# CprE 381 – Computer Organization and

# Assembly-Level Programming

# Proj-A Report

Lab Partners: Chimzim Ogbondah & Parth Padmanabhan

## Section / Lab Time: Section 4 Thursday 4:10-6 pm

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

1. [Part 0] Updated Project Team Contract (if applicable). With your project group members, create a list of best practices / tips for designing, compiling, and testing VHDL modules based on your experiences so far with these labs, both working individually and as a group.
   1. Delegating who will work on what, Setting time a side to make sure that either partner is not struggling or if they are how to help them move pass the roadblock
   2. Coming together to explain how each component works and the working together to integrate the design
2. [Part 1 (a)] Draw a schematic for a 1-bit ALU that supports the following operations: add/sub (both signed and unsigned), slt, and, or, xor, nand, and nor. What are the inputs and outputs that are needed?



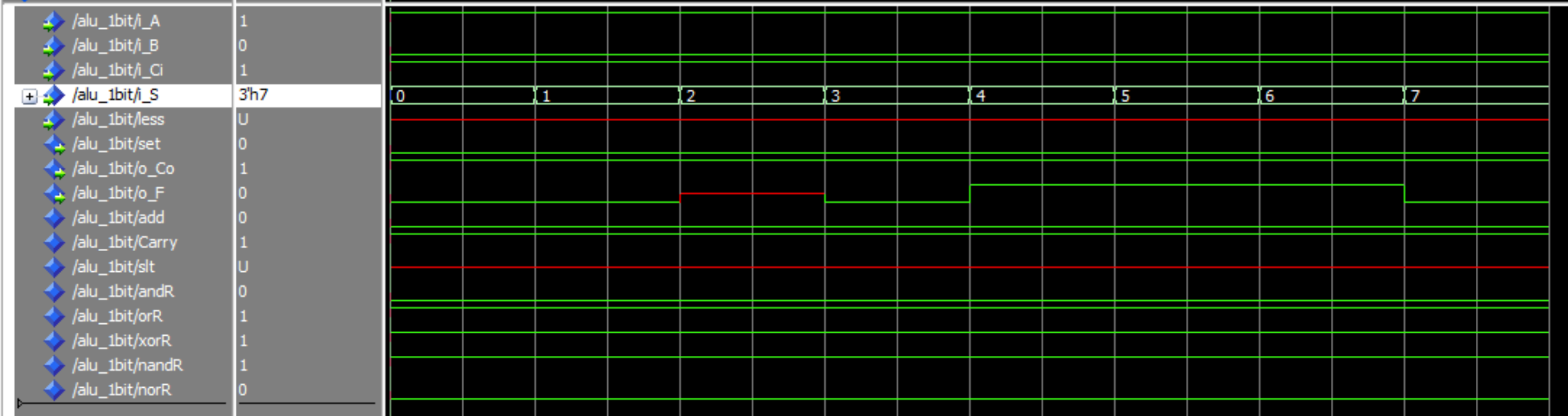
1. [Part 1 (b)] In your project writeup, describe your design in terms of the VHDL coding style you chose and the control signals that are required.

**For the 1bit ALU, I used a combination of structural and dataflow architectures. Structural was used for the full adders and the 8to1 multiplexor. It was also used for assigning values to some signals in the ALU. Dataflow was used for the logic gates like and, or, and the like.**

**The control signal used is a 3bit wide bus for the select line for the mux. Another input bit is used as the carry in for the full adder.**

1. [Part 1 (c)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.

**add and sub are the same here since you can not really perform 2s complement on 1bit numbers. As seen in the VHDL code, they share the same input line. and, or, xor, nand and nor work like you would expect them to. less did not have an input which is why slt is undefined.**



1. [Part 2 (a)] Draw a simplified schematic for this 32-bit ALU. Consider the following questions: how is Overflow calculated? How is Zero calculated? How is slt implemented?

**Overflow is calculated by xor’ing the last bits of the carry in and carry out from the ALU.**

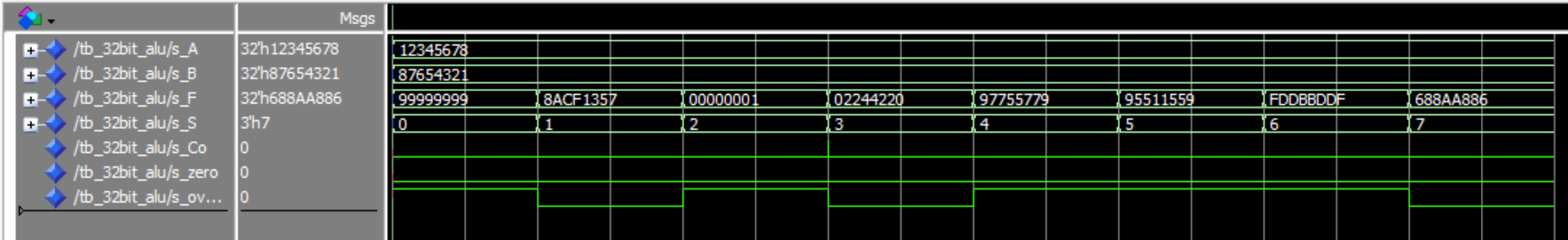
**Zero is calculated by or’ing each bit of the final output.**

