**Table of Contents**



| **Course Title:** | **COE** |
| --- | --- |
| **Course Number:** | **328** |
| **Semester/Year (e.g.F2016)** | **F2024** |

| **Instructor:** | **Arghavan Asad** |
| --- | --- |

| *Assignment/Lab Number:* | *6* |
| --- | --- |
| *Assignment/Lab Title:* | *Design of A Simple Central Processing Unit* |

| *:*  *Submission Date* | **2024-12-12** |
| --- | --- |
| *Due Date:* | *2024-12-12* |

| **Student**    **LAST Name** | **Student**    **FIRST Name** | **Student**    **Number** | **Section** | **Signature\*** |
| --- | --- | --- | --- | --- |
| **Nimalraaj** | **Sharllene** | **501180933** | **21** | **Sharllene Nimalraaj** |
|  |  |  |  |  |
|  |  |  |  |  |

## Table of Contents

### Introduction

### Description of the overall lab and the components

### Components

### Description of components: Latch1, Latch2, 4:16 Decoder and FSM

### Problem Set 1 ALU

### Description of the problem

### Purpose of inputs and outputs

### Table of Microcodes

### Block schematic and waveform

### Problem Set 2 ALU

### Description of the problem

### Purpose of inputs and outputs

### Table of Microcodes

### Block schematic and waveform

### Problem Set 3 ALU

### Description of the problem

### Purpose of inputs and outputs

### Table of Microcodes

### Block schematic and waveform

### Conclusion

### 

**Introduction**

# Introduction:

This lab focuses on the implementation and design of a simple CPU in a VHDL environment, which will be implemented on an FPGA board. The CPU consists of an arithmetic and logic unit (ALU), a control unit, a memory and an input/output unit. The ALU performs various arithmetic and logical operations on 2 8-bit inputs A and B and produces an output. The control unit then supplies the opcodes to the ALU, which defines the operations to be performed. The memory is simulated by a finite-state machine (FSM), which generates the current state signal that controls the ALU’s operation.

**Components**

# Components:

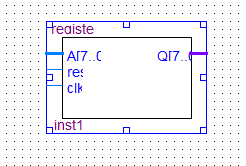
During this lab the following components were used, Latch1 and 2, FSM, and a 4to16 decoder.

## Basic Latch (Latch 1)

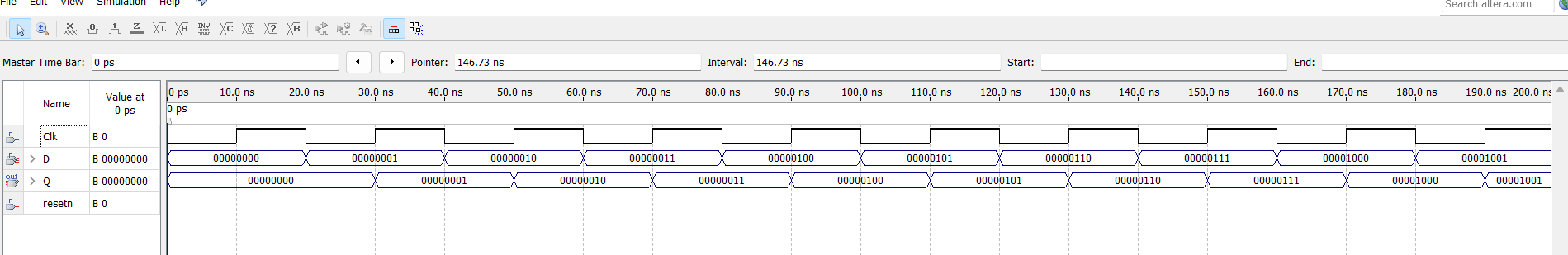
The basic latch outlined in this lab is the main component of the storage unit. It takes in a binary number as an input and gives it out as an output for each cycle.

The circuit uses 2 basic latches for each binary number.

Circuit Diagram for Latch:



Waveform for Latch:



Truth Table for Latch1 (and Latch2):

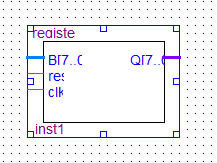
| CLK | D | Q(t+1) |
| --- | --- | --- |
| 0 | X | Q(t) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Latch 2

