Course Number	COE 328
Course Title	Digital Systems
Semester/Year	Sem3/Fall 2023
Instructor	Reza Sedaghat
TA Name	Luzalen Marcos

Lab Report No.	6

Report Title	Design of a Simple General-Purpose Processor

Section No.	04
Submission Date	2023-12-10
Due Date	2023-12-11

Student Name	Student ID	Signature
Dip Tandel	501171728	Dip
Lab Partner: Muhammad Wajeeh Ul Hassan	501203009	[illegible]

Table of Contents

Introduction

This lab experiment aimed to display both a student's ID number and the results of various boolean functions that were related to the last four digits of that ID. The circuit we used included two latches, an Arithmetic Logic Unit (ALU), a control unit made up of a Finite State Machine (FSM), and a 4:16 decoder.

Latches are parts of digital circuits that function like storage, holding binary data temporarily. The Arithmetic Logic Unit (ALU) then processes these input values and incorporates them into the desired boolean operation.

The control unit, made up of an FSM and a 4:16 decoder, directs the ALU and it decides/outputs the microcode or function to run. The Finite State Machine (FSM) does up-counting as it steps through 0-8 states, one by one, the current states are then put into the 4:16 decoder. The decoder guides the 16-bit output to the ALU.

The Arithmetic Logic Unit, or ALU for short, acts like the brain. It's created to carry out arithmetic and logical actions on binary data and performs important tasks such as adding, subtracting, multiplying, and dividing.

<u>**Components: Latch1, Latch 2, 4:16 Decoder, FSM**</u>

As mentioned in the introduction, the goal was to showcase a student's ID number along with boolean function results related to the last four digits. The circuit employed two D-latches for temporary binary data storage. A control unit, comprising a Mealy Finite State Machine (FSM) and a 4:16 decoder, directs the core processing unit, an Arithmetic Logic Unit (ALU), by cycling through 0-8 states, with the FSM's current state input into the decoder, which, in turn, guides the 16-bit output (microcode) to the ALU. The ALU then executes the arithmetic and logical operations on the binary data.

LATCH 1 AND 2:

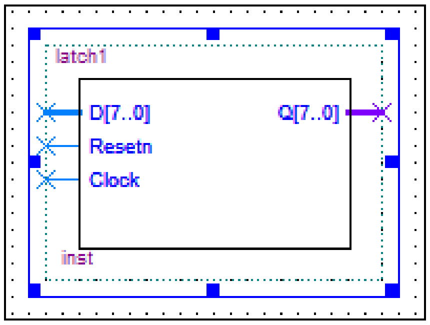


Figure #1: Symbol file of Latch 1/2

Reset	Clock	D	Q(t+1)
1	0	X	Q(t) - output unchanged when 'Clock' = 0
1	1	0	0
1	1	1	1
0	1	1	0 - Resets to 0 when 'Reset' = 0

Table #1: Truth table for Latch 1/2

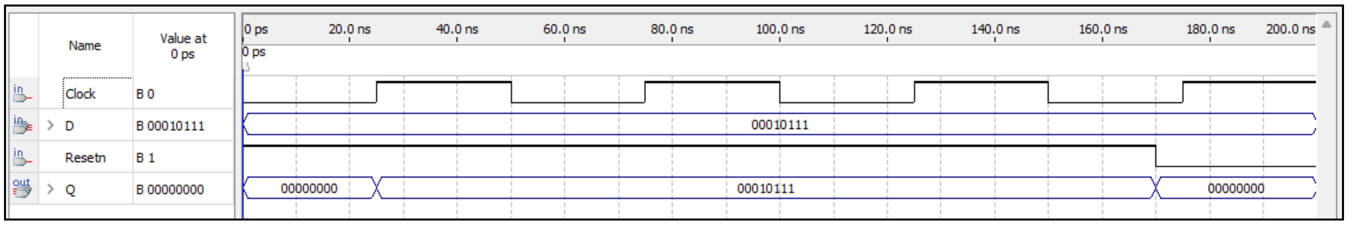


Figure #2: Waveform for Latch 1/2

4:16 DECODER:

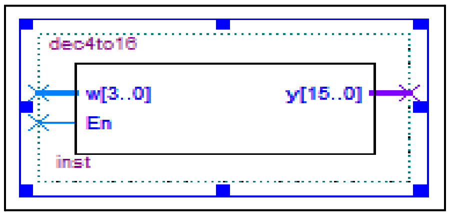


Figure #3: Symbol file of the 4-to-16 decoder

Enabler	W3	W2	W1	W0	Output
1	0	0	0	0	0000000000000001
1	0	0	0	1	0000000000000010
1	0	0	1	0	0000000000000100
1	0	0	1	1	0000000000001000
1	0	1	0	0	0000000000010000
1	0	1	0	1	0000000000100000
1	0	1	1	0	0000000001000000
1	0	1	1	1	0000000010000000

