CprE 381, Computer Organization and Assembly Level Programming

Lab 1 Report

Student Name ______________Ethan Roepke_________________

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.

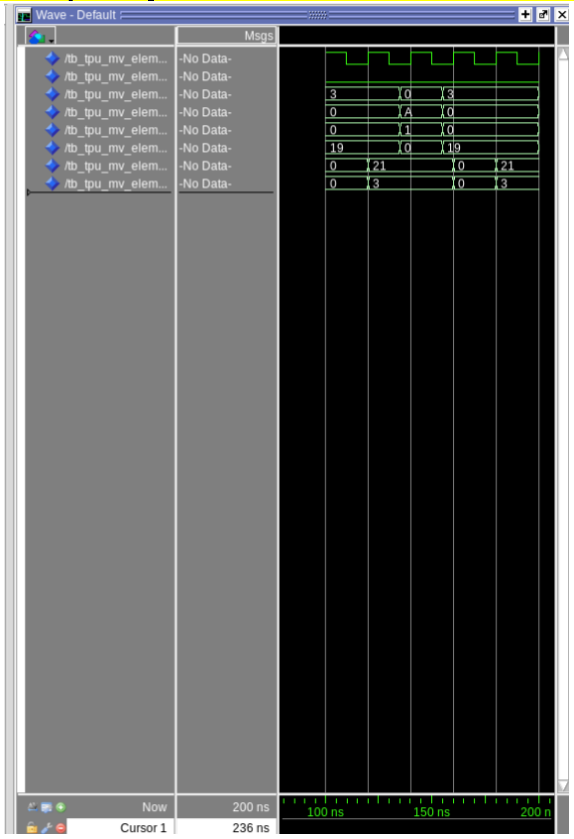
[Part 1.c] Think of three more cases and record them in your lab report.

Case 1: Initialize a weight value register by setting "iW" to the desired value of 15
and "iLdW" to
1 prior to a positive edge of the clock and holding them until after the positive edge of the clock. Then, "s_W" should take on my new value of 15 at the positive edge of the clock.
Case 2: Modify the weight value register by setting "iW" to the desired value of 20 and "iLdW" to 1 across a positive edge of the clock and holding them until after the positive edge of the clock. Then, "s_W" should change to 20.
Case 3: With the above weight modified to 10, I set the activation input "iX" to 2 and "iY" to 10. I make sure the weight is not changed by setting "iLdW" to 0. After the cycles, the activation output "oX" will be 2 and "oY" will be 50.

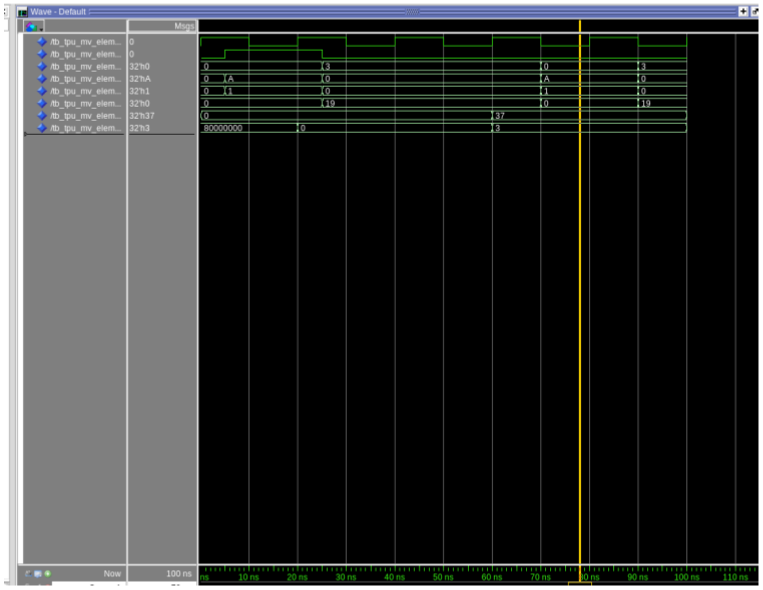
[Part 1.e] For labels 9, 20, 32, and 33, specify where (VHDL file and line number) these values are located – some will be found in more than one place. Also attempt to explain the functionality of each label as it occurs in the code

