



# Faculty of Engineering & Architectural Science

Course Number: COE 328

Course Title: Digital Systems

Semester: F2023

Instructor: Arghavan Asad

TA: Nahid Sahelgozin


Lab 6

Design of a Simple General-Purpose Processor

Section 14

Submission Date: December 4, 2023

Due Date: December 4, 2023

Name	Student ID	Signature
Rose Pagano	65663	

*\*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic*

## Table of Contents

Introduction.....	2
Data & Analysis.....	3
Components.....	3
Latch 1 & 2.....	3
Finite State Machine.....	5
4 to 16 Decoder.....	7
Part I: The Initial Design.....	9
Part II: Modified ALU Core.....	12
Part III: Modified FSM.....	15
Conclusion.....	17

## Introduction

The goal of this experiment was to build a simple general purpose processor that was modified in various ways to solve a great variety of problems. Through utilizing the Quartus II software and Cyclone II FPGA board, this simple processor was able to be coded using VHDL programming language to take an input, process it, and produce an output. There were several components used to achieve this. The first component that was created was a Register (latch), which is a storage component that played an important role in the input data which will be discussed later. The second and third component was a part of the control unit of the processor which was the finite state machine and decoder. This machine has a certain amount of states that interact with each other (jumping in between states) depending on the data in. The decoder takes in the state of the finite state machine and depending on that state will produce a certain bit output depending on what type of decoder it is. Lastly, is the arithmetic logic unit which is the main component where most of the processing is done, where its three main inputs are the two numbers that are used to do a certain operation and the other is the control input which controls what operation is done. In a general sense, these units work together to process information. By the Storage unit being the inputs into the arithmetic logic unit and the control unit controlling which operation occurs, information can then be processed.

## Data & Analysis

For this design, the student number of 501165663 will be used. This means by two primary inputs of  $A = (56)_{16}$  and  $B = (63)_{16}$ .

## Components:

### Latch 1 & 2

circuit diagram of VHDL code:

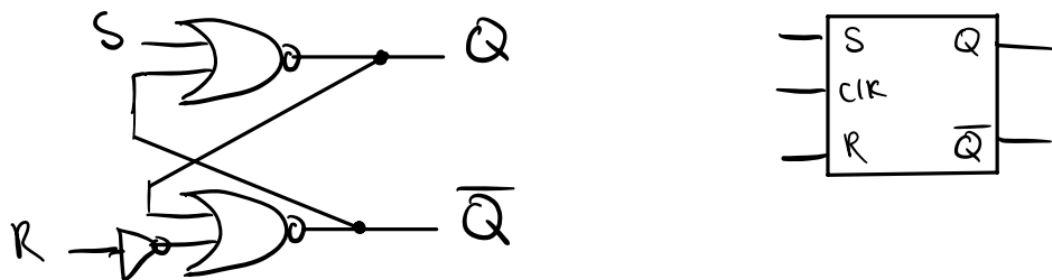


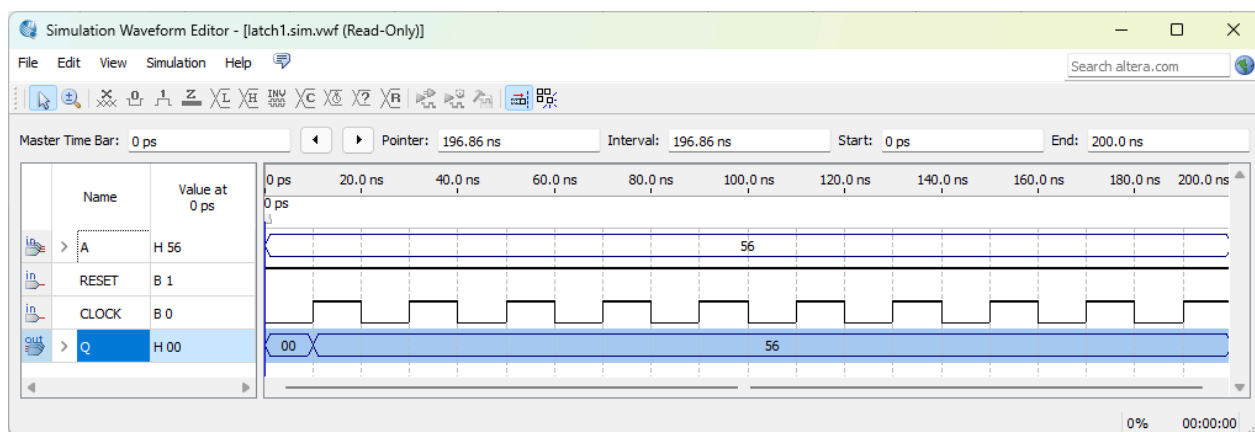
Figure 1: Hand Drawn circuit diagram of the latch component.