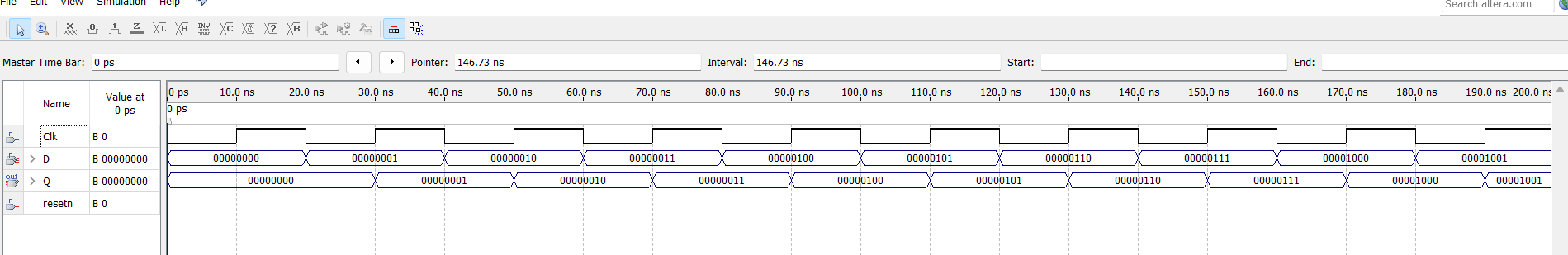
## Description:

Latch 2 functions similarly to Latch 1 but is used for a different input channel. This modular design allows the system to process multiple data streams concurrently. The circuit diagram and truth table illustrate its role in the CPU, and the waveform file confirms its operation under clocked conditions.

Component’s Circuit Diagram:



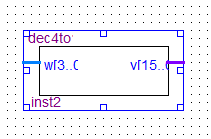
Component Waveform:



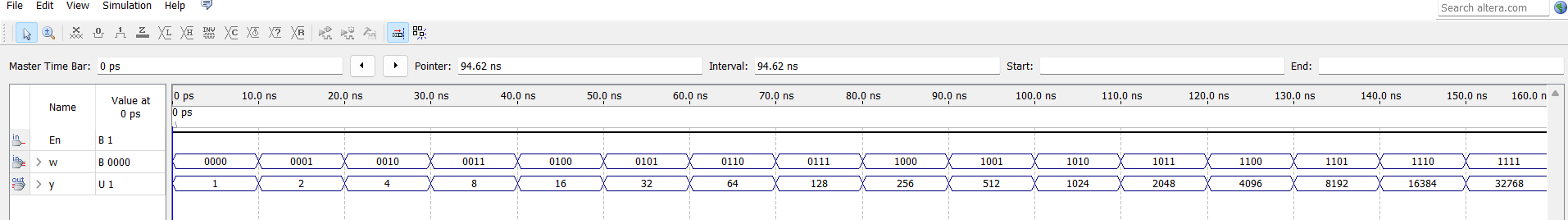
## 4:16 Decoder

The 4:16 Decoder was a component designed to take in the 4-bit state output of the FSM and give out a unique 16-bit output for each state, which selects one operation for the ALU. The mapping facilitates the control of multiple ALU functions using a compact opcode format. The decoder’s truth table, block diagram, and simulation waveforms demonstrate its role in the control unit.

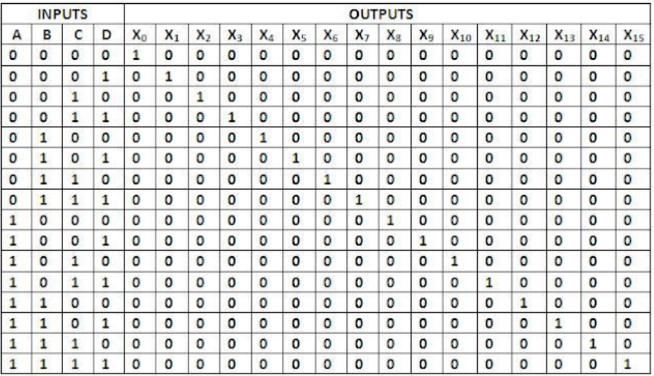
Component’s Circuit Diagram:



Waveform:



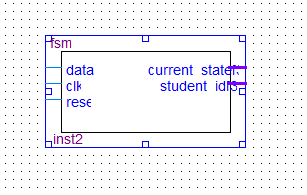
Truth Table for Component:



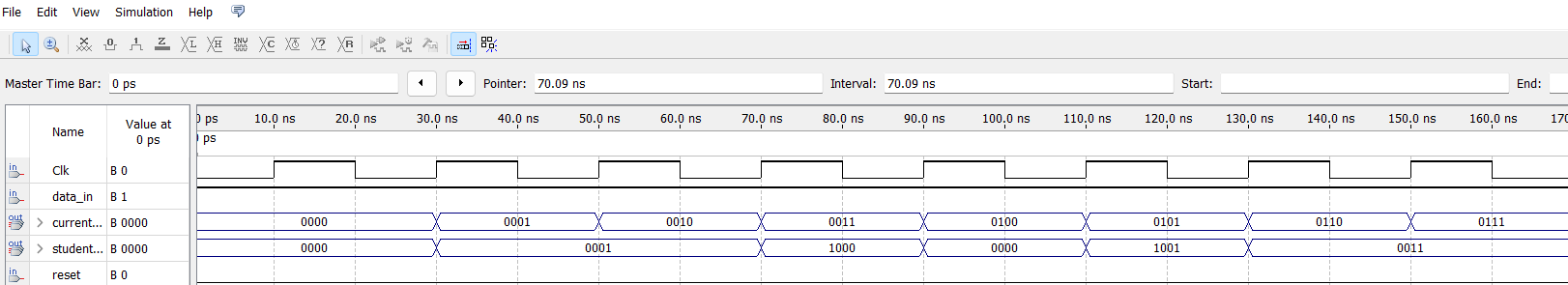
## Finite State Machine (FSM)

