# CprE 381, Computer Organization and Assembly Level Programming

# Lab 1 Report

Student Name Sullivan Hart

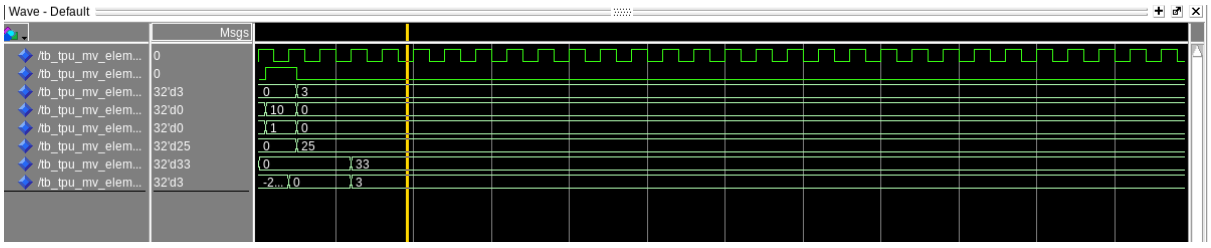
***Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the lab document for the context of the following questions***.

[Part 1.c] Think of three more cases and record them in your lab report.
*All values besides CLK are integer
If iX = 10, iY = 5, and iLdW = 0; then oX = 10, and oY = 105.
If iX = 5, iY = 0, iW = 5, and iLdW = 1; then oX = 5, and oY = 25.
If iX = 3, iY = 40, and iLdW = 0; then oX = 10, and oY = 70.

[Part 1.e] For labels 1, 7, 22, and 28, specify where (VHDL file and line number) these values are located – some will be found in more than one place. Also attempt to explain the functionality of each label as it occurs in the code
1: TPU_MV_Element; TPU_MV_Element.vhd (23: defines module called TPU_MV_Element) (33: ends the definition)(35: gives structure to the previously defined TPU module)
7: g_add1; TPU_MV_Element.vhd (117: creates an Adder called g_add1)
22: oQ; TPU_MV_Element.vhd (86: binds the output of the register to a wire, s_W); RegLd.vhd (31: sets oQ as an output of type integer)(51: sets oQ to be sQ, the stored value)
28: iD; TPU_MV_Element.vhd (114: takes input for the delay module from a wire, s_X1); Reg.vhd (29: sets iD as an input of type integer)(40: sends the value of iD to oQ on a rising edge)

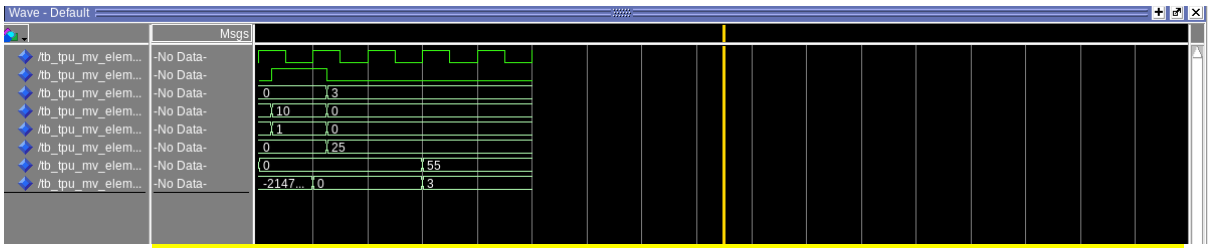
[Part 1.g.v] In your lab report, include a screenshot of the waveform. Describe, in plain English, any differences between what you expected and what the simulation showed.

The test bench outcome was different than expected. The input equation was 3 * 10 + 25, and that should equal 55. However, the outcome was 33.