SR-Latches are the most simple storage device used for various sequential circuits that require a level sensitive clock cycle. This means that when the clock is equal to one the input in this case 'A' is pushed into the storage device and is stored there until a new 'A' value is pushed into the or the Reset is set to 1. In this case, the code was modified that when reset is equal to 0 the circuit resets. Due to the nature of the latch, it is shown that there will be a time delay for each clock cycle for the value that we are setting takes one cycle to reach the next component.

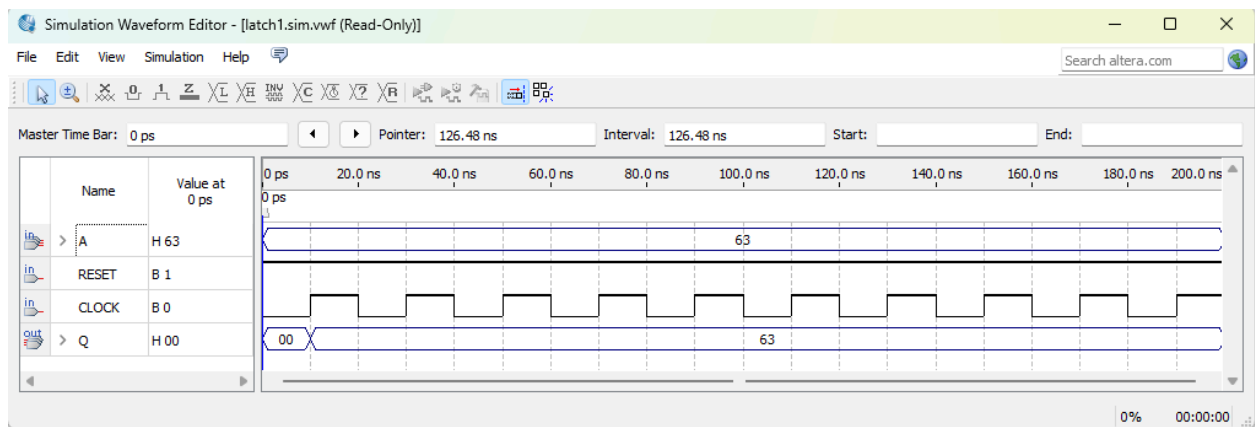
**Table 1: Basic S-R Latch**

CLOCK	S	RESET	Q(t+1)
-------	---	-------	--------

0	0	0	Q(t)
1	0	1	0
1	1	0	1
1	1	1	0
0	1	1	Q(t)