The FSM serves as a program counter, cycling through predefined states to sequence operations. Each state corresponds to an opcode fed into the decoder. The FSM ensures orderly execution of instructions, with a state transition diagram and simulation results highlighting its operation. Modifications were made to adapt the FSM to our student numbers.

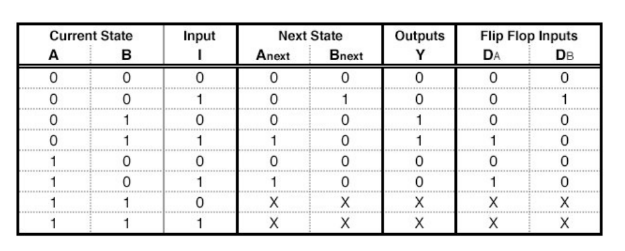
Circuit Diagram:



Waveform:



Truth Table for Component:



**ALU**

**Arithmetic Logical Unit (ALU)**

## ALU 1:

The ALU (Arithmetic Logic Unit) serves as the processor that determines the operation to perform on the inputs based on the state of the FSM (Finite State Machine). It has five inputs:

1. **Clock**
2. **A** (binary input)
3. **B** (binary input)
4. **student\_id**
5. **OP** (operation code)

Inputs **A** and **B** are binary numbers that act as the inputs to the storage unit, provided that **data\_in** is set to 1. The **Clock** input toggles between 0 and 1. On a rising edge (when the clock transitions from 0 to 1), the ALU reads the value of the **OP** input, which is produced by the decoder. The ALU then executes a switch statement that matches the value of **OP** to a corresponding case, performing the specified operation on **A** and **B** as outlined in the table provided in the lab manual.

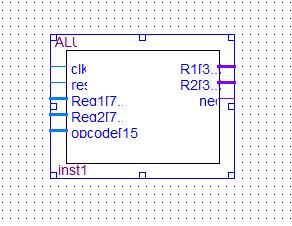
In this application of the ALU, student\_id is not used.

There are 3 outputs for the ALU;

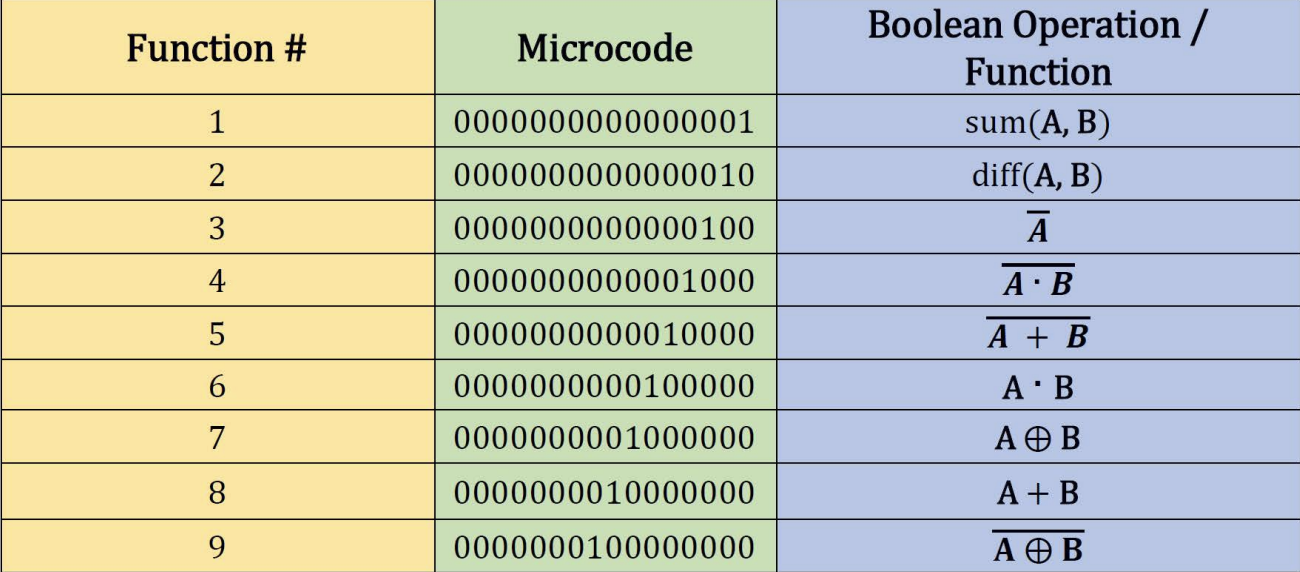
1. Neg
2. R1
3. R2

Neg outputs 1 if the number is negative and lights up an LED on the 7-segment display. R1 and R2 are 4-bit outputs, which can give an 8-bit output when combined. Each bit goes to the 7-segment display to output the result in hexadecimal.