1	1	0	0	0	0000000100000000
1	d	d	d	d	0000000000000001
0	X	X	X	X	0000000000000001

Table #2: Truth table for the 4-to-16 decoder

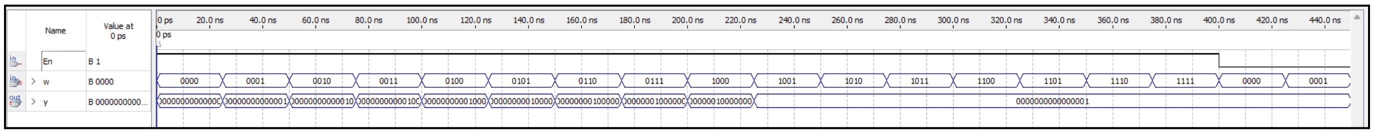


Figure #4: Waveform for the 4-to-16 decoder

FSM:

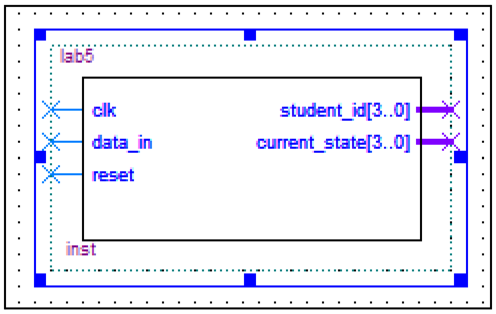


Figure #5: Symbol file of FSM

Present States (Decimal)	Present States y3y2y1y0	Next States		Output z (w = 0)	Output (Decimal, student num)
		w = 0 Y3Y2Y1Y0	w =1 Y3Y2Y1Y0		
0	0000	0000	0001	0101	5
1	0001	0001	0010	0000	0
2	0010	0010	0011	0001	1

3	0011	0011	0100	0010	2
4	0100	0100	0101	0000	0
5	0101	0101	0110	0011	3
6	0110	0110	0111	0000	0
7	0111	0111	1000	0000	0
8	1000	1000	0000	1001	9

Table #3: Truth table for FSM (Mealy, so if w = 1, output goes 1 state ahead)

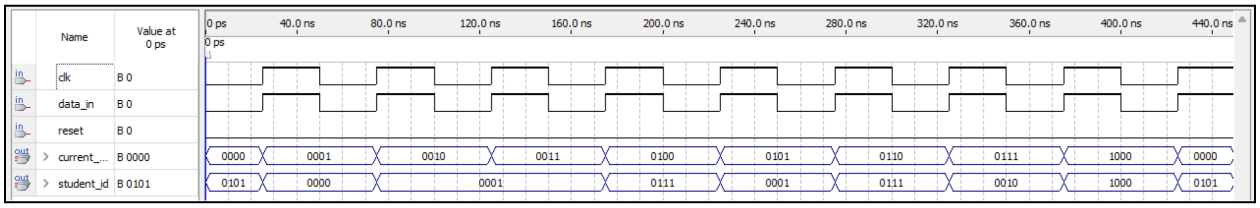


Figure #6: Waveform for FSM

ALU 1 for Problem Set 1

Purpose and Design:

The ALU functions as a part of the GPU (general processor unit), handling mathematical and logical tasks, and outputting the results to a display. In this lab, the ALU utilized four inputs: 'A', 'B', 'Clock', and 'OP' as well as three outputs: 'Neg', 'R1', and 'R2'. The objective was to modify the ALU to execute the given boolean operation between the 8-bit values of A and B, based on the given microcode, and obtain the correct result. In this case, there are nine functions to be done, which means there were some don't-care values in the decoder as shown in Table #2. The decoder allows the ALU to know which operation (microcode) to perform, and the latches allow the ALU to read the values of A and B. In the end, the component 'SSEG' is present to translate the output of the ALU into a form that can be displayed on a 7-segment display.

Purpose of Inputs and Outputs:

'A' and 'B' represent the chosen values to perform the function on, in this case, A and B represent the last 4 digits of my student number (1728, 17 is A, 28 is B). The 'OP' input selects the desired boolean operation and the role of the 'Clock' is to provide a positive edge trigger to activate the outputs of the components. Outputs 'R2' and 'R1' divide the 8-bit boolean operation result into two pieces, each into 4 bits and these values are passed to the SSEG (7-segment) component to be displayed (R2 is displayed first on the left, then R1). When the boolean operation generates a negative number, the 'Neg' output displays a negative sign. Additionally, outside of the ALU, another 7-segment display (SSEG) is connected to the FSM in order to show the student ID.