For label 9, the "oQ" output of the load/weight module is defined as an integer on line 31 in the RegLD.vhd file and set on Line 51. For this specific instance of the load module, the output "oQ" is connected to a signal called "s_W" on line 86 of the TPU_MV_Element.vhd file.
For label 20, the "iD" input of the delay3 module is defined as an integer on Line 29 of Reg.vhd file. The input "iD" is connected from a signal called "s_X1" on line 114 of TPU_MV_Element.vhd file.
For label 32, the adder module is designed in Adder.vhd. The output "oC" is connected to "oY" on line 121 of TPU_MV_Element.vhd file. Input "iA" to a signal called "s_WxX" on line 119 and "iB" to a signal called "s_Y1" on line 120 of TPU_MV_Element.vhd file.
For label 33, the entire system is designed in TPU_MV_Element.vhd file.

[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.


Wave - Default
Msgs
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
/tb_tpu_mv_elem... -No Data-
3 0 3
0 A 0
0 1 0
19 0 19
0 21 0 21
0 3 0 3
Now 200 ns
Cursor 1 236 ns
100 ns
150 ns
200 n

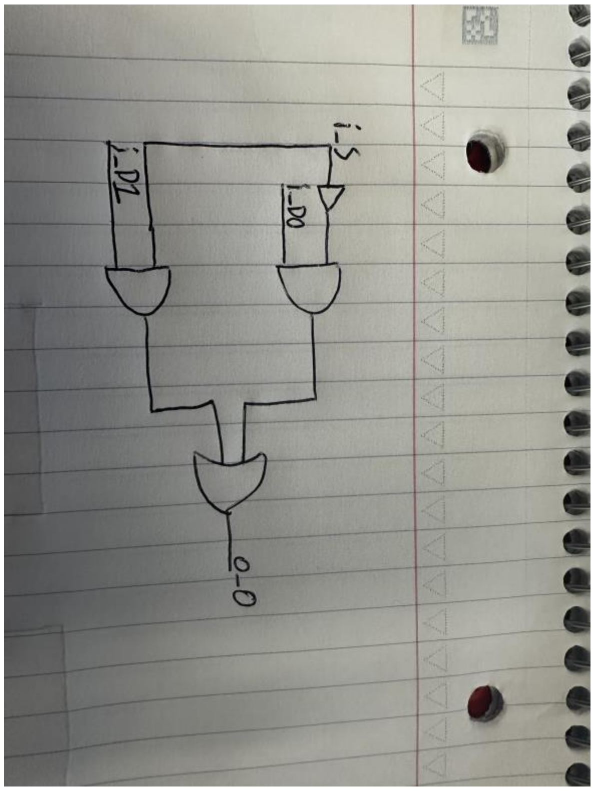
[Part 1.h] In your lab report, include a screenshot of the waveform. Describe, in plain English, how your waveform matches the expected result (e.g., reference the specific cycles and times). In your submission zip file, provide the completed *TPU_MV_Element.vhd* file in a folder called ‘MAC’.