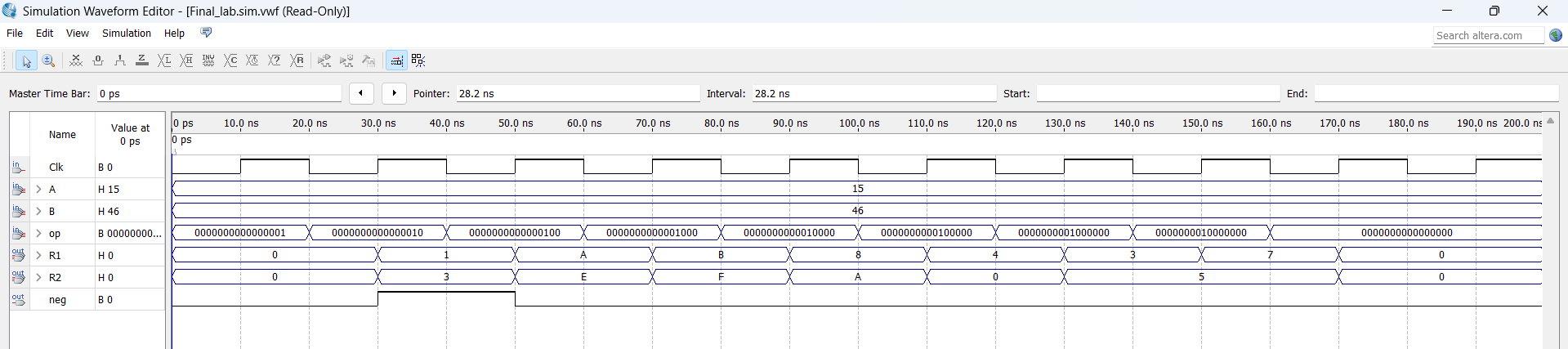
BDF:



Microcode, and Waveform



*Table 3.1: ALU problem set 1 microcode.*

****

## ALU 2:

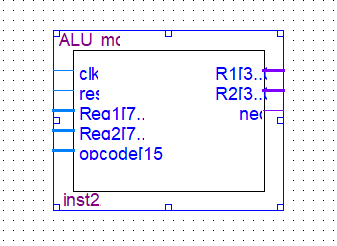
The Arithmetic Logic Unit (ALU) is the component responsible for determining the operation to perform on the inputs, guided by the state of the Finite State Machine (FSM). It accepts the following five inputs:

1. **Clock**
2. **A**
3. **B**
4. **student\_id**
5. **OP**

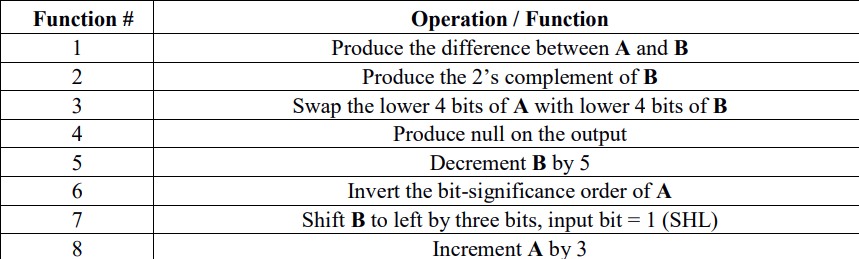
Inputs **A** and **B** are binary numbers that serve as the inputs to the storage unit whenever **data\_in** is set to 1. The **Clock** input alternates between 0 and 1. On the rising edge (when the Clock transitions from 0 to 1), the ALU reads the value of the **OP** input, which is the output of the decoder. It then executes a corresponding operation on **A** and **B** using a switch statement, where each case corresponds to a specific operation as outlined in the lab manual.