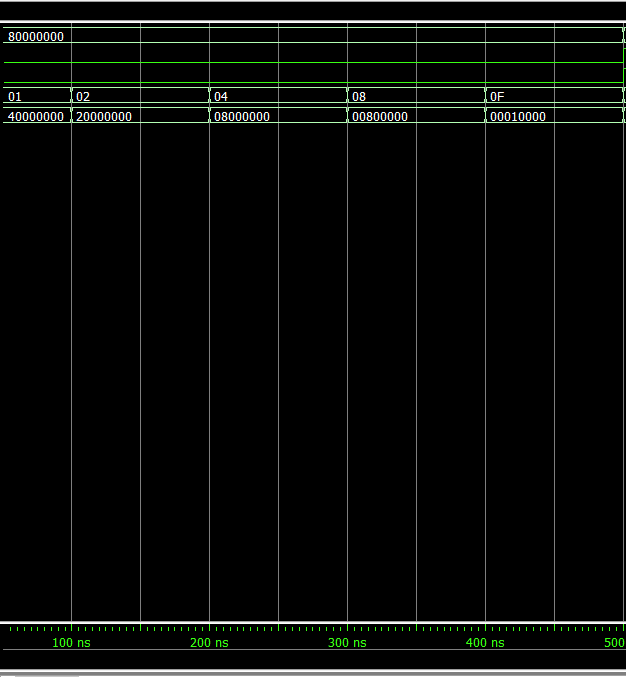
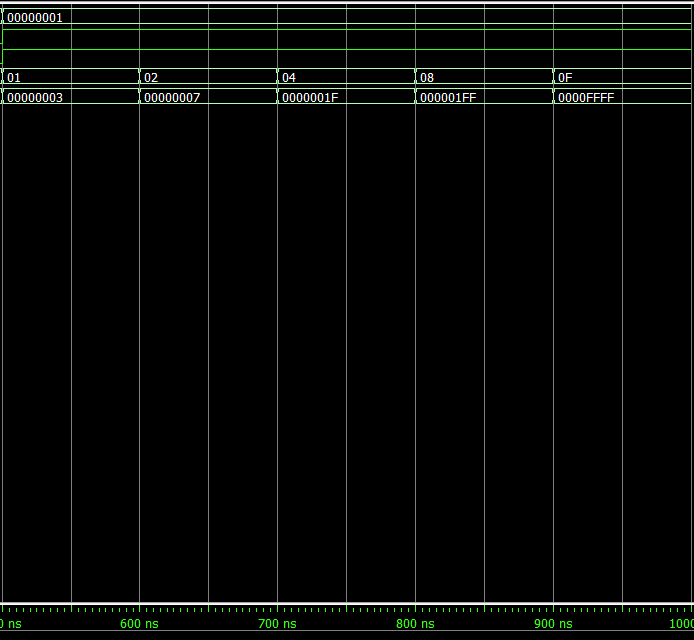
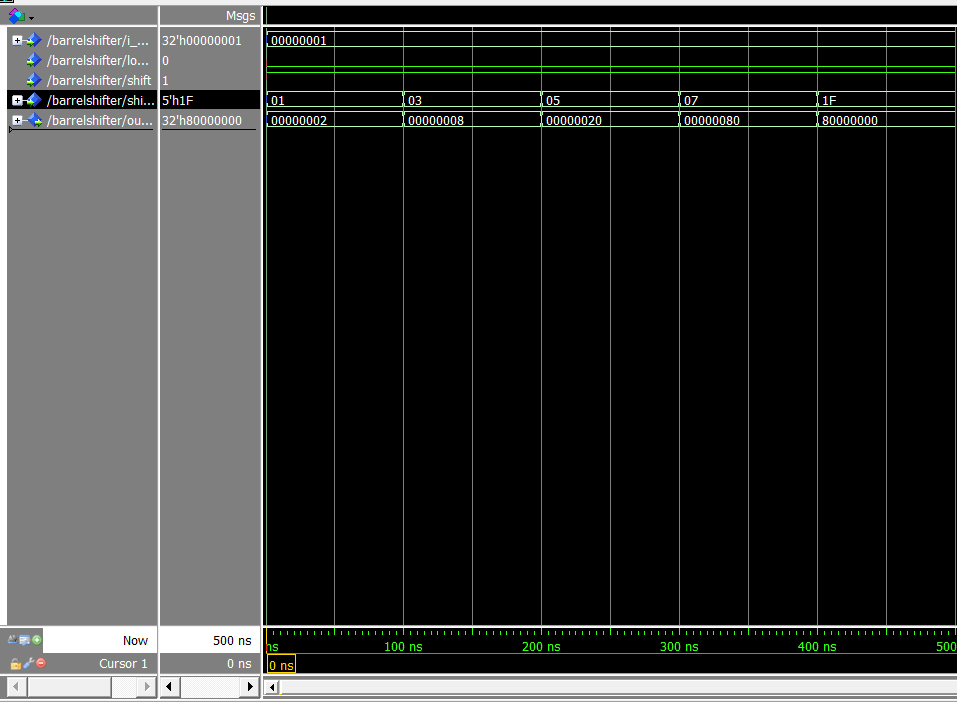
**slt is implemented by using the less and set ports in the 1bit ALU. For the zeroth bit, the ALU accepts the last bit of the output as its input for less. This is done because in the case of signed integers, the most significant bit is the sign.**