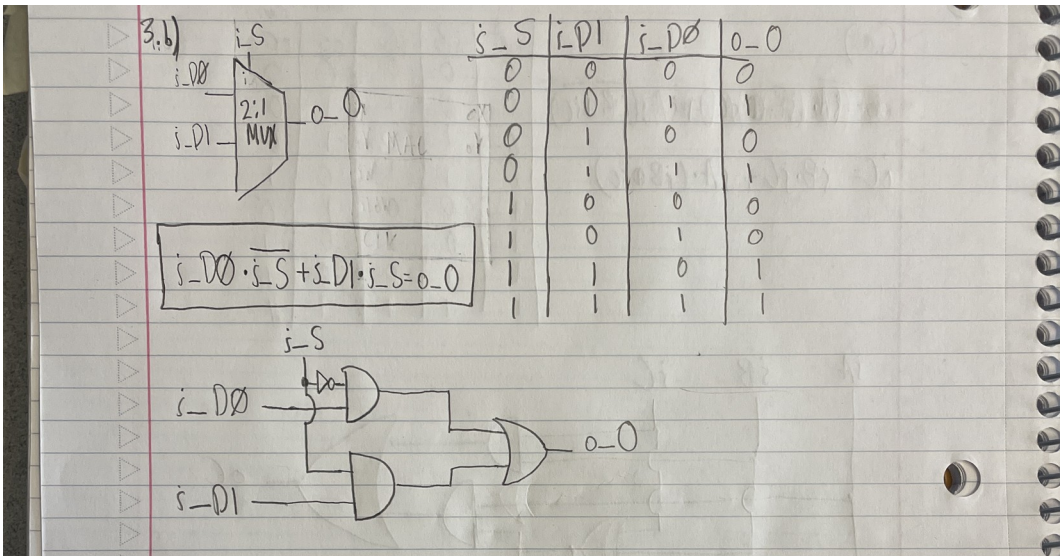


[Part 1.h] In your lab report, include a screenshot of the waveform. Describe, in plain English, how your waveform matches the expected result (e.g., reference the specific cycles and times). In your submission zip file, provide the completed *TPU_MV_Element.vhd* file in a folder called 'MAC'.

The waveform matched the expected outcome. The equation now results in 55, the expected result. The inputs and outputs occur in the appropriate "waves." First, the weight is loaded with a weight of 10 by setting the load control to 1. Next, the weight initialization stops and both values go to 0 because they aren't being used. At the same time, the inputs for X (to be multiplied) and Y ( to be summed) are gathered. Lastly, the outputs are given.




[Part 3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.




[Part 3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.

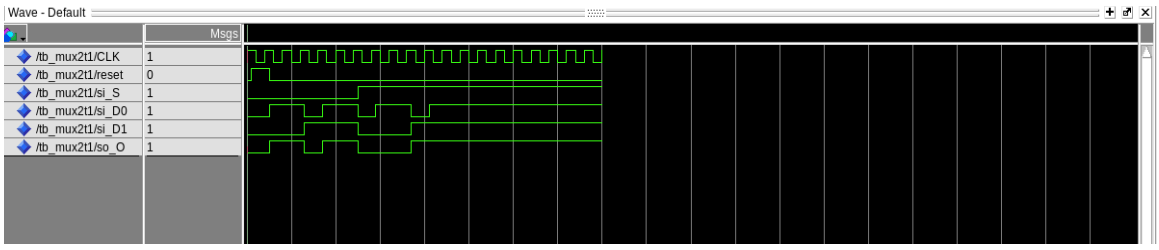



*Figure 1: Structural mux waveform*

[Part 3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.

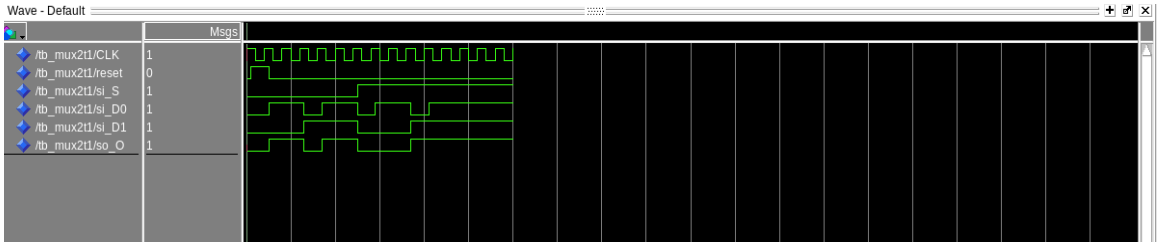



*Figure 2: Dataflow mux waveform*

[Part 4] Include a waveform screenshot and corresponding description demonstrating it is working correctly.