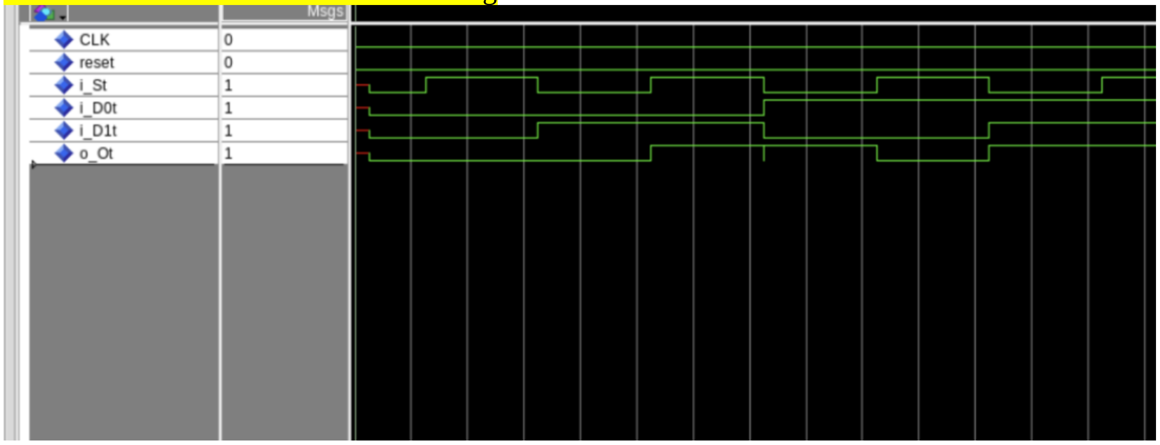
[Part 3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.

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

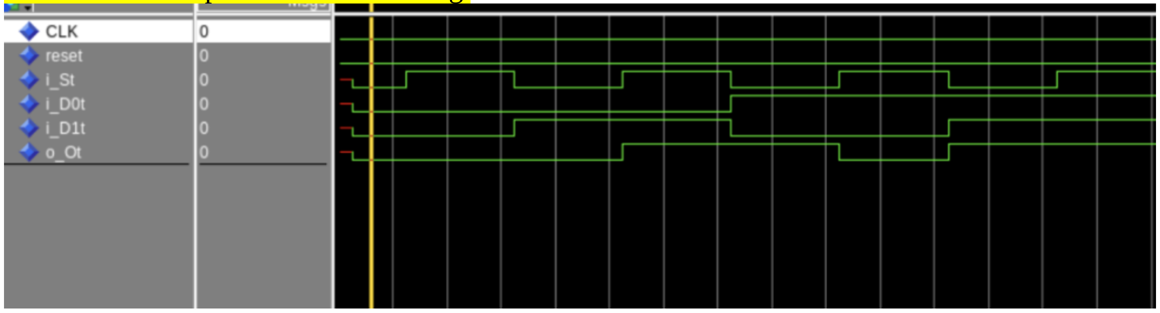
o_O = (i_D0 * i_S’) + (i_D1 * i_S)



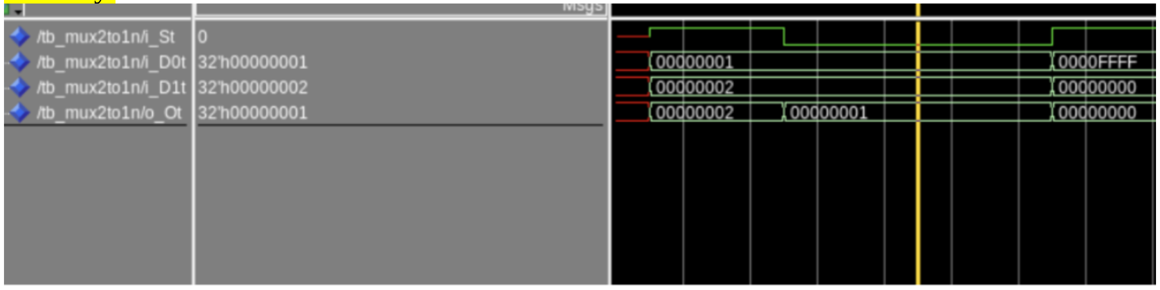
[Part 3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.