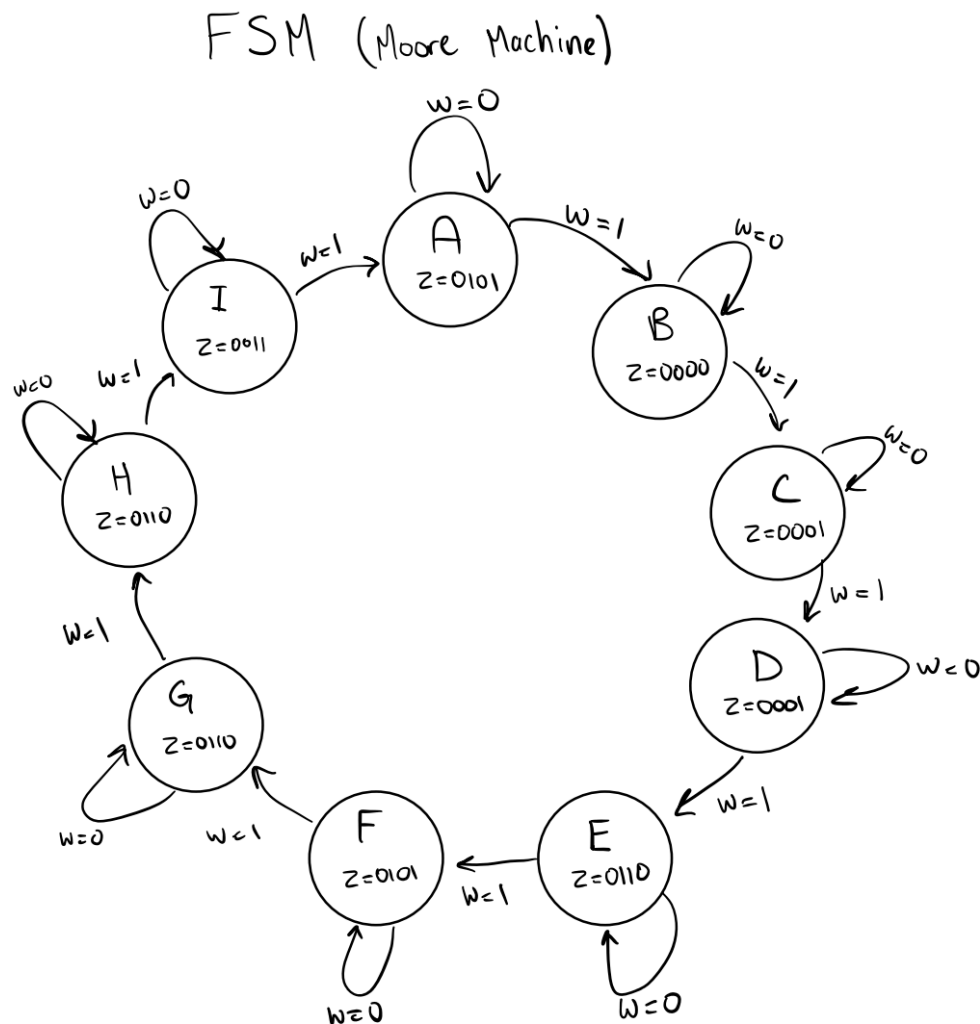
**Figure 2:** Latch 1 setting 56 each positive (1) part of the clock, and not changing during the negative part of the clock (0).



**Figure 3:** Latch 1 setting 63 each positive (1) part of the clock, and not changing during the negative part of the clock (0).

## The Finite State Machine

The finite state machine that was used in this experiment was the Moore finite state machine. This machine works by the primary input “data in” must have a value of 1 to move up in states. This FSM is quite simple in the sense that it starts at state 0 and continues until it reaches state 8 and then restarts. Each one of these states refer to the index of the student number (from 0 to 8) {501165663}. For example when the current state is 1 the primary output is 0.



**Figure 4:** Above is the the state diagram of the finite state machine

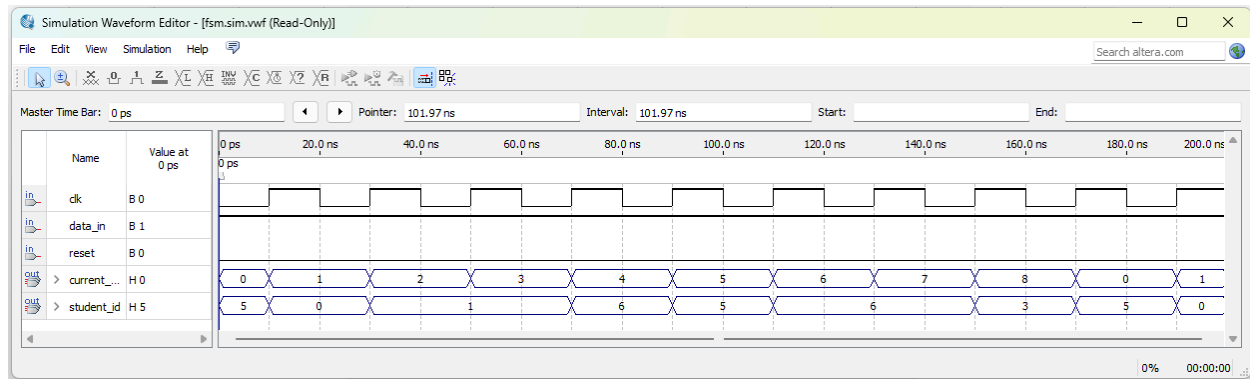
**Table 2:** Below is the state table where we simplify the value of each state to simple letters.