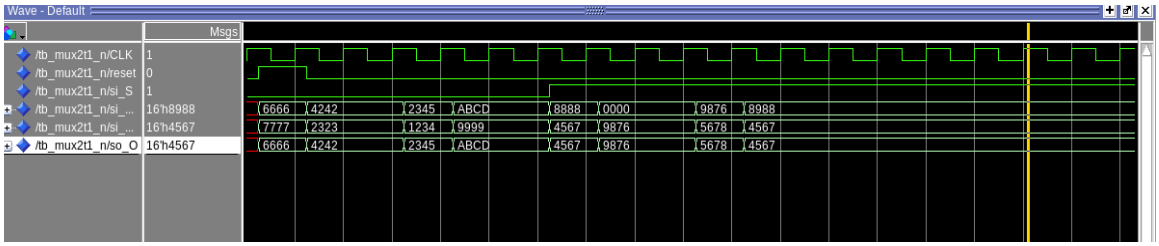



*Figure 3: When compared with the expected outcomes in the testbench, it is clear the component is working properly*

[Part 5.b] Include a waveform screenshot and description in your lab report.

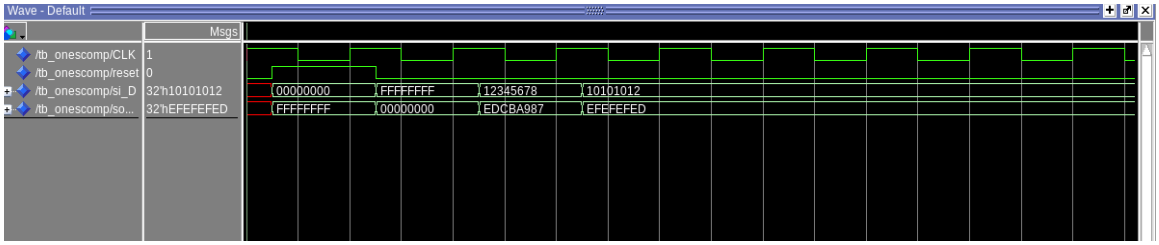
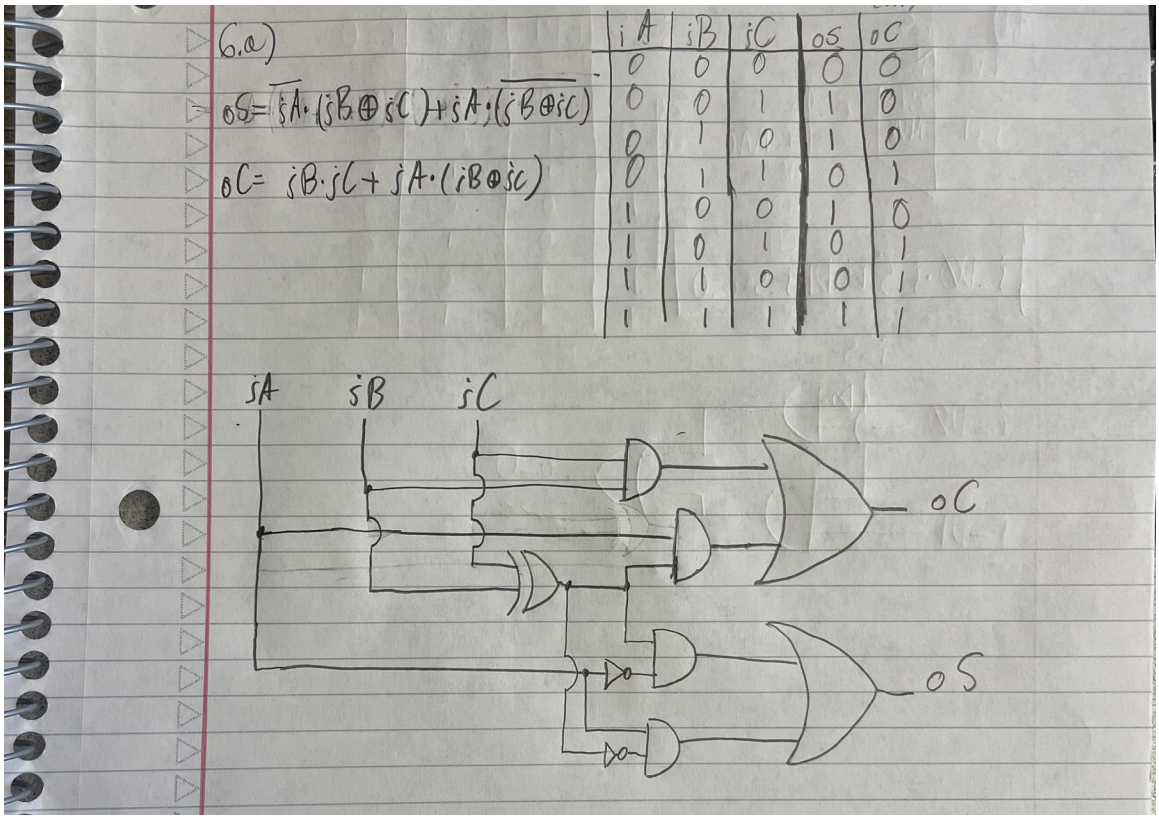
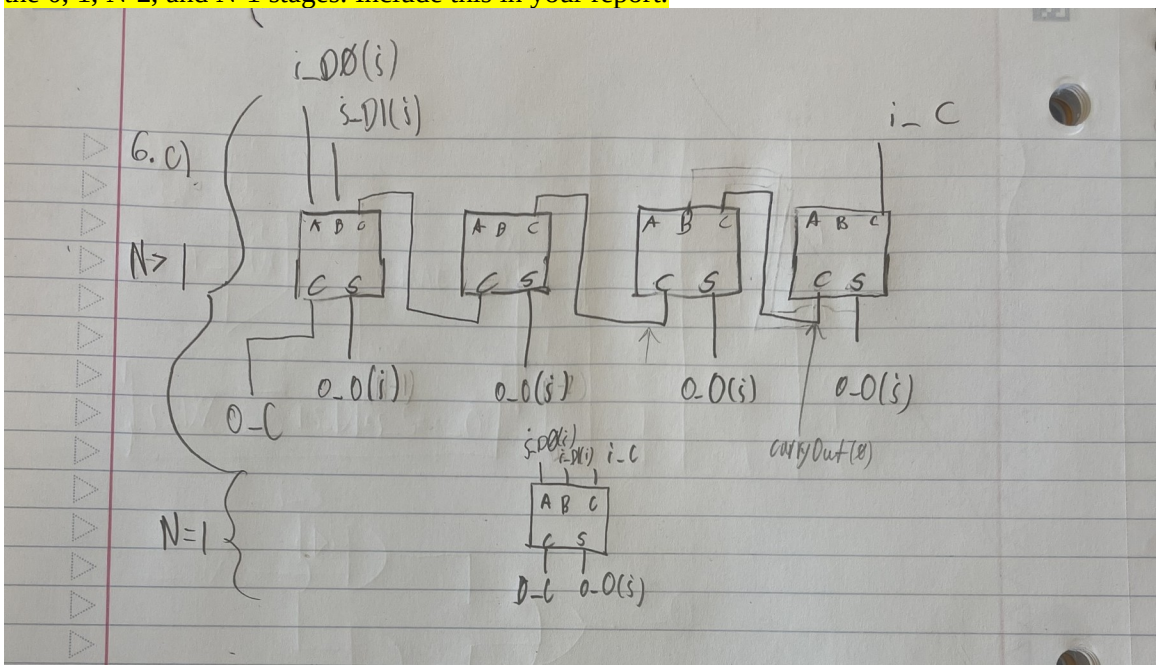



*Figure 4: The one's compliment working for N = 32*

[Part 6.a] A full adder takes three single-bit inputs and produces two single-bit outputs – a sum and carry for the addition of the three input bits. Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 1-bit full adder. Include this in your report.




[Part 6.c] Then draw a schematic of the intended design, including inputs and outputs and at least the 0, 1, N-2, and N-1 stages. Include this in your report.



[Part 6.d] Include an annotated waveform screenshot in your write-up.

