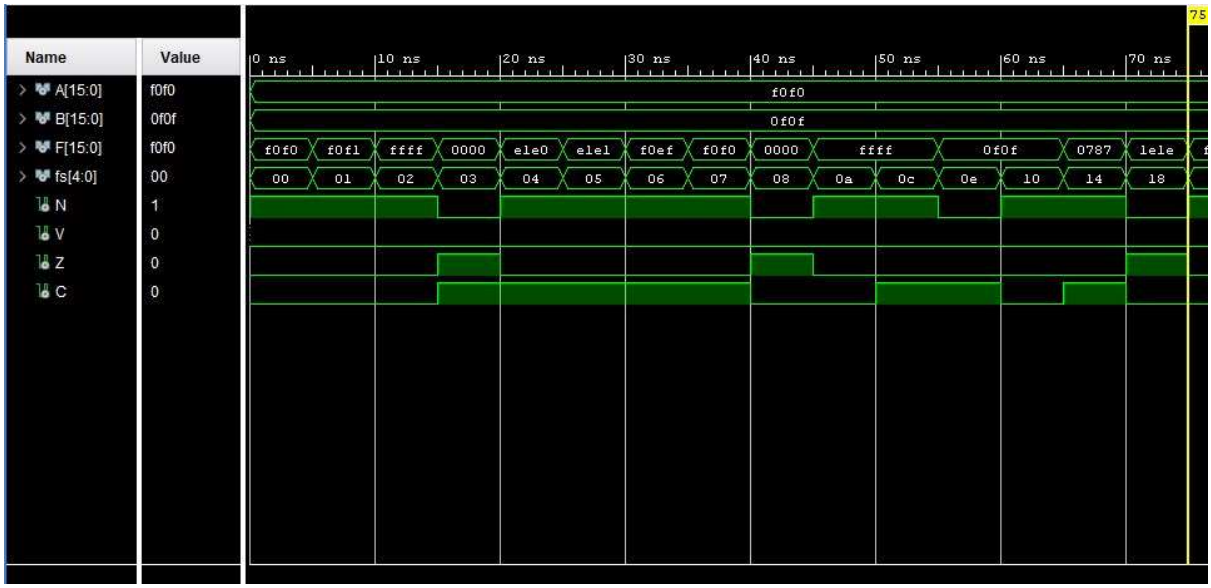


Shifter

I tested both shifting right and shifting left, and transferring by testing both h_sel options.