Present State	Next State		Primary Output	
	w = 0	w = 1	z (binary)	z (decimal)
A	A	B	0101	5
B	B	C	0000	0
C	C	D	0001	1
D	D	E	0001	1
E	E	F	0110	6
F	F	G	0101	5
G	G	H	0110	6
H	H	I	0110	6
I	I	A	0011	3

**Table 3:** Below is the state assign table where we assign somewhat arbitrary numbers to each letter to convert the state table to binary

Present State	Next State		Primary Output
$y_3y_2y_1y_0$	w = 0 $Y_3Y_2Y_1Y_0$	w = 1 $Y_3Y_2Y_1Y_0$	$Z_3Z_2Z_1Z_0$
0000	0000	0001	0101
0001	0001	0010	000
0010	0010	0011	0001
0011	0011	0100	0001
0100	0100	0101	0110
0101	0101	0110	0101

0110	0110	0111	0110
0111	0111	1000	0110
1000	1000	0000	0011



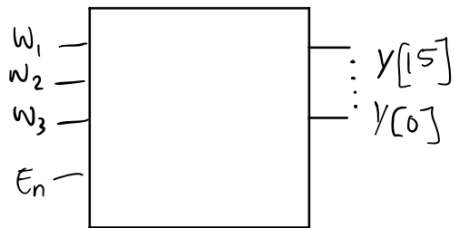
**Figure 5:** The waveform of the finite state machine shows that each state corresponds to a number in the student number. Display is in hexadecimal.

## The 4 to 16 decoder

The 4 to 16 decoder inputs a 4 bit number and returns a 16 bit number with a '1' the number of bits away from the LSB which the 4 bit number corresponds to. There is also a 1 bit enable input which enables the number to be imputed.

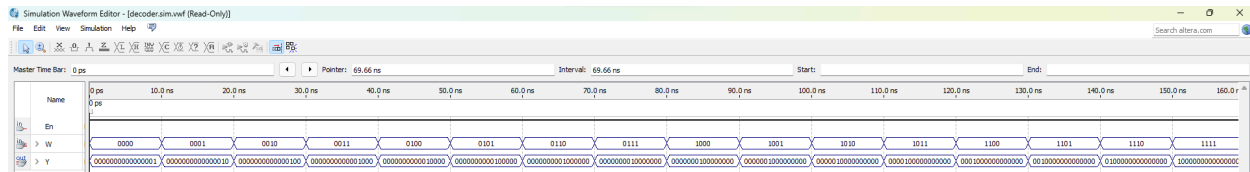


4 to 16 decoder

**Figure 6:** Hand Drawn component of the 4 to 16 decoder.**Table 4:** Truth table for a 4 to 16 decoder

Enable	W	Y
1	0000	0000000000000001
1	0001	0000000000000010
1	0010	0000000000000100
1	0011	0000000000001000
1	0100	0000000000010000
1	0101	0000000000100000
1	0110	0000000001000000
1	0111	0000000010000000
1	1000	0000000100000000
1	1001	0000001000000000
1	1010	0000010000000000
1	1011	0000100000000000
1	1100	0001000000000000
1	1101	0010000000000000

