CprE 381, Computer Organization and Assembly Level Programming

Lab 1 Report

Student Name Will Galles

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.

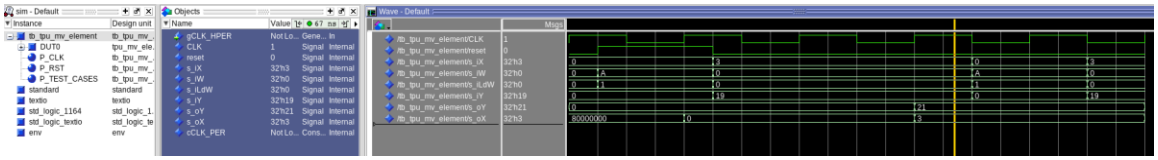
[Part 1.c] Think of three more cases and record them in your lab report.

- Test Case 1 (Birthdate): Weight = 4, I_X = 5, I_Y = 5, O_Y should be 25, O_X should be 5.
- Test Case 2: Weight = 0, I_X = 5, I_Y = 10, O_Y should be 10, O_X should be 5
- Test Case 1: Weight = 3, I_X = 7, I_Y = 0, O_Y should be 21, O_X should be 7

[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

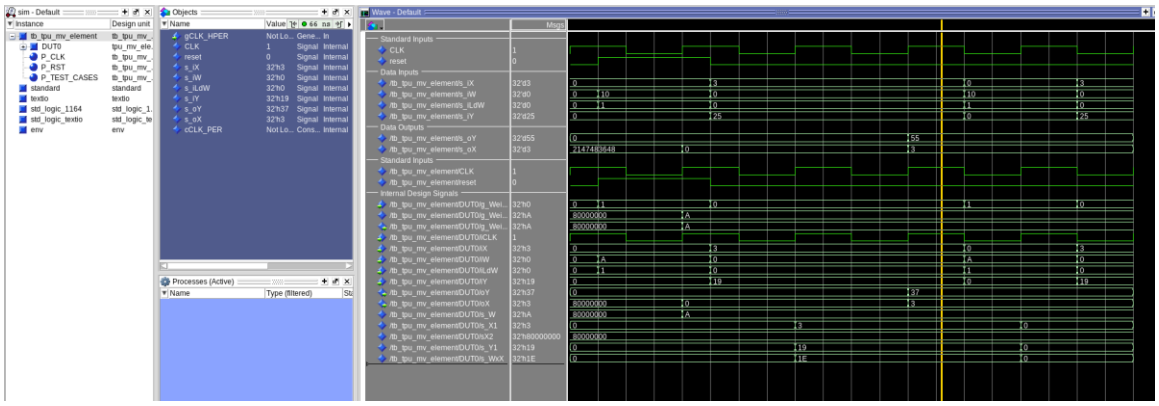
- Label 1 corresponds with the entire TPU_MV_Element.vhd file, this points to the entire unit that contains and connects all of the smaller subcomponents.
- Label 7 corresponds with the instantiated adder g_Add1 on line 117 of the TPU_MV_Element.vhd file, this is the adder responsible for adding the carry input Y with the product of the weight and the X input.
- Label 22 corresponds with the Q output of the g_Weight load register module that is instantiated on line 92 of the TPU_MV_Element.vhd file, this register is responsible for loading the weight when the load signal is asserted. The Q output of this module is the weight that is stored in the register.
- Label 28 corresponds with the s_X1 that is defined on line 70 of the TPU_MV_Element.vhd file, this signal is used to map the output of the first delay of the X input to the input of the second delay of the X input. Routes between the output of the g_Delay1 module to the input of the g_Delay3 module.

[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.