Function #	Microcode of the Operation	Boolean Operation/Function	Expected SSEG output (R2&R1) when:

			A = 0001 0111 (1 7)
			B = 0010 1000 (2 8)

1	0000000000000001	$sum(A, B)$	3F
2	0000000000000010	$diff(A, B)$	-11
3	0000000000000100	$\overline{A}$	E8
4	0000000000001000	$\overline{A \cdot B}$	FF
5	0000000000010000	$\overline{A + B}$	C0
6	0000000000100000	$A \cdot B$	00
7	0000000001000000	$A \oplus B$	3F
8	0000000010000000	$A + B$	3F
9	0000000100000000	$\overline{A \oplus B}$	C0

Table #4: Microcode for ALU 1

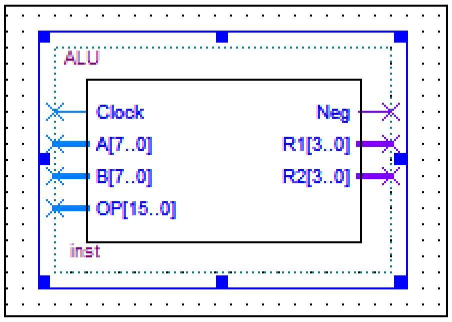


Figure #7: Symbol file of ALU 1

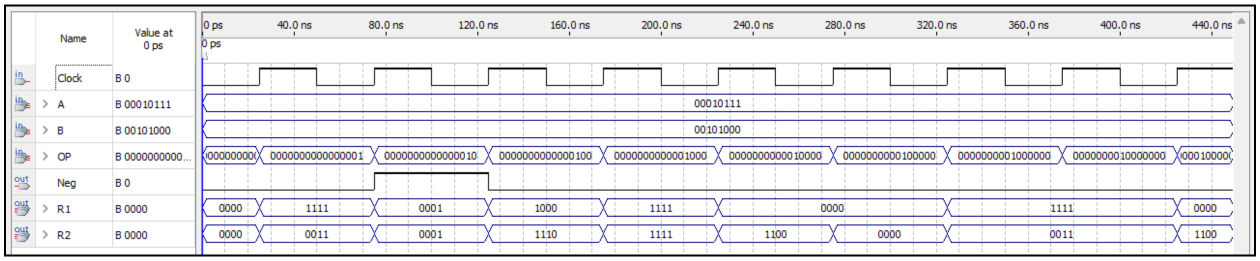


Figure #8: Waveform for ALU 1

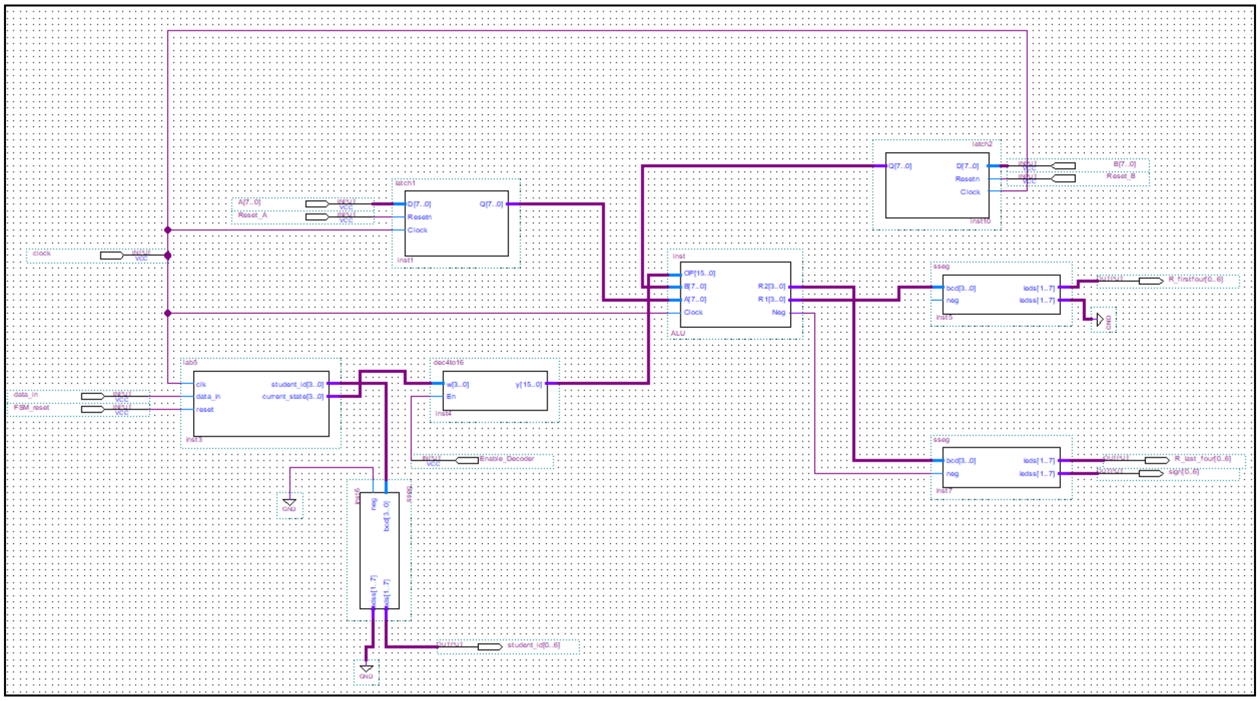


Figure #9: Block Diagram for problem set 1