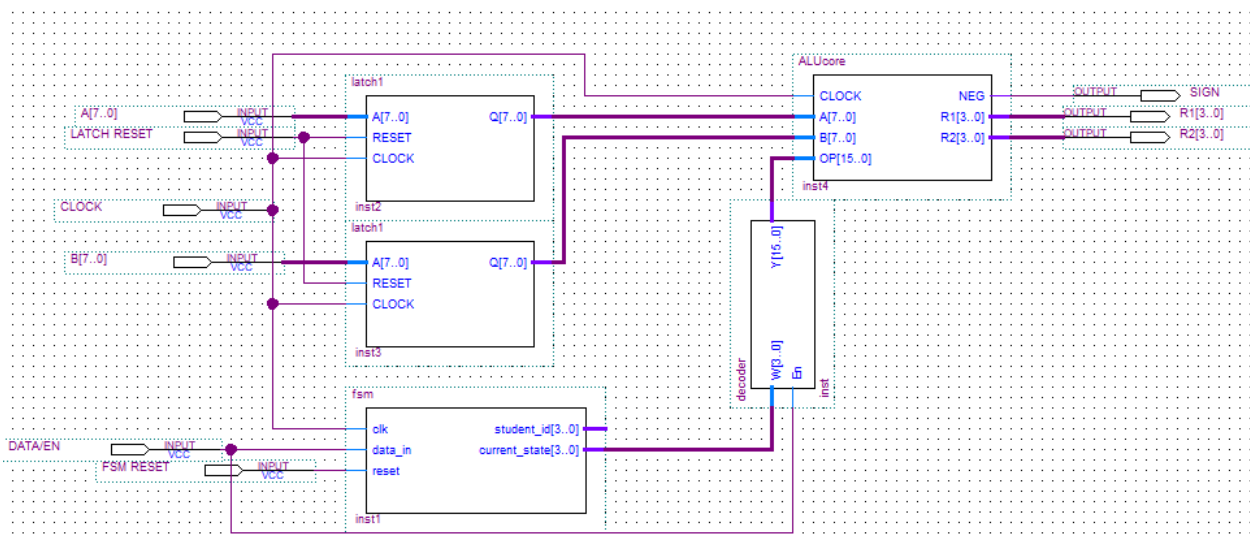
1	1110	0100000000000000
1	1111	1000000000000000



**Figure 7:** The output of the decoder which matches the table

## Part I: The Initial Design

The purpose of the design of this component was to take two inputs and do 9 different arithmetic operations, consecutively. This was achieved by having 3 main sections of the design. Firstly, the storage unit, which was two latches that stored each input. Secondly, an arithmetic unit which does the operation for that clock cycle. Lastly, the control unit which controls which operation will be performed based on the state of the finite state machine then converted into a 16 bit number using the decoder and then sent through the arithmetic unit to determine which operation is executed.



**Figure 7:** The schematic block diagram of the complete design (excluding seven segment display for simplicity).

Purpose of each input and output:

There are 6 main inputs in this design which all play an important role in the execution of this design. The 'A' and 'B' represent the two 8 bit inputs that experience arithmetic function. The next input is the latch reset that resets all the latches to 0. There is the clock that clocked all storage elements along with the ALU. There is data in/enable which are inputs into the FSM and the decoder to give them ability to do the combinational work that it needs to do. Lastly is the FSM reset which resets the finite state machine which has to be different from the latch reset due to the way it was coded to be one due to the inputs on the FPGA board during the demonstration part of this lab.

There are 3 main outputs which are all 4 bit numbers. There's R1 which is the LSB of the number and there's R2 which is the MSB of the number. There is also a negative slot. During the demonstration part of this lab, there were seven segment displays in order for these output displays to look like hexadecimal numbers.

## ALU calculations - Problem set 1

$$A = (56)_{16} = (0101 \ 0110)_2 = (86)_{10} \quad B = (63)_{16} = (0110 \ 0011)_2 = (99)_{10}$$

Sum(A, B)

$$\begin{array}{r} 56 \\ + 63 \\ \hline B9 \end{array}$$

Difference (A, B)

$$\begin{array}{r} 86 \\ - 99 \\ \hline -13 \end{array} \quad \begin{array}{l} \text{(in decimal)} \\ \text{(in hex)} \end{array} \quad \rightarrow -D$$

 $\overline{A}$ 

$$\begin{array}{r} 0101 \ 0110 \\ = 1010 \ 1001 \\ A \quad 9 \end{array}$$

 $\overline{A \cdot B}$ 

$$\begin{array}{l} = 0101 \ 0110 \cdot 0110 \ 0011 \\ = 01000010 \\ = 10111101 \\ = B \ D \end{array}$$

 $\overline{A + B}$ 

$$\begin{array}{l} = 0101 \ 0110 + 0110 \ 0011 \\ = 0111 \ 0111 \\ = 1000 \ 1000 \\ = 8 \ 8 \end{array}$$

 $A \cdot B$ 

$$\begin{array}{l} = 0101 \ 0110 \cdot 0110 \ 0011 \\ = 01000010 \\ = 4 \ 2 \end{array}$$