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

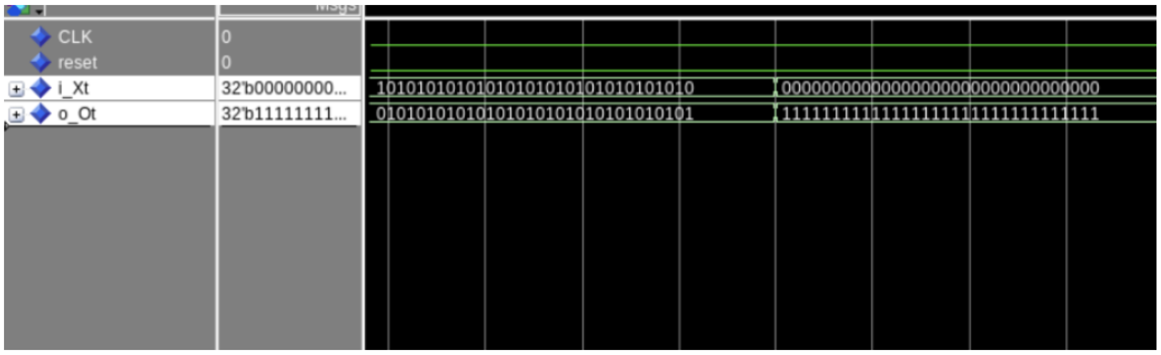


[Part 4] Include a waveform screenshot and corresponding description demonstrating it is working correctly.


Msgs
/tb_mux2to1n/i_St 0
/tb_mux2to1n/i_D0t 32'h00000001
/tb_mux2to1n/i_D1t 32'h00000002
/tb_mux2to1n/o_Ot 32'h00000001
00000001
0000FFFF
00000002
00000000
00000002
00000001
00000000

The waveform above is displaying the Mux2to1_N operation. The waveform is working correctly. In the initial scenario, the signal is set to 0, it correctly selects D1 as the output. When change signal bit to 1, the output switches to D0.

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

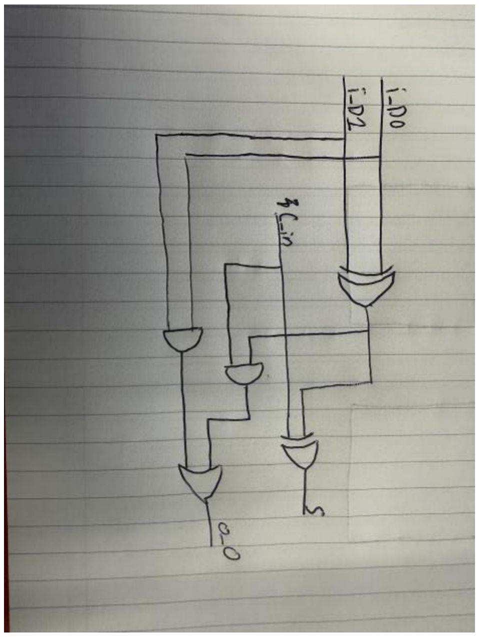


```
CLK     0
reset   0
i_Xt    32'b00000000...   10101010101010101010101010101010   00000000000000000000000000000000
o_Ot    32'b11111111...   01010101010101010101010101010101   11111111111111111111111111111111
```

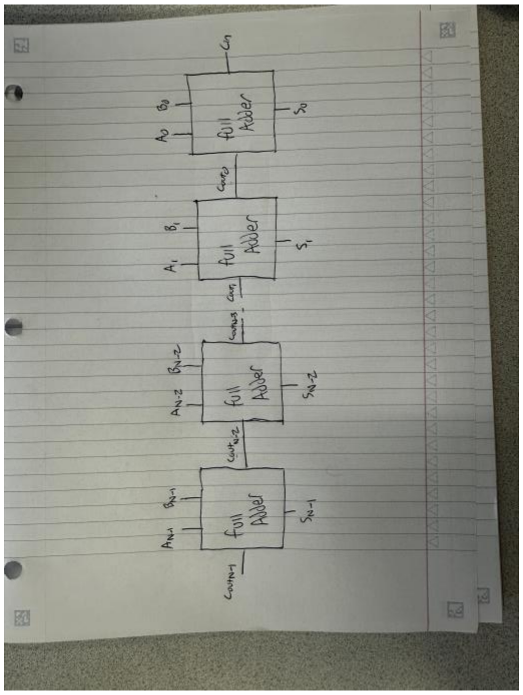
[Part 6.a] A full adder takes three single-bit inputs and produces two single-bit outputs – a sum and carry for the addition of the three input bits. Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 1-bit full adder. Include this in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