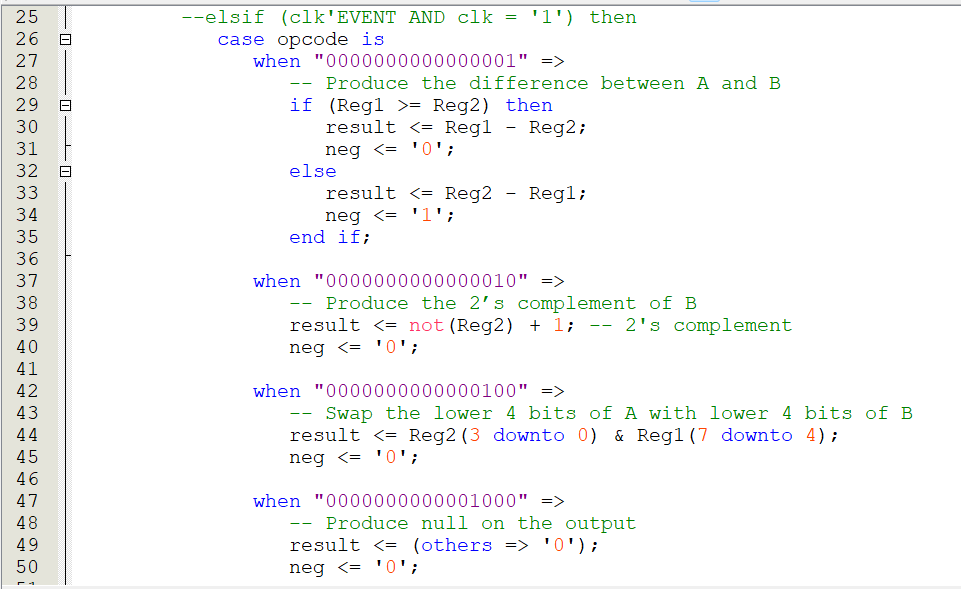
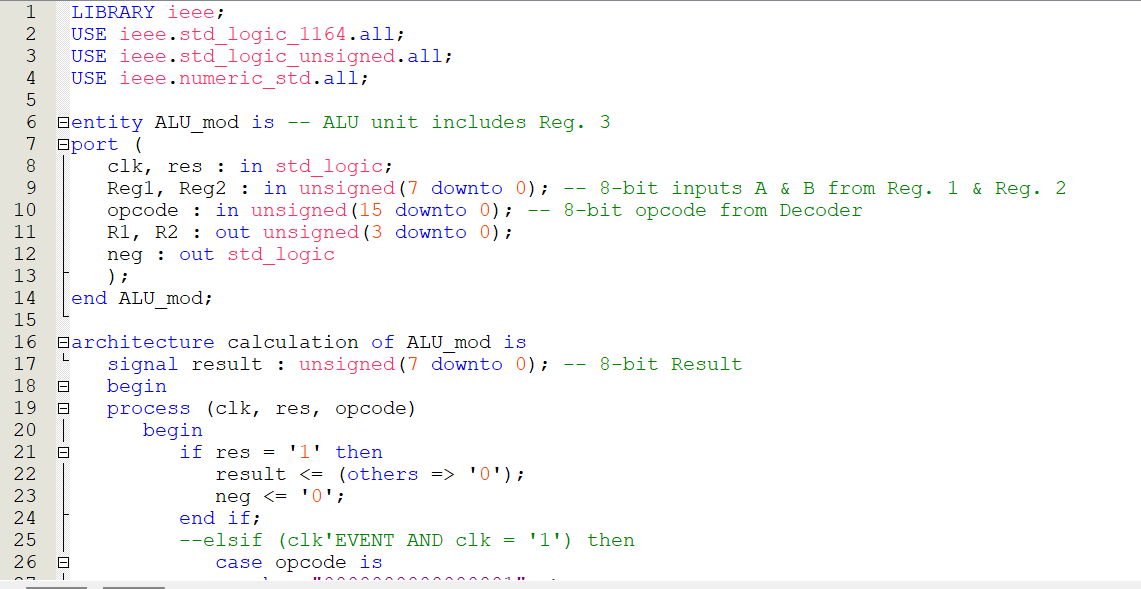
The **student\_id** input serves no functional purpose in this part.