 $A \oplus B$ 

$$\begin{array}{l} = 0101 \ 0110 \oplus 0110 \ 0011 \\ = 0011 \ 0101 \\ = 3 \ 5 \end{array}$$

 $A + B$ 

$$\begin{array}{l} = 0101 \ 0110 + 0110 \ 0011 \\ = 0111 \ 0111 \\ = 7 \ 7 \end{array}$$

 $\overline{A \oplus B}$ 

$$\begin{array}{l} = 0101 \ 0110 \oplus 0110 \ 0011 \\ = 0011 \ 0101 \\ = 1100 \ 1010 \\ C \quad A \end{array}$$

**Figure 8:** Hand calculations of the proper output of the design.**Table 5:** Microcode generated by ALU

Function #	Microcode	Boolean Operation/Function	Expected Output of $A = (56)_{16}$ $B = (63)_{16}$ (in hexadecimal)
1	00000000000000000001	sum(A,B)	B9
2	00000000000000000010	diff(A,B)	-0D

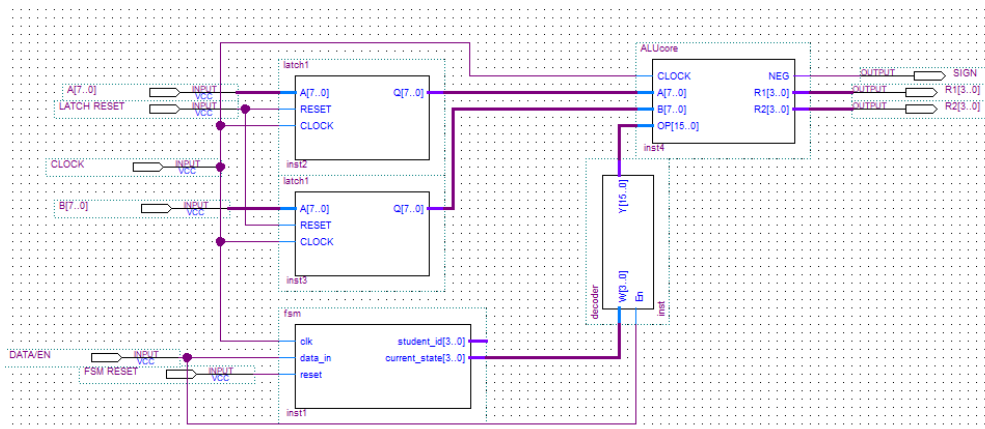
3	0000000000000100	$\overline{A}$	A9
4	0000000000001000	$\overline{A \bullet B}$	BD
5	0000000000010000	$\overline{A + B}$	88
6	0000000000100000	$A \bullet B$	42
7	0000000001000000	$A \oplus B$	35
8	0000000010000000	$A + B$	77
9	0000000100000000	$\overline{A \oplus B}$	CA



**Figure 9:** The waveform of problem set 1. Note the time delay due to the pushing of the latch storage component.

## Part II: Modified ALU

The purpose of the design of this component was to take two inputs and do 9 different arithmetic operations from the initial design, consecutively. It is very similar to initial design except the arithmetic operations have changed.



**Figure 10:** The schematic block diagram of the modified ALU (excluding seven segment display for simplicity).

Purpose of each input and output: The inputs and outputs are the exact same as the initial design.

