University of South Florida

CDA 4205L Computer Architecture Lab

Lab Report

|  |  |  |  |
| --- | --- | --- | --- |
| **Semester: Spring 2025** | | | |
| ***Experiment*** | *Number:* | 7 |
| *Date:* | Feb 26, 2025 |
|  | | | |
| ***Lab*** | *Section:* | 4 |
| *Lab TA:* | Rupal Agarwal |
|  | | | |
| ***Report*** | *Due Date:* | March 5, 2025 |
|  | | | |
| ***Group*** | *Member #1 name:* | Claude Watson |
| *Member #2 name:* | Anirudh Sesham |

# ABSTRACT:

The purpose of this lab was to design and implement an Arithmetic Logic Unit (ALU) and a Control Unit (CU) in Verilog, simulating their behavior to understand their functionality in a RISC-V architecture. The ALU was responsible for performing arithmetic and logic operations, while the CU was responsible for decoding instructions and generating control signals. This lab involved designing these modules, implementing testbenches, and verifying their correctness using simulations. Through this experiment, we gained insights into the role of ALUs and CUs in processors and learned the importance of testbenches in functional verification.

# INTRODUCTION:

This lab focuses on understanding the design and functionality of an Arithmetic Logic Unit (ALU) and a Control Unit (CU) within a processor. The ALU is responsible for executing arithmetic and logical operations, such as addition, subtraction, multiplication, division, bitwise operations, and comparisons. The CU, on the other hand, decodes instruction opcodes and generates appropriate control signals to manage data flow within the processor.

The implementation of these components was done using Verilog, and the functionality was validated through simulation in EDA Playground. A testbench was used to verify the accuracy of the ALU operations and control signal outputs of the CU. The experiment aimed to strengthen our understanding of processor design principles and the significance of testing methodologies in digital design.

# METHOD:

## Software Tools Used:

* EDA Playground: An online Verilog simulator used for writing and testing the Verilog modules.

## Files:

* alu.v – This file contains the Verilog implementation of the ALU, performing operations based on a 4-bit opcode.
* alu\_testbench.v – This file serves as a testbench for the ALU, verifying its operations using different test cases.
* cu.v – This file implements the Control Unit, decoding the opcode and generating appropriate control signals*.*
* cu\_testbench.v – A testbench that verifies the correctness of the control signals generated by the CU.

# RESULTS:

T1: Take a screenshot of your output when A and B are both positive values. Confirm that

the results are correct.

A screenshot of a computer

AI-generated content may be incorrect.

T2: Take a screenshot of your output when A is positive, and B is negative. Confirm that the

results are correct.

A screenshot of a computer

AI-generated content may be incorrect.

T3: Remove the signed modifier from A in the ALU module and rerun using the same A and

B values as in T2. Take a screenshot of your output and explain what happens.

A screenshot of a computer

AI-generated content may be incorrect.

Explanation: Since the signed modifier is removed from A, if A is negative, it will not be treated as a negative number. Instead, it will be treated as a large positive number. With this in effect, it will not produce the correct answers with operations, such as addition, subtraction, comparisons, etc.

T4: Remove the signed modifier from B in the ALU module and rerun using the same A and

B values as in T2. Take a screenshot of your output and explain what happens.

A screenshot of a computer

AI-generated content may be incorrect.

Explanation: Since the signed modifier is removed from B, if A is negative, it will not be treated as a negative number. Instead, it will be treated as a large positive number. With this in effect, it will not produce the correct answers with operations, such as addition, subtraction, comparisons, etc.

T5: Take a screenshot of your output with all “opcodes”. Confirm that the results are correct.

A screenshot of a computer

AI-generated content may be incorrect.

T6: The #delay specified in step 13 was 5 “time units”. What is the real value of these 5

“time units” in the simulation?

* The real value of these 5 times units is 5 nano seconds (ns).

T7: What is the purpose of a testbench? Why is it needed for each module (ALU and CU)?

* The purpose of a testbench is used to test and verify the functionality of a module. In this case, a testbench for ALU was needed to verify the correct implementation of operations given an input, A and B. these operations included addition, subtraction, shifting and comparison. The CU testbench was needed to verify that the correct opcode generated for each control signal.

T8: Can we obtain power, performance, area (PPA) results with the testbench? Why?

* No, a testbench cannot obtain Power, Performance, and Area (PPA) results because it is only used for functional verification in simulation. Testbenches do not model real hardware behavior, they simply check if a module produces the expected outputs.

# DISCUSSION:

The design and implementation of an ALU and CU provided a hands-on approach to understanding processor operations. Through the ALU module, we successfully implemented arithmetic, logical, and shift operations. The correctness of these operations was verified using testbenches, which confirmed that the expected outputs were produced. However, the experiment also highlighted the importance of signed and unsigned data representation, as removing the signed modifier led to incorrect results due to how binary values are interpreted.

For the CU module, the primary challenge was ensuring the correct mapping of opcodes to control signals. Misinterpreting an opcode could lead to incorrect processor behavior. By carefully constructing the testbench and testing multiple opcodes, we ensured accurate control signal generation. Future work could involve extending this CU design to handle more complex instruction sets or integrating a complete datapath to simulate full instruction execution.

# CONCLUSION:

This lab provided a comprehensive understanding of ALU and CU design in a RISC-V processor. By implementing these modules in Verilog and verifying their functionality using testbenches, we gained valuable insights into digital design and processor control logic. The importance of thorough testing was emphasized, as even minor errors in opcode interpretation or signed data handling could lead to incorrect outputs.