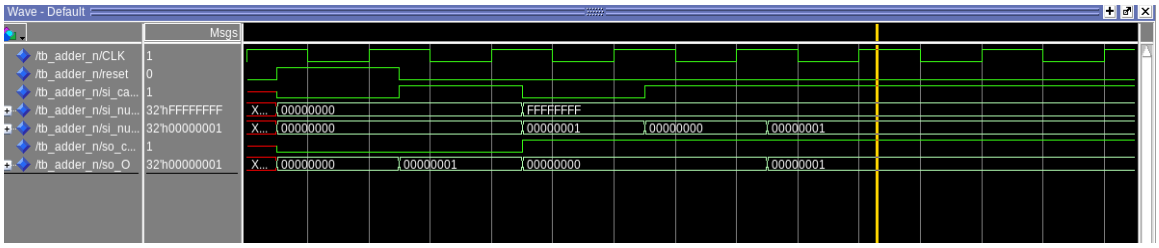
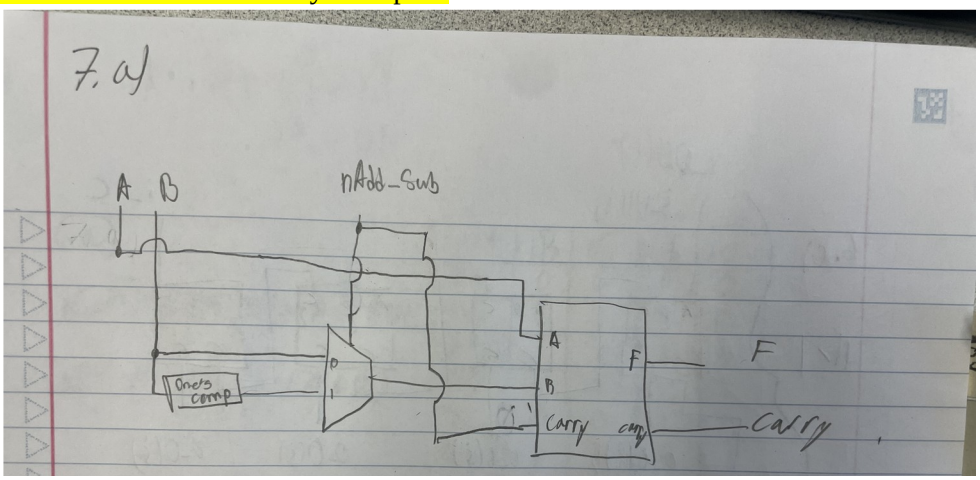



*Figure 5: When compared with the test cases in tb_adder_N.vhd, all inputs and outputs match*

[Part 7.a] Draw a schematic (don't use a schematic capture tool) showing how an N-bit adder/subtractor with control can be implemented using only the three main components designed in earlier parts of this lab (i.e., the N-bit inverter, N-bit 2:1 mux, and N-bit adder). How is the 'nAdd_Sub' bit used? Include this in your report.




[Part 7.c] Provide multiple waveform screenshots in your write-up to confirm that this component is working correctly. What test-cases did you include and why?

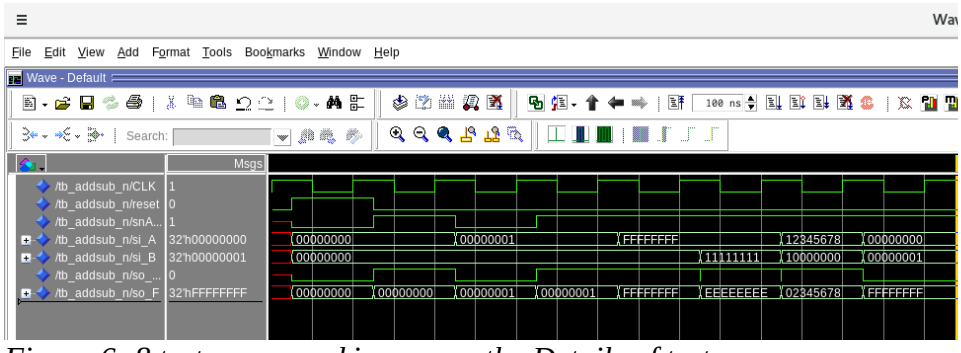



*Figure 6: 8 test cases working correctly. Details of test cases are available in the testbench. Chose tests with basic functionslity as well as cases with edge cases.*