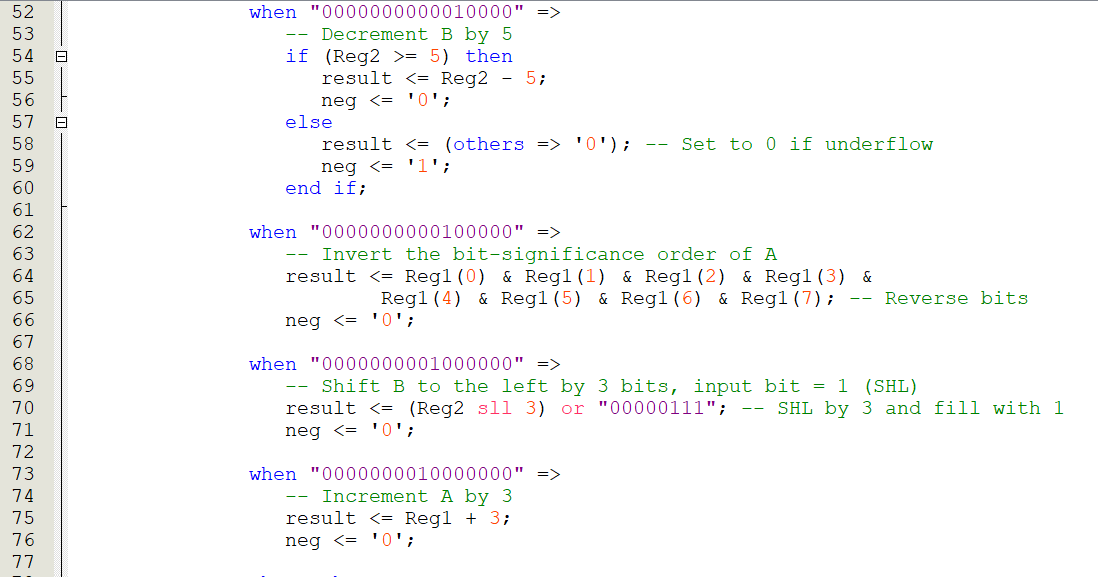
BDF:

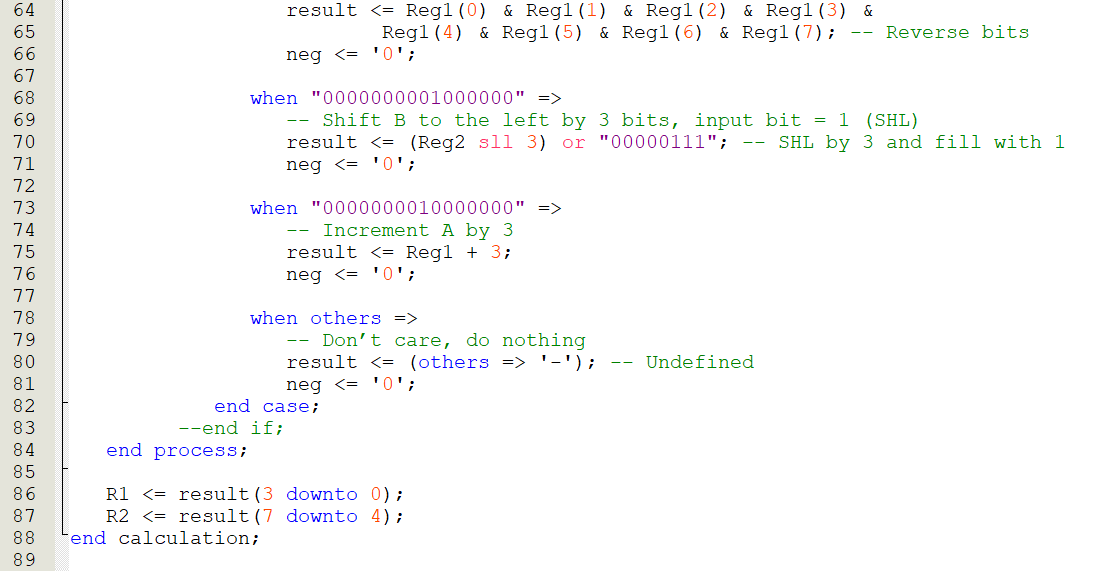


Microcode:

****







### 

## ALU 3:

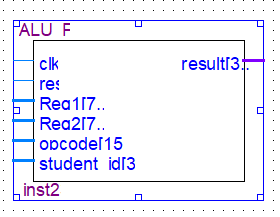
ALU from Problem 1 is used to incorporate an additional data input — the **student\_id** — which is provided as an output from the FSM component of the Control Unit. The updated ALU3 processes three inputs: **Reg1**, **Reg2**, and **student\_id**.

