CDA 4203L Spring 2025
Computer System Design Lab
Lab 2 – Behavioral and Structural Verilog

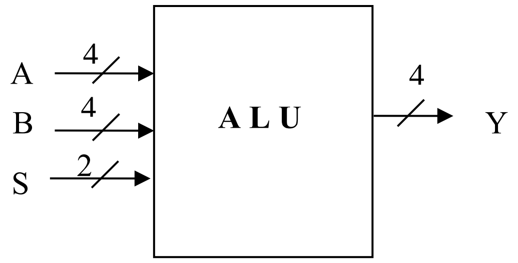
Lab Date: Feb. 4

Report Due Date: Feb. 14 on Canvas

This is an individual assignment. No teaming allowed. No demonstration is needed. No hardware is needed, use ISE as instructed.

Objectives: To learn and practice Verilog based circuit modeling and validation by simulation.

Problem: Implement an ALU (Arithmetic Logic Unit) satisfying the following functional requirements. **You should complete tutorial 2 (Tutorial_Lab2.pdf) before starting this lab.**



Function	Select (S) S1	S0	Y
Invert	0	0	$\overline{A}$
Add	0	1	A + B
Subtract	1	0	A - B
Double	1	1	2*A

Figure 1: ALU Port Interface and Function Table

1. (10 pts.) **Problem 1, Behavioral-Only Verilog:** Design an ALU in behavioral Verilog with port interface and functionality as shown in Figure 1. Test the ALU functionality by simulation. Include at least two vectors per function.

Hint for Problem 1: Construct a module with an always block with appropriate sensitivity list. Depending on your select, you would do these cases "**~A**", "**A+B**", "**A-B**", or "**2*A**" for 4-bit inputs.

2. (20 pts.) **Problem 2, Behavioral Components and Structural Top Level ALU:** Design the ALU in structural Verilog. Only use the following components: 4-bit NOT, 4-bit Full Adder, and 4-bit input/output 2-to-1 MUX. First you need to develop behavioral Verilog module for each of the above three components (3 behavioral modules). Then, you can instantiate these three components to build the ALU. Test the ALU functionality by simulation. Include at least two vectors per function. You may re-use the test bench you have created in Lab 1.

Hint for Problem 2: Construct three behavioral modules with always blocks with appropriate sensitivity list. Extend NOT and 2-to-1 MUX behavioral codes which have already been discussed in the tutorial to 4-bit blocks. In your final ALU code, you need to instantiate your four-bit 2-to-1 MUX three times for your four-bit 4-to-1 MUX (make sure your selects are correct).

You can also use the below example of always block for 4-bit adder as part of your design (it has to be designed so that you instantiate it in your structural ALU for addition and subtraction depending on your inputs/carry_in):

```
always @ (a or b or c_in) begin
assign {c_out, add_out}= a+b+c_in;
end
```

Deliverables: A concise PDF report that includes your Verilog codes, testbenches, and simulation results.

Report Organization (A template is provided on Canvas):

- ☐ Cover sheet
- ☐ Problem 1: Behavioral Verilog Code, Test Bench, and Simulation Results (Waveforms)
- ☐ Problem 2: Behavioral Verilog for three Components, ALU Structural Code, Test Bench, and Simulation Results (Waveforms)

Important:

- ☐ Do not discard your designs. They may be used as starting point for subsequent labs.
- ☐ You need to submit your report on Canvas in PDF format. No hardcopy is needed.