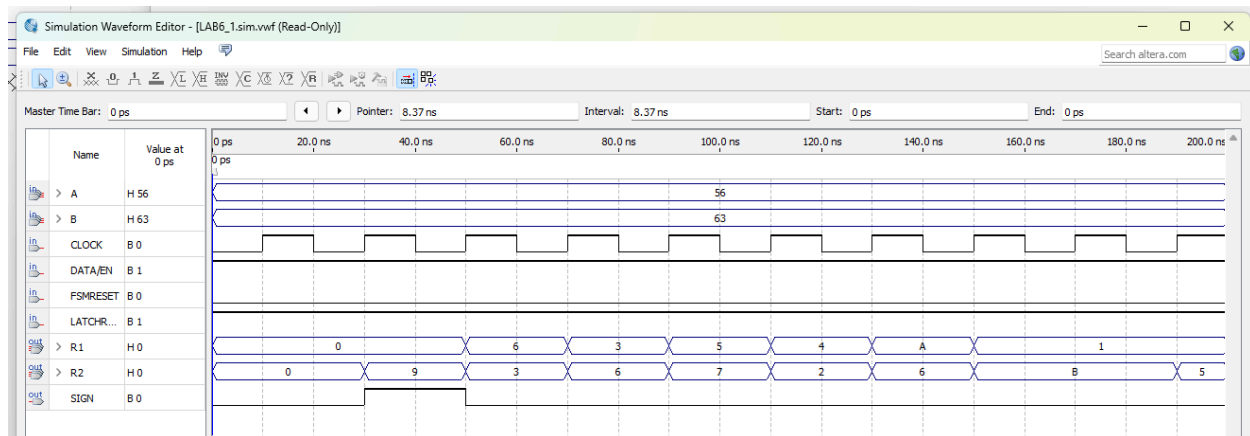
### Problem Set 2

<p>SHR</p> <p>0101 0110 → 0001 0101</p> <p>1 5</p>	<p><math>(A - B) + 4</math></p> <p><math>-13 + 4 = -9</math></p> <p><math>-09</math></p>	<p>Max(A, B)</p> <p>6 3</p>
<p>0101 0110 0110 0011</p> <p>0011 0110</p> <p>3 6</p>	<p>A + 1</p> <p>56</p> <p>+ 1</p> <p>57</p>	<p>A · B</p> <p><math>= 0101\ 0110 \cdot 0110\ 0011</math></p> <p><math>= 01000010</math></p> <p>= 4 2</p>
<p>0101 0110</p> <p>1010 0110</p> <p>A 6</p>	<p>ROL B</p> <p>ROL 0110 0011</p> <p><math>= 0001\ 1011</math></p> <p>1 B</p>	<p>null</p>

**Figure 11:** Hand calculations of the proper output of the design.

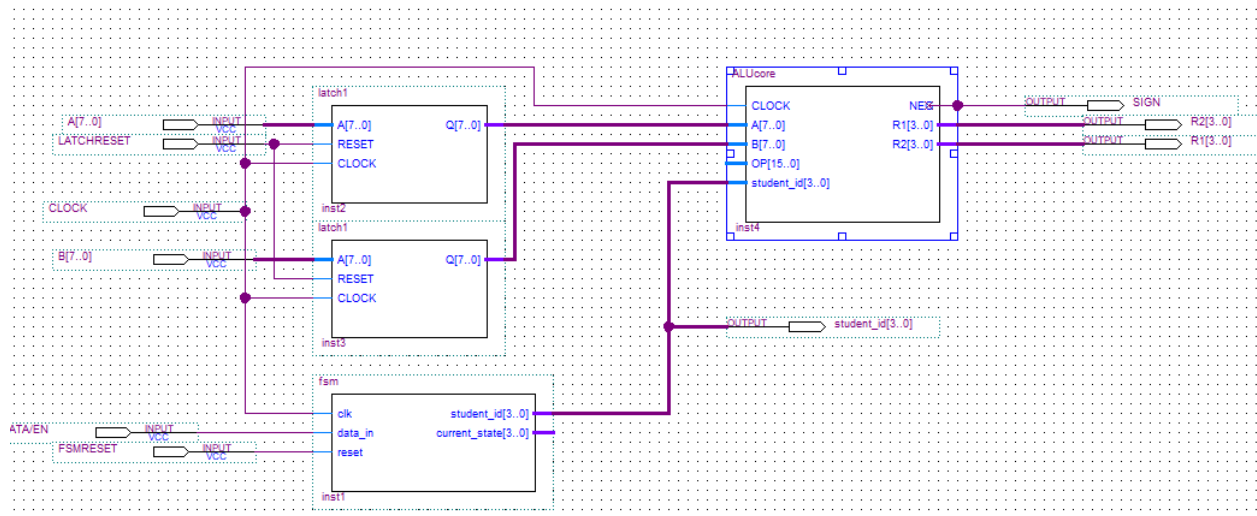
**Table 6:** Microcode generated by ALU modified