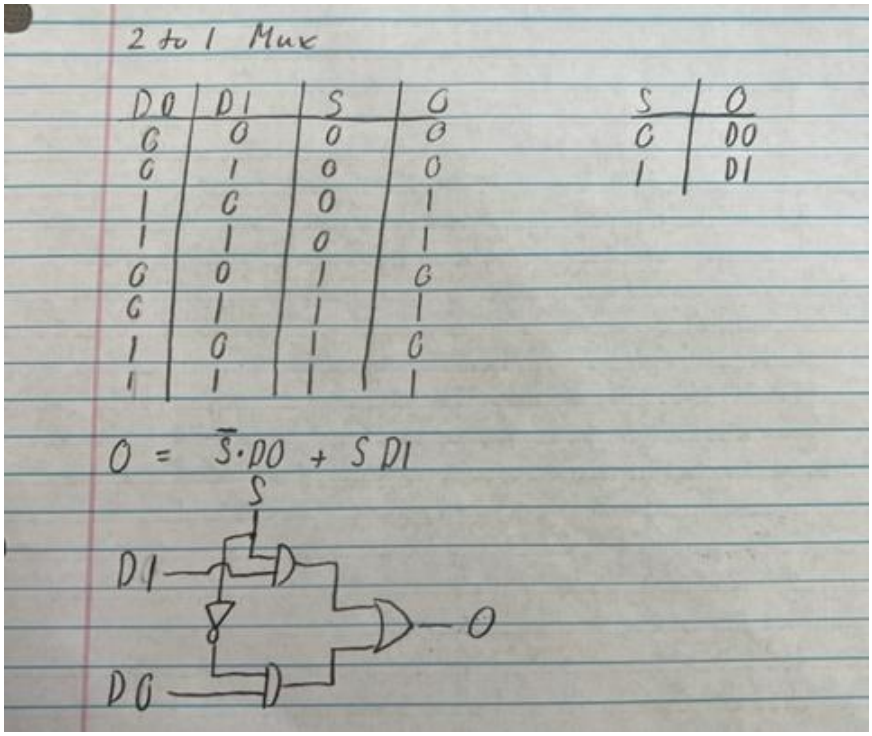
The output o_Y has a final value of 0x21 or 0d33 but it should have a final output value of 0x37 or 0d55. The final ouput o_X is correct in the case however.

[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'.

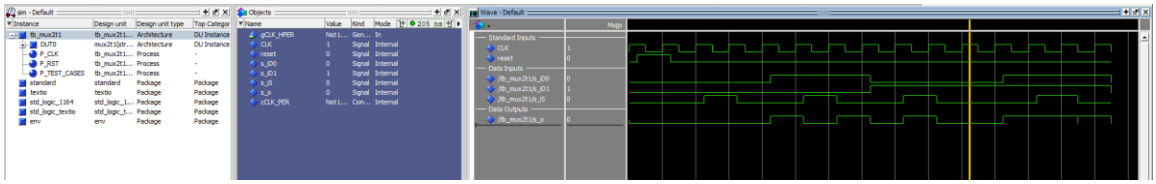


To start with in between the first rising and falling edges of the clock we set our weight input to 10 and load to 1. When the next rising edge occurs the internal weight (output of weight register) becomes 10. In the next quarter cycle, we clear both our weight input and load while setting out X input to 10 and out Y input to 25. On the next rising edge, we see the X value propagate through its first delay register along with the Y value propagate through its delay register. During that same time, we see the X value also multiplied by the internal weight for a product of 30. On the next rising clock edge, we see the delayed X value propagated through the second delay register along with the product of X and the Weight added to the delayed Y value for a final value of 55. From there we see the original X input of 3 as the X output and the new sum of 55 as the Y output.

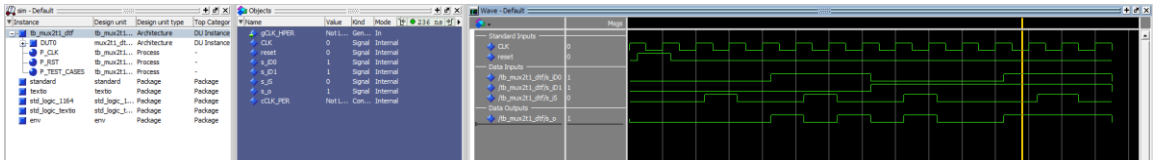
[Part 3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.



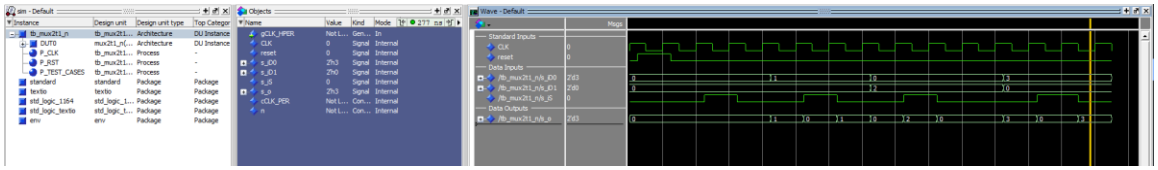
[Part 3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.