1. [Part 2 (b)] In your writeup, describe what challenges (if any) you faced in implementing this module.

**Implementing slt was challenging since VHDL had a few intricacies that I had to work around such as pulling data from an output signal.**

1. [Part 2 (c)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.

**Except for slt and sub, everything works the same as it did for the 1bit ALU, just on a larger scale. Since we can now take 2s complement of one of the inputs, the sub instruction works. As seen in the waveform, since the first input is less than the second one, the slt signal returns a 32bit ‘1’, which is the desired output.**

1. [Part 3 (a)] Describe the difference between logical (srl) and arithmetic (sra) shifts. Why does MIPS not have a sla instruction?
   1. Shift left logical shifts the data to the left and pads with zeros while arithmetic pads with 1s.
2. [Part 3 (b)] In your writeup, briefly describe how your VHDL code implements both the arithmetic and logical shifting operations.
   1. It implements both arithmetic and logical shifting by using a 2-1 mux which is given an input of 1 and 0. If the LogArth signal is set to 0 it takes zero and 1 when the signal is one. It then uses stages of shifting 1 bit, 2 bits, 4 bits, 8 bits, 16 bits and at each step adding the LogArth to the amount of bits being shifted at the end. (1, 2, 4, 8, 16) each stage feeds into the next where it either takes logArth if its to be shifted or the output from the previous bit stage shift and then continues along with the shifting pattern where the bit data is moved to the right by the amount corresponding to the shift stage.
3. [Part 3 (c)] In your writeup, explain how the right barrel shifter from part b) can be enhanced to also support left shifting operations.
   1. It allows left shifting by shifting both right and shifting left (which follows the same process of shifting as the right shifting except it flips the process) there is a Signal for shifting that is connected to a mux at the end where if 0 is selected it gives the output right shifting and if it is 1 it gives the output left shifting.
4. [Part 3 (d)] Describe how the execution of the different shifting operations corresponds to the Modelsim waveforms in your writeup.
   1. The first wave form shows right shifting where LogArth is 0 for Logical and then shift is set to 0 for right and shift amount is 01, 02, 04, 08, 0F in hex values to show how much they are shifting
   2. 
   3. This waveform shows left shifting where shift is 1, LogArth is set to 1 for arithmetic shifting and the shift amount corresponds to 01, 02, 04, 08, 0F in hex to show how each shift stage shifts the bits
   4. 
   5. The last waveform uses logical shifting on right shifting to show that values can be stacked 01, 03, 05, 07, 1F I only show left shifting as this implements the right shifting idea so it should give the same output but in reverse
   6. 

1. [Part 4(b)] Justify why your test plan is comprehensive. Include waveforms that demonstrate your test program is functioning.
2. [Part 5(c) BONUS] Justify why your test plan is comprehensive. Include waveforms that demonstrate your test program is functioning.
3. [Feedback] You must complete this section for your lab to be graded. Please complete each column **separately** for each team member; I expect it to take roughly 10 minutes (do not take more than 20 minutes).
   1. How many hours did you spend on this lab?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **During lab time** | | | **Outside of lab time** | | |
| **Team Initials** |  | PP | CO |  | PP | CO |
| Reading lab |  | 0.2 | .2 |  | 0.2 | .2 |
| Pencil/paper design |  | 0.1 | .1 |  | 0.2 | .1 |
| VHDL design |  | 0.5 | 1.0 |  | 1 | .5 |
| Assembly coding |  | 0 |  |  | 0 |  |
| Simulation |  | 0.2 | .2 |  | 0.5 | .5 |
| Debugging |  | 1 | .5 |  | 1 | .5 |
| Report writing |  | 0 | 0 |  | 0.5 |  |
| Other: |  |  |  |  |  |  |
| Total |  | 2 | 2 |  | 3.4 | 1.8 |

* 1. If you could change one thing about the lab experience, what would it be? Why?
  2. What was the most interesting part of the lab?
     1. I thought implementing a shifter only using 2-1 muxs was really cool