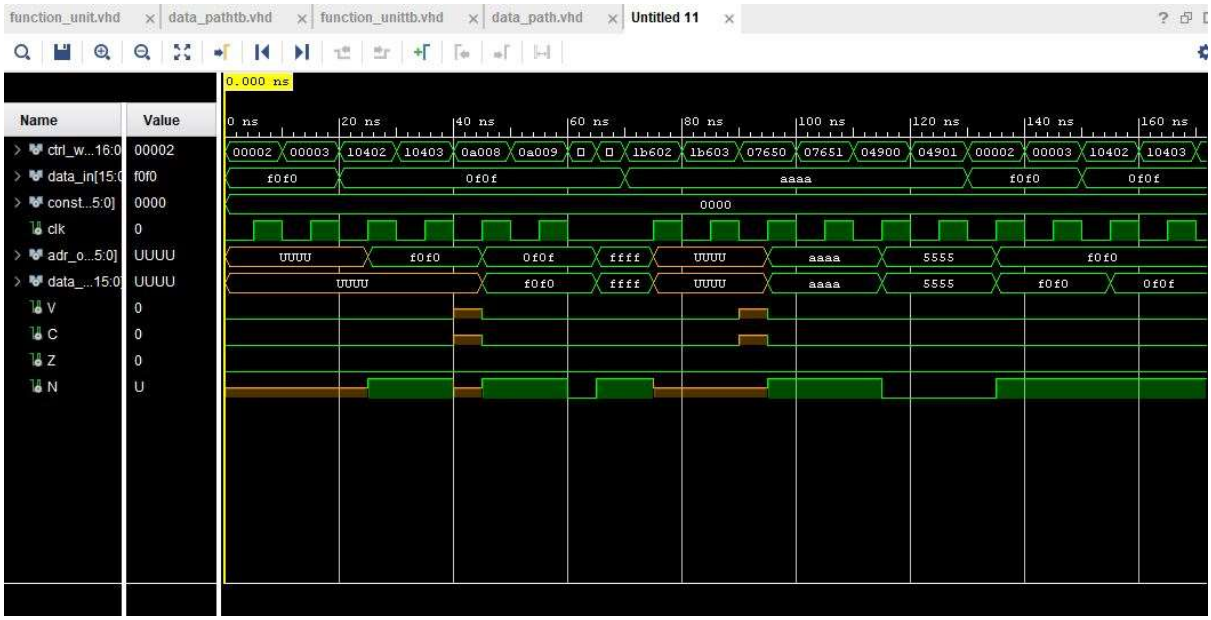


I tested the entire function unit by iterating through all the fs options to ensure the correct operation occurred and the correct output was generated.



Data Path

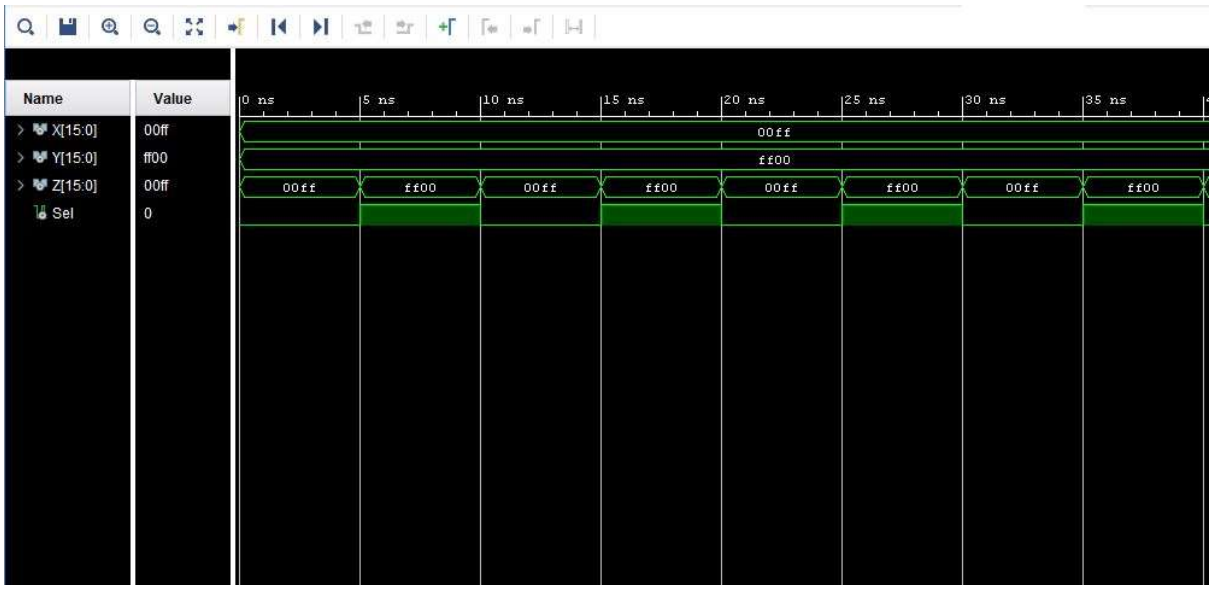
The data path contains the function unit and the register file. I tested the entire data path by testing that the result of the operation is stored where I declare it and the output comes from the register I declare in the control word. I didn't test all the operations again as they are thoroughly tested in previous test benches. Instead I focused on ensuring the data loads to and from where I declare in the control word when the load enable is 1.



Multiplexers

In this section of the project I added two more multiplexer types the multiplexers already created and tested in the previous part. The first one added was the mux2_1.

I iterated through combinations of X Y with select changing from 0 to 1.



Mux3_1 I also had to create a multiplexer with 3 inputs and 1 output.

Again I tested with 2 bit select alternating from 0 1 and 2.