Function #	Microcode	Boolean Operation/Function	Expected Output of $A = (56)_{16}$ $B = (63)_{16}$ (in hexadecimal)
1	0000000000000001	SHR 2 bits rights	15
2	0000000000000010	$(A-B) + 4$	-09
3	0000000000000100	Max(A,B)	63
4	0000000000001000	Swap the upper 4 bits of A by the lower 4 bits of B	36
5	00000000000010000	$A + 1$	57
6	000000000000100000	$A \cdot B$	42
7	00000000001000000	Invert the upper four bits of A	A6
8	00000000010000000	ROL(B)	1B
9	00000000100000000	null	U

**Figure 12:** The waveform of problem set 2.

### Part III: The Modified FSM

The purpose of this design was to edit the finite state machine to display a 'y' when the student number had an even parity, and a n when the student number had an odd parity. This was achieved by editing the FSM and alu for the outputs to produce a 0 if 'n' and 1 if 'y' an edited seven segment was also created.



**Figure 13:** The block schematic diagram of problem set 3.

Purpose of each input and output: The inputs and outputs are the exact same as the initial design.

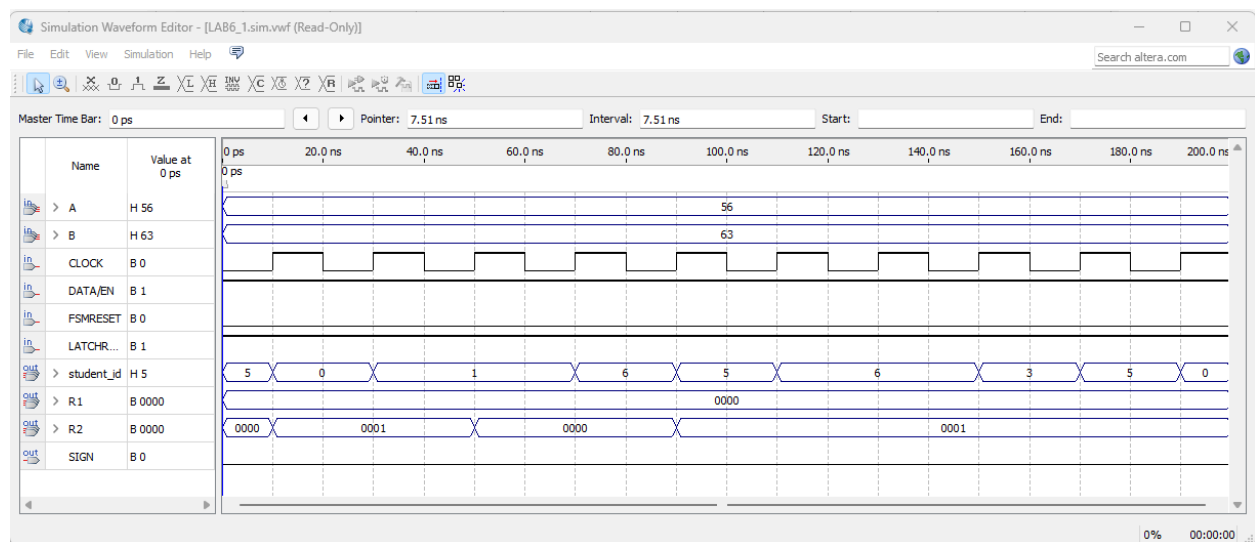
Problem Set 3:

0000	0001	0010	0011	0100	0101	0110
Parity = 0	Parity = 1	Parity = 1	Parity = 2	Parity = 1	Parity = 2	Parity = 2
y	n	n	y	n	y	y
0111	1000	1001				
Parity = 3	Parity = 1	Parity = 2				
n	n	y				

**Table 7:** Student number and Parity



# (in hexadecimal)	Student # (in binary)	Parity	Even (y or n)
0	0000	0	y
1	0001	1	n
2	0010	1	n
3	0011	2	y
4	0100	1	n
5	0101	2	y
6	0110	2	y
7	0111	3	n
8	1000	1	n
9	1001	2	y



**Figure 14:** The waveform of problem set 3.

## Conclusion

In conclusion, a successfully working general processing unit was created in this experiment. This was achieved through making 4 main components and connecting them to work together. In part 1, a simple design was done with a simple finite state machine and ALU with simple functions. In part 2, the ALU was then modified to do more complex functions. Lastly, in part 3 the finite state machine was modified to consider parity and display parity based on if the parity was odd or even. Overall, the experiment was successful and the result was a fully functional general processing unit.