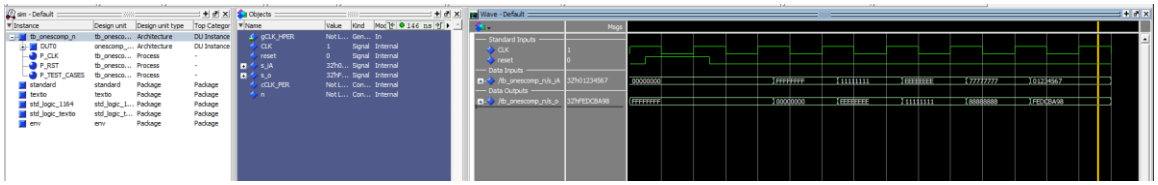
[Part 3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.



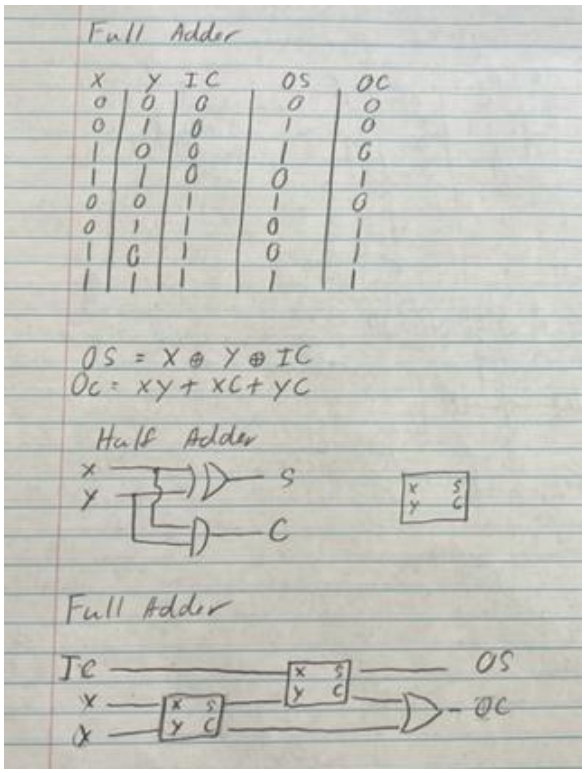
[Part 4] Include a waveform screenshot and corresponding description demonstrating it is working correctly.



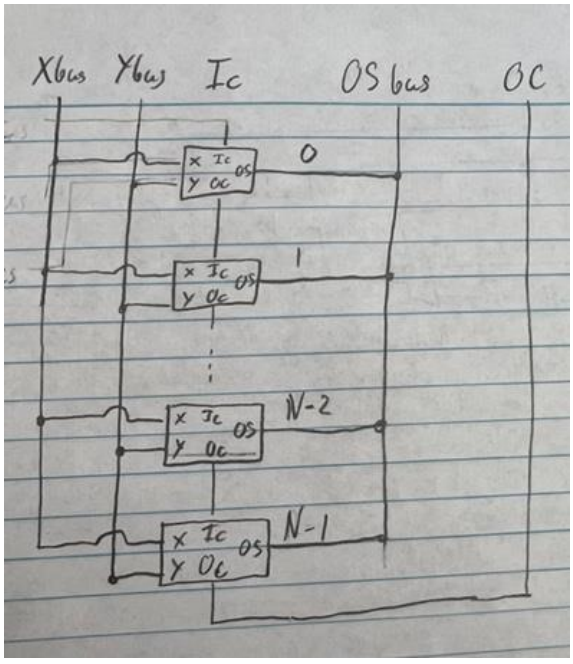
[Part 5.b] Include a waveform screenshot and description in your lab report.



