CprE 3810, Computer Organization and Assembly Level Programming

Lab 1 Report

Student Name **Cai Chen**

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 3.1.c] Think of three more cases and record them in your lab report.
What the TPU does: X * W + Y = oY

X = 0, W = 3, Y = 1
0 * 3 + 1 = 1

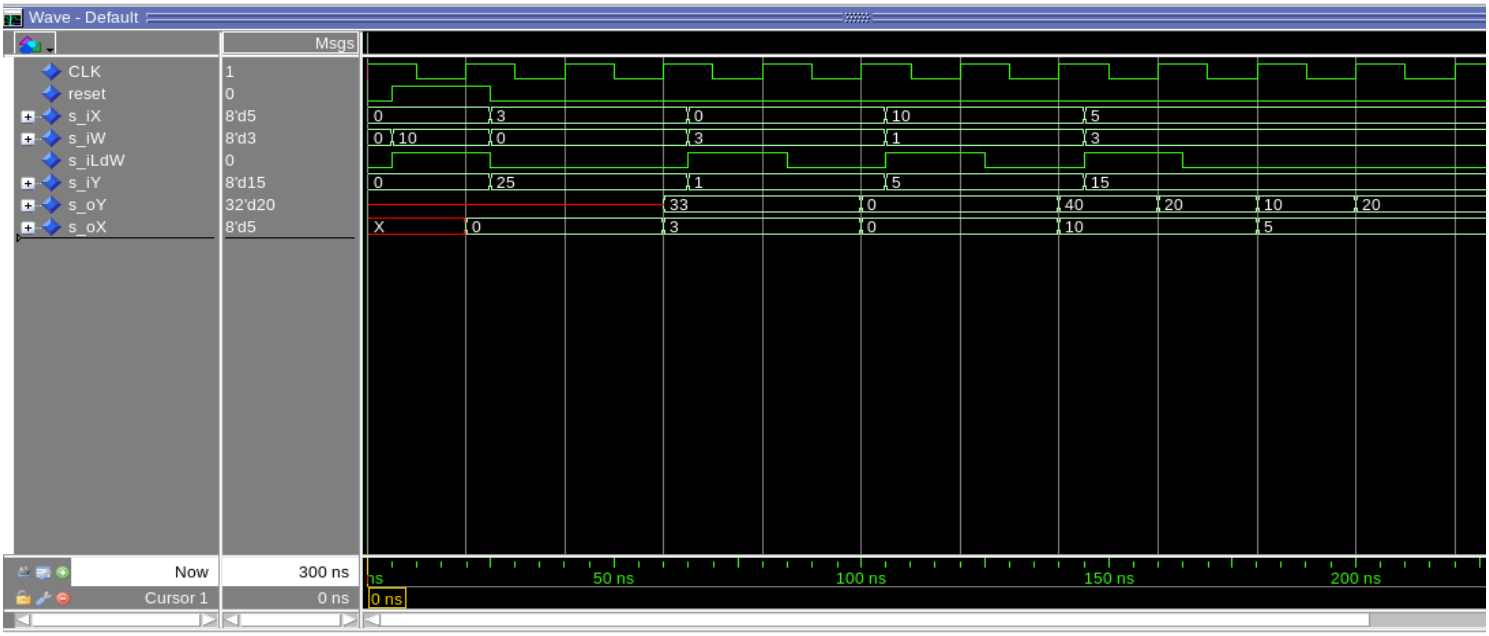
X = 10, W = 1, Y = 5
10 * 1 + 5 = 15

X = 5, W = 3, Y = 15
5 * 3 + 15 = 30

[Part 3.1.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
Label 1: /TPU/Lab1/TPU_MV_Element.vhd – does the calculations based on the above equation
Label 7: TPU_MV_Element.vhd line 131 – calculates y = w*x
Label 22: TPU_MV_Element.vhd lines 77, 96, 118 – stores the weight value
Label 28: TPU_MV_Element.vhd lines 63 & 70 – stores values between delays?