i_D0	i_D1	C_in	Sum(S)	o_O
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

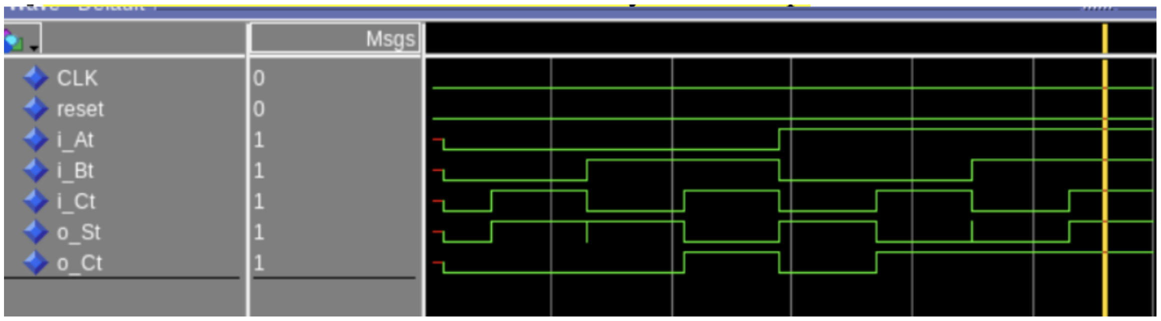
S = i_D0 ⨁ i_D1 ⨁ C_in
o_O = (i_D0 * i_D1) + (i_D1 * C_in) + (i_D0 * C_in)



[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.

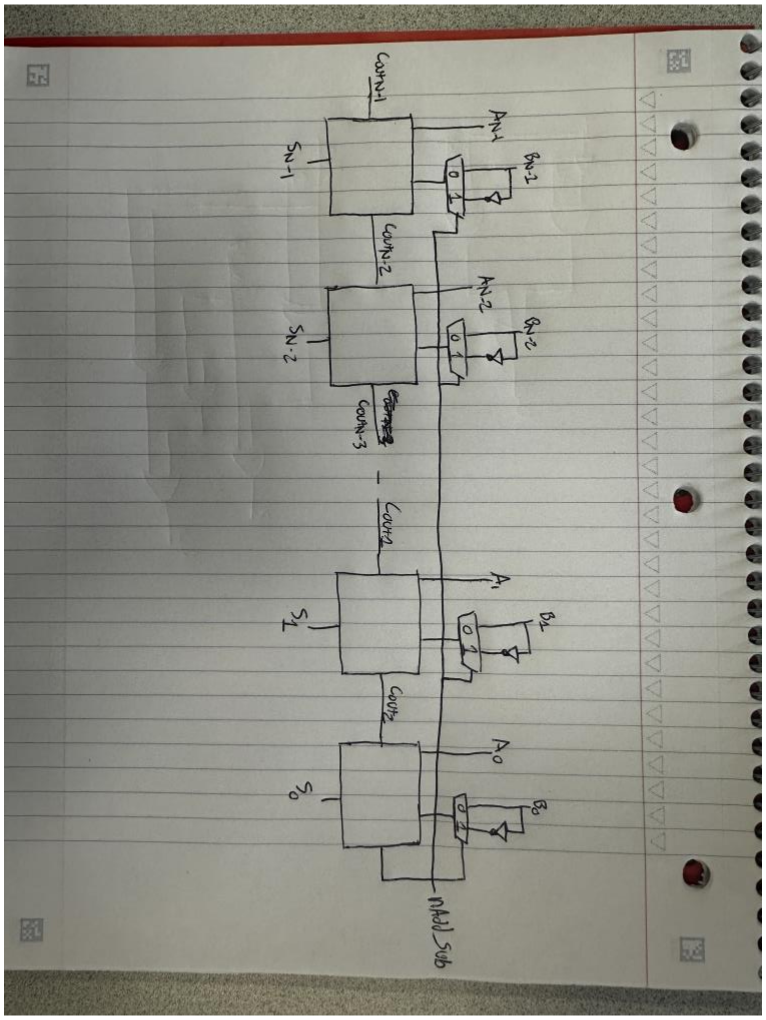


[Part 6.d] Include an annotated waveform screenshot in your write-up.