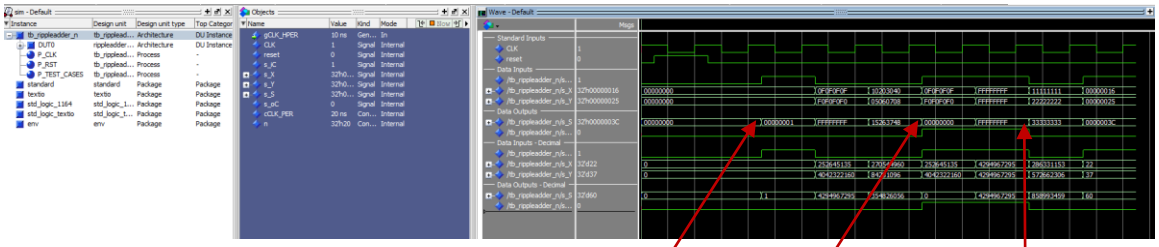
[Part 6.a] A full adder takes three single-bit inputs and produces two single-bit outputs – a sum and carry for the addition of the three input bits. Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 1-bit full adder. Include this in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



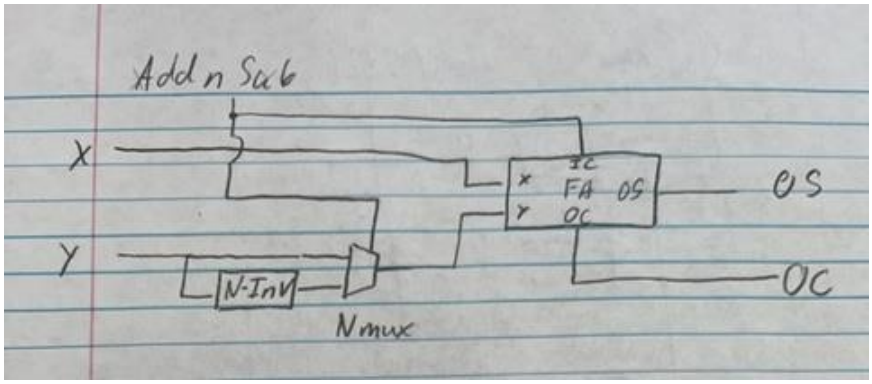
First one is adding 0 + 0 with a carry of 1 to get a total output of 1

Another is adding 0F0F0F0F and F0F0F0F0 with a carry bit so when the add together the sum clears and all is left is the carry

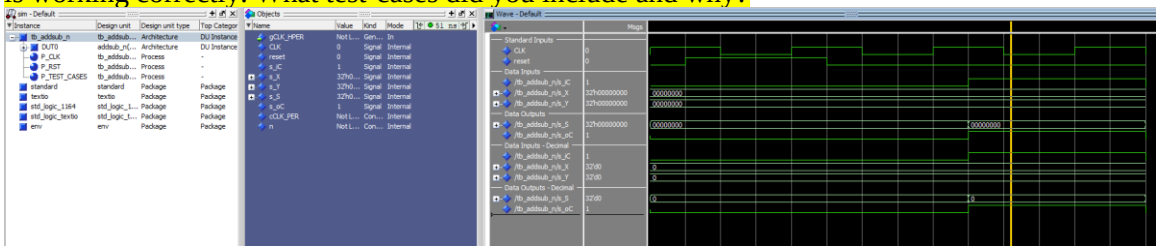
This last one is adding 11… and 22…for a result of 33…

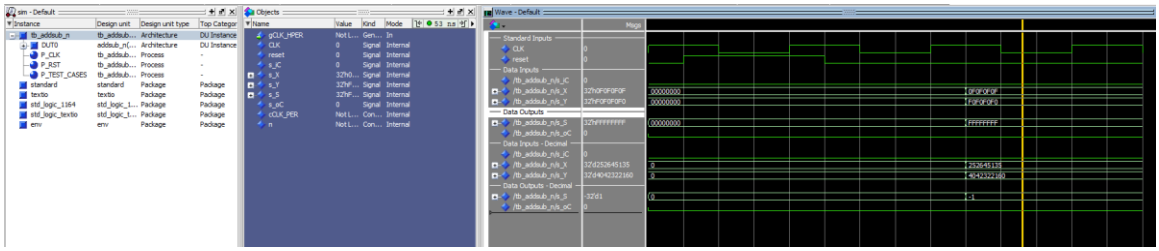
[Part 7.a] Draw a schematic (don't use a schematic capture tool) showing how an N-bit adder/subtractor with control can be implemented using only the three main components designed in earlier parts of this lab (i.e., the N-bit inverter, N-bit 2:1 mux, and N-bit adder). How is the 'nAdd_Sub' bit used?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