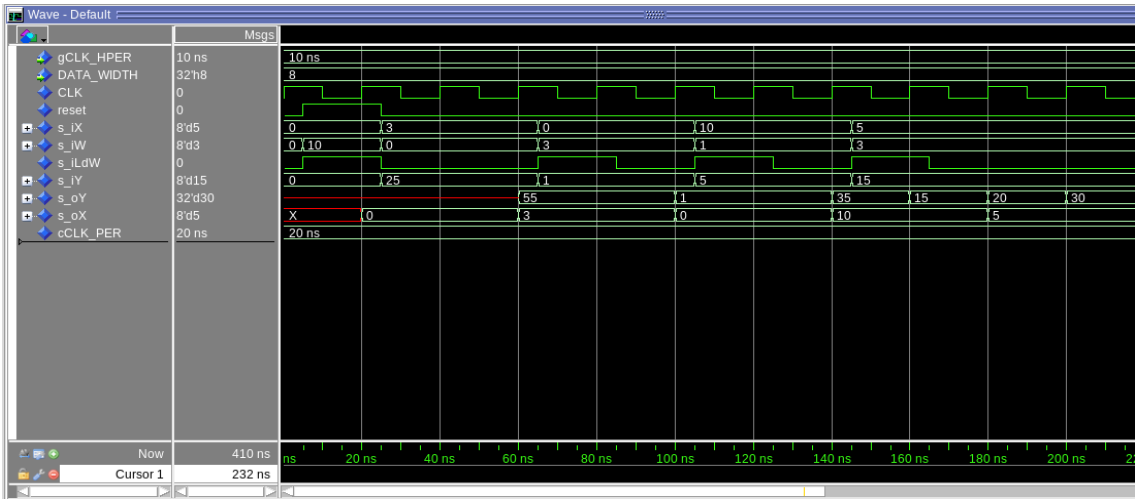
[Part 3.1.f] Implement your test cases from part c into this testbench. Remember to include these in your waveform screenshot.

[Part 3.1.g] 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.
Waveform Picture:
There is a bug here since the MAC is supposed to do (3 * 10) + 25, which equals 55 but on the waveform oY equals 21. The bug is most likely the fact that s_iY is 25 and not weight. The same is true for the rest of the test cases, there is a bug that doesn't properly calculate the correct value.



[Part 3.1.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 bug is actually caused by a incorrect delay in Level 1 of the TPU_MV_element. Instead of delaying iX and iY, it was delaying iX twice. Now it functions as expected. (3 * 10) + 25 = 55. The test cases also display correctly.

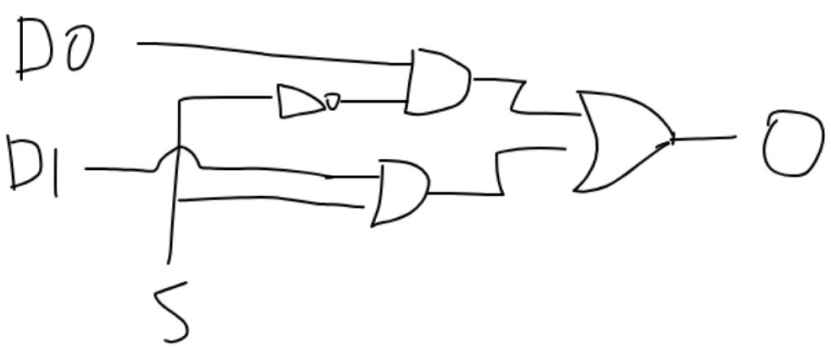


[Part 3.3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.

o_O = Not (i_S) (i_D0) + (i_S)(i_D1)

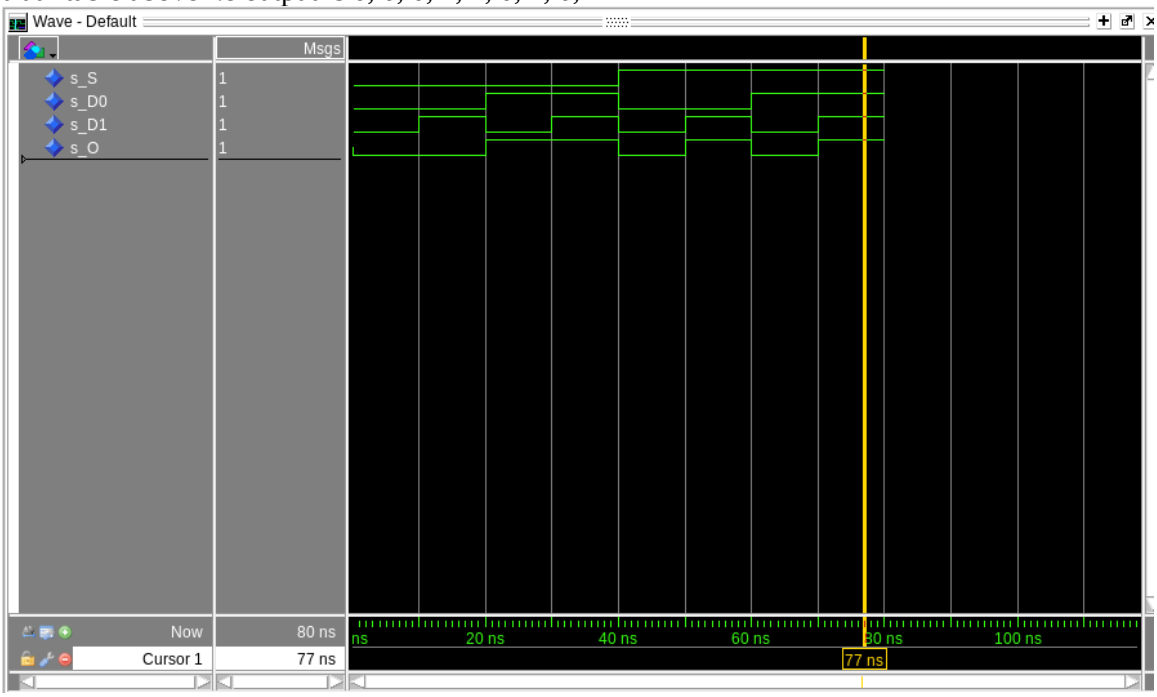
i_S	i_D0	i_D1	o_O
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

[Part



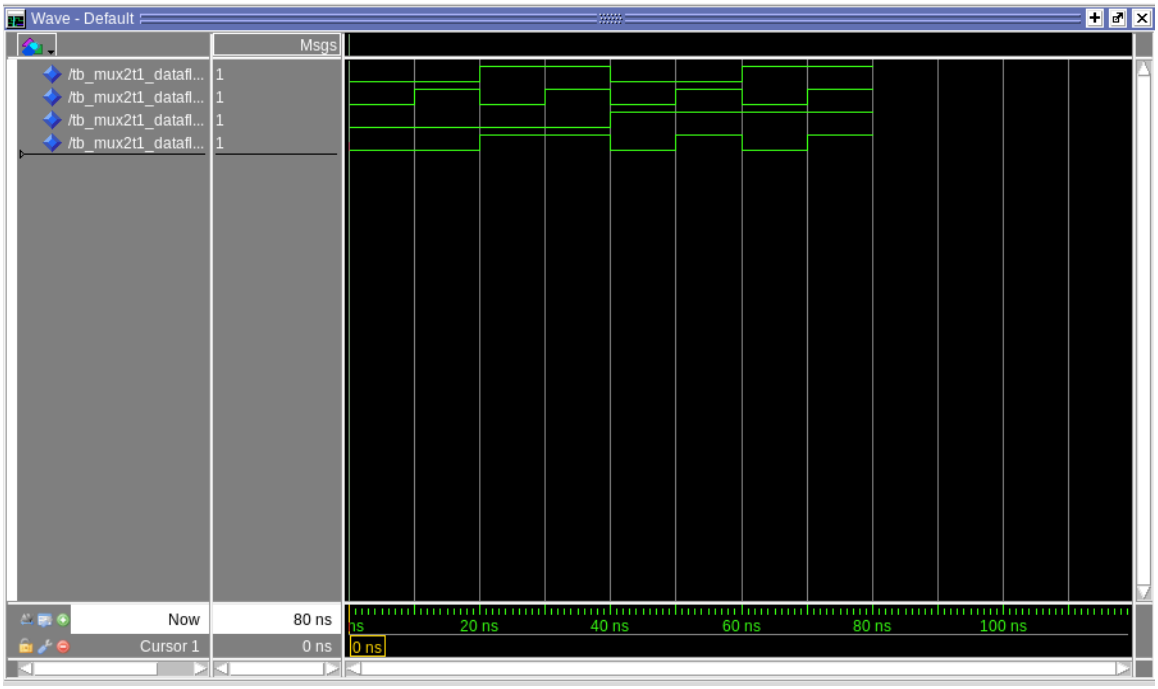
3.3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.

This picture shows the mux working correctly: it has the same expected values as the truth table above I.e output is 0, 0, 0, 1, 1, 0, 1, 0, 1



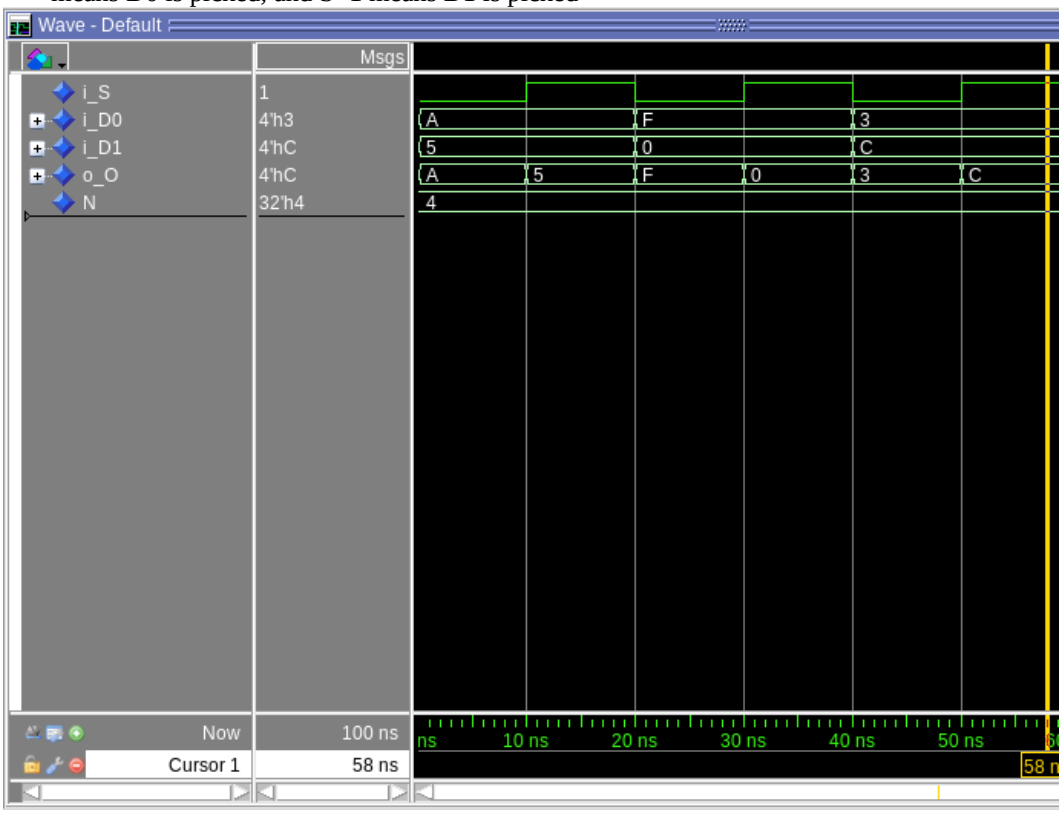
[Part 3.3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.

This picture shows the correctly working dataflow version of the MUX. The out is again 0, 0, 0, 1, 1, 0, 1, 0, 1



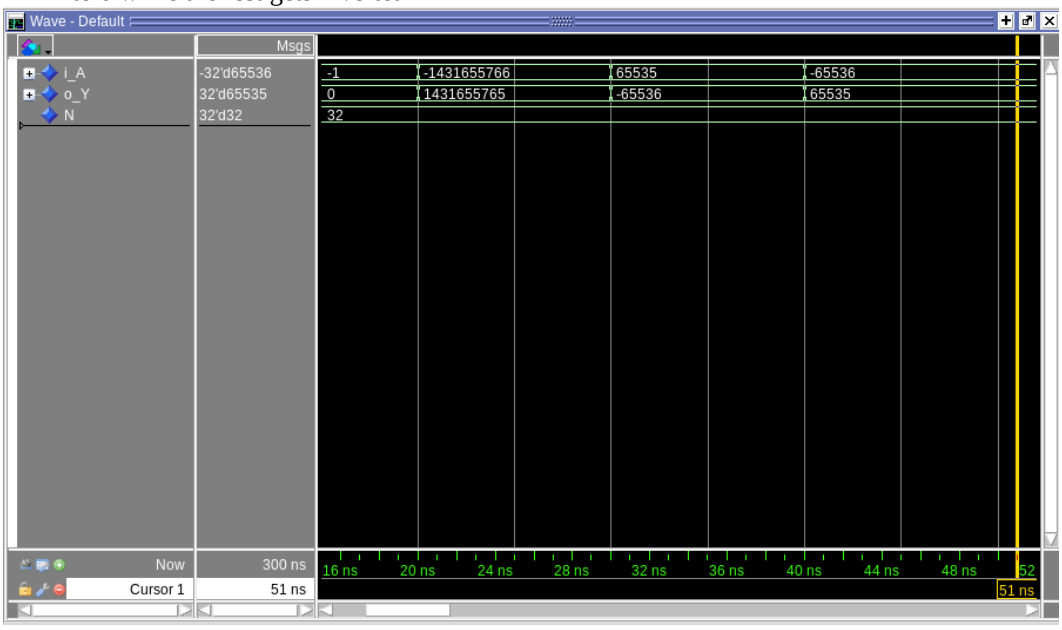
[Part 3.4] Include a waveform screenshot and corresponding description demonstrating it is working correctly.

The Mux is working correctly, it chooses between the 4 bit value as shown in the waveform. S=0 means D0 is picked, and S=1 means D1 is picked



[Part 3.5.b] Include a waveform screenshot and description in your lab report.

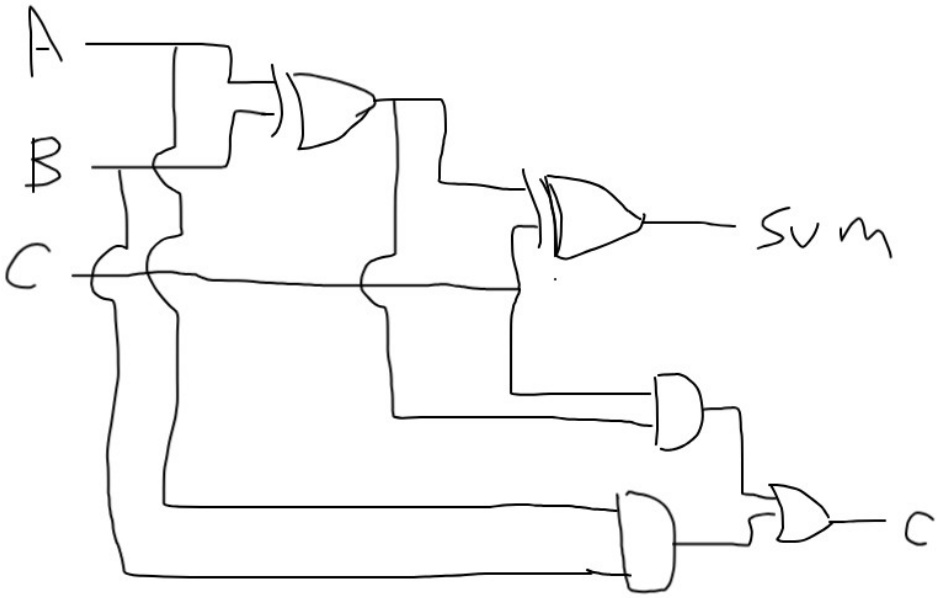
The ones complement works as expected, it properly inverts the input with -1 being transformed into 0 while the rest gets invertet



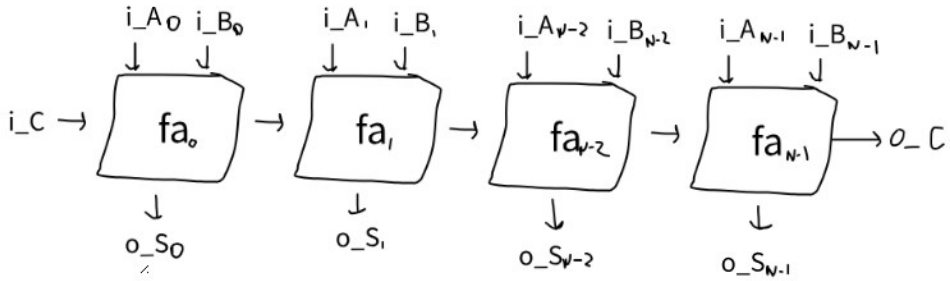
[Part 3.6.a] A full adder takes three single-bit inputs and produces two single-bit outputs – a sum and carry for the addition of the three input bits. Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 1-bit full adder. Include this in your report.

o_SUM = i_A XOR i_B XOR i_C
o_C = i_A * i_B + i_C (i_A XOR i_B)

i_A	i_B	i_C	o_SUM	o_C
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

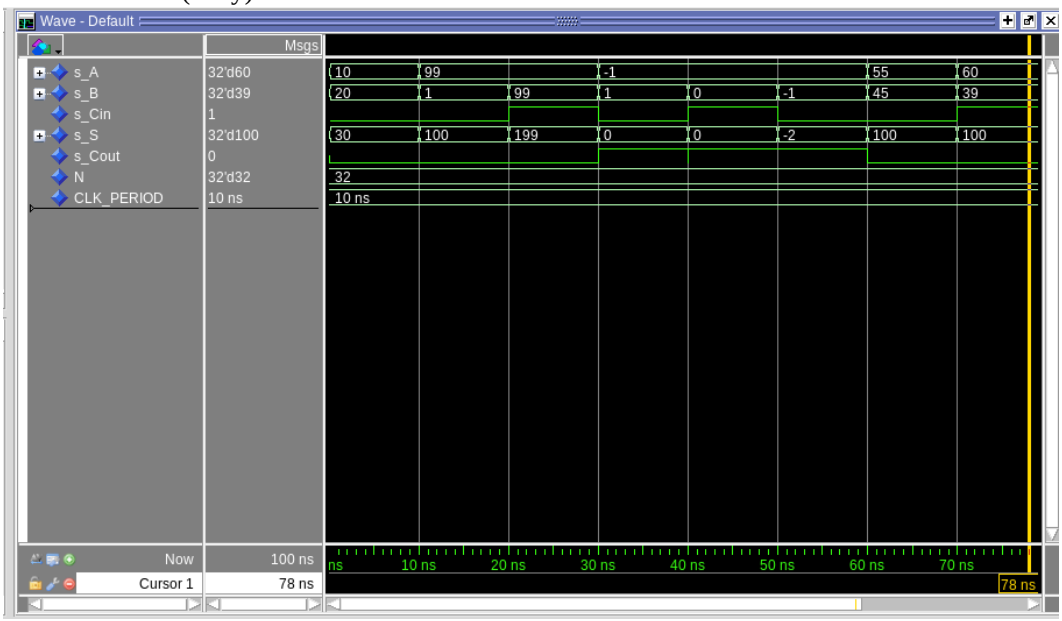


[Part 3.6.c] Then draw a schematic of the intended design, including inputs and outputs and at least the 0, 1, N-2, and N-1 stages. Include this in your report.



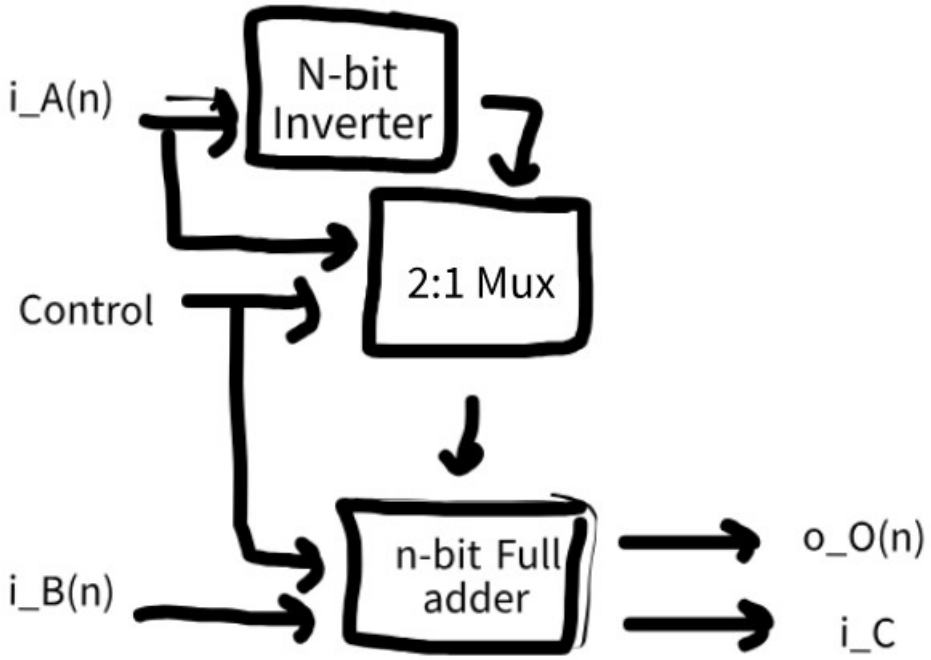
[Part 3.6.d] Include an annotated waveform screenshot in your write-up.

Each test properly matches the expected outcome. For example:
10 + 20 = 30, no carry
99 + 99 + 1 (carry) = 199 (no carry)
-1 + -1 = -2 (carry)



[Part 3.7.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.

The nAdd_Sub is used as the control (nAdd_Sub = control), choosing if the circuit is an adder (nAdd_Sub = 0) or subtractor (nAdd_Sub = 1)



[Part 3.7.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?

I included 6 tests, each testing an individual part of the functionality. They include: simple addition/subtraction, small number – big number, negative – negative, negative + negative, negative small - negative big.