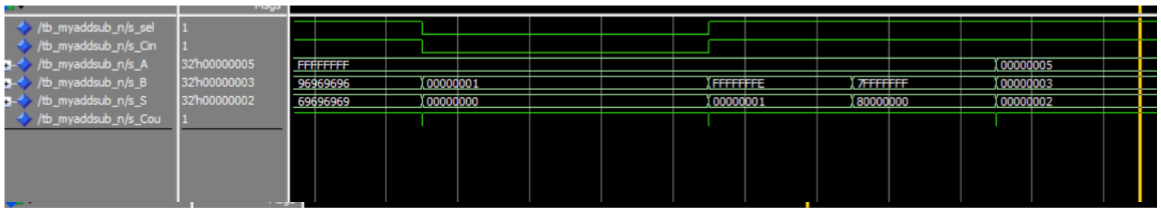
	Msgs
CLK	0
reset	0
i_At	1
i_Bt	1
i_Ct	1
o_St	1
o_Ct	1

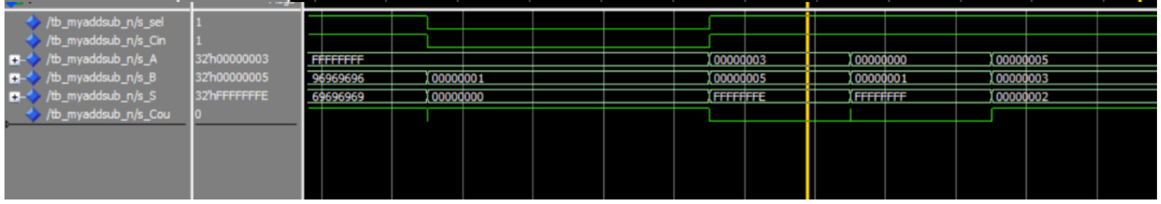
[Part 7.a] Draw a schematic (don’t use a schematic capture tool) showing how an N-bit adder/subtractor with control can be implemented using only the three main components designed in earlier parts of this lab (i.e., the N-bit inverter, N-bit 2:1 mux, and N-bit adder). How is the ‘nAdd_Sub’ bit used?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



nAdd_Sub is going to be used as a selector or the muxes to be used in the design and also as the initial carry in. When nAdd_Sub is 1, subtraction will be performed. All muxes will be set such that the inverted signal goes through for a selector value of 1. When nAdd_Sub is 0, addition will be performed. The value of B that is sent to the adder will be the initial value

[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?


/tb_myaddsub_n/s_sel 1
/tb_myaddsub_n/s_Cin 1
/tb_myaddsub_n/s_A 32'h00000005
/tb_myaddsub_n/s_B 32'h00000003
/tb_myaddsub_n/s_S 32'h00000002
/tb_myaddsub_n/s_Cou 1
FFFFFFFF
00000005
96969696
00000001
FFFFFFFE
7FFFFFFF
00000003
69696969
00000000
00000001
80000000
00000002


/tb_myaddsub_n/s_sel 1
/tb_myaddsub_n/s_Cin 1
/tb_myaddsub_n/s_A 32'h00000003
/tb_myaddsub_n/s_B 32'h00000005
/tb_myaddsub_n/s_S 32'hFFFFFFFE
/tb_myaddsub_n/s_Cou 0
FFFFFFFF
00000003
00000000
00000005
96969696
00000001
00000005
00000001
00000003
69696969
00000000
FFFFFFFE
FFFFFFFF
00000002