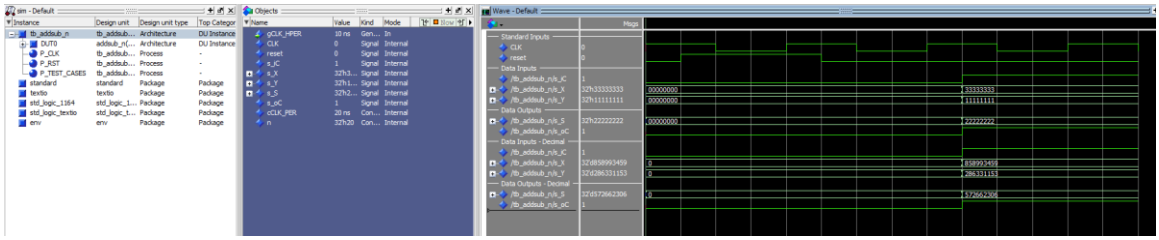
The nAdd_Sub signal is both used to toggle between Y and inverted Y along with controlling the carry bit of the adder. Both of these together create a situation where you are either just adding both X and Y when the signal is low or adding X and 2's compliment of Y when the signal is high. This addition of the 2's compliment acts just the same as subtraction.

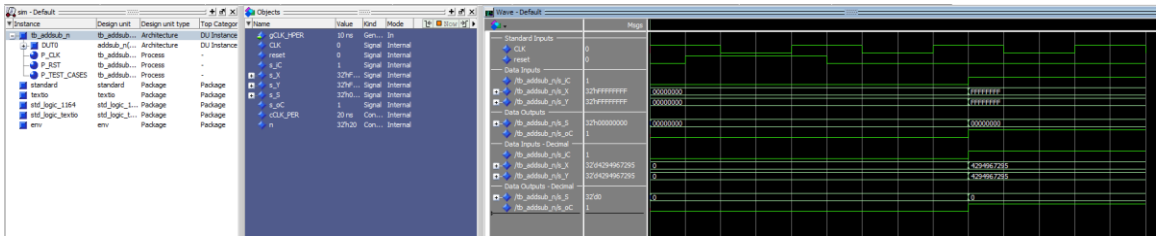


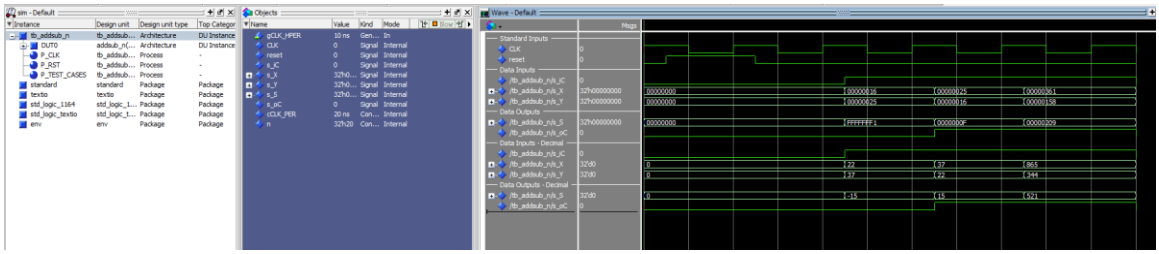
[Part 7.c] Provide multiple waveform screenshots in your write-up to confirm that this component is working correctly. What test-cases did you include and why?











I chose my test cases to provide a good insight over the typicall runcase for the adder along with some common edge cases. The first test was taking 0-0 which provided a output of 0 and a carry of 1. Next I added 0F0F0F0F + F0F0F0F0 which added up to FFFFFFFF. Next I calculated 33333333 – 11111111 for an answer of 22222222. Then I tested subtracting FFFFFFFF from it self for an answer of 0. Finally I did some smaller math problems like 22-37 = -15, 37-22 = 15 and 865 – 344 = 521.