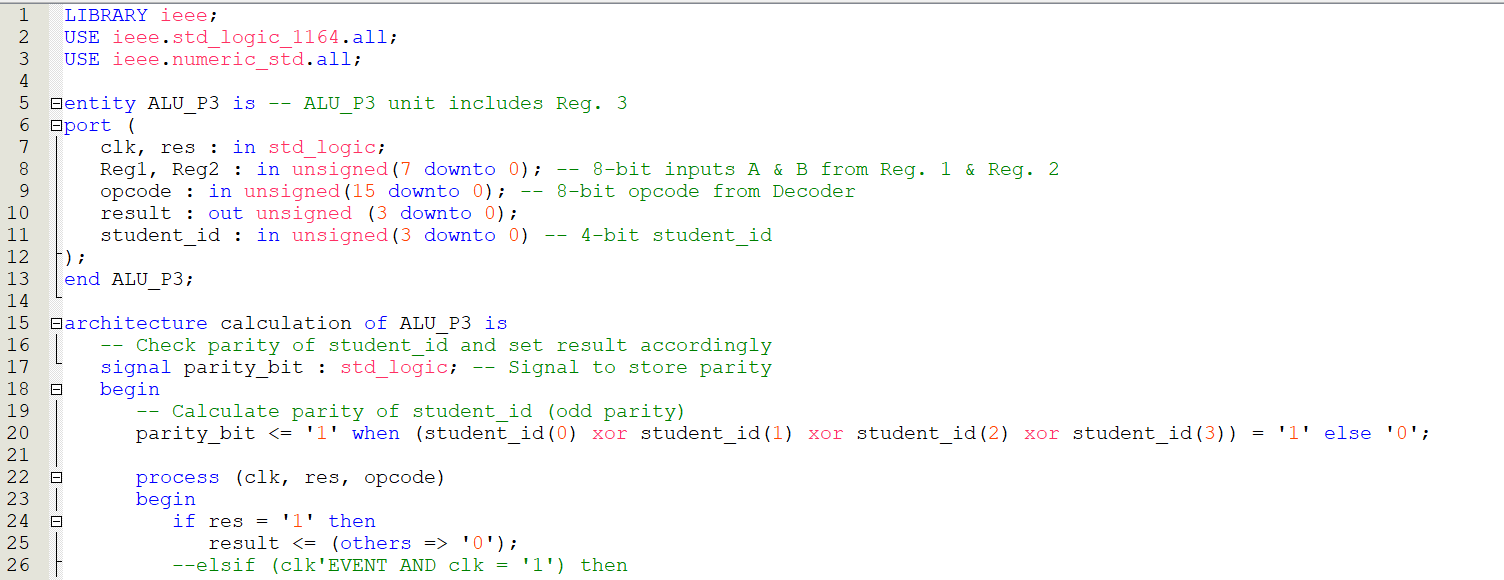
In this part, the **student\_id** input is utilized by the ALU to determine if its least significant bit (LSB) is 1 or 0. Since an LSB of 1 indicates an odd number and an LSB of 0 indicates an even number, the ALU assigns a new output, **E**, which outputs 1 for odd numbers and 0 for even numbers. This **E** signal is then sent to the 7-segment display, where an **if statement** controls the display to show **"n"** for odd numbers and **"y"** for even numbers.

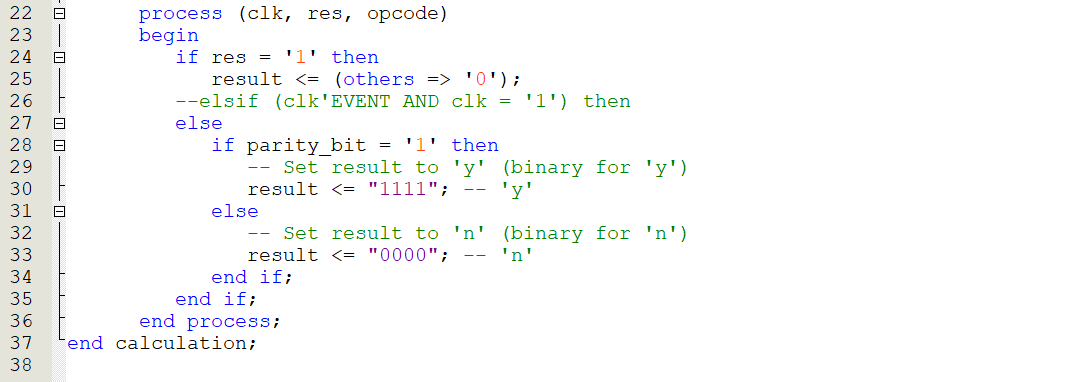
Function that was chosen: **FUNCTION 3 (below)**

****

BDF Screenshot:

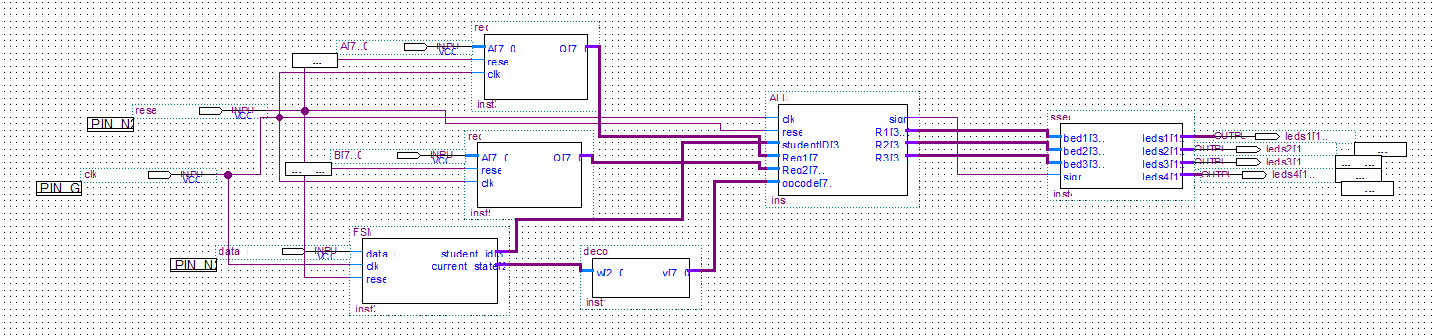






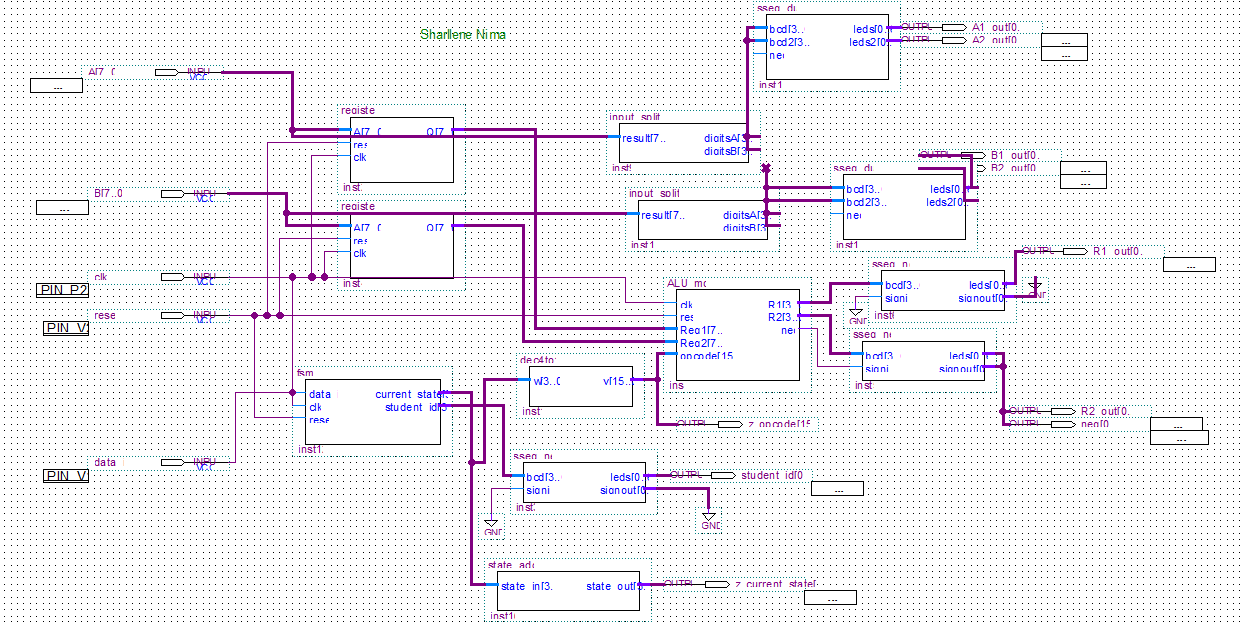
**Circuit Diagrams**

# Circuit Diagrams:



Circuit Version 1 with ALU 1

After realizing that the circuit diagram needed to be changed for ALU2 and 3, it was completely redesigned to the one below.



Circuit Diagram for ALU 2

**References/Conclusion**

# References

Brown, S. D., & Vranesic, Z. G. (2009). *Fundamentals of Digital Logic with VHDL*

*Design*. New York, United States: McGraw-Hill Education.

# Conclusion:

The successful design and implementation of a simple Central Processing Unit (CPU) provided valuable insight into the fundamental components and operation of modern computer architecture. This lab emphasized the integration and coordination of key subcomponents, including the Arithmetic Logic Unit (ALU), Control Unit (incorporating a finite state machine and decoder), registers, and input/output displays.

Through the design and synthesis process, concepts such as data storage, control sequencing, and functional simulation were applied. By using VHDL for hardware description, the CPU's operational logic was effectively translated into a functional hardware model, capable of executing arithmetic and logical operations. The structured approach of developing each subcomponent independently before integrating them into a complete system reinforced the modular nature of digital design.

Additionally, functional testing and simulation using Quartus software allowed for the verification and validation of the CPU's performance. The waveform analysis confirmed that the CPU correctly executed the specified operations, cycling through control states and producing accurate results on the 7-segment displays.

This lab reinforced key principles of digital system design, including modular design, synchronization, and binary arithmetic. The hands-on experience with FPGA implementation bridged the gap between theoretical knowledge and practical application. The ability to design, simulate, and test a functioning CPU lays a strong foundation for more complex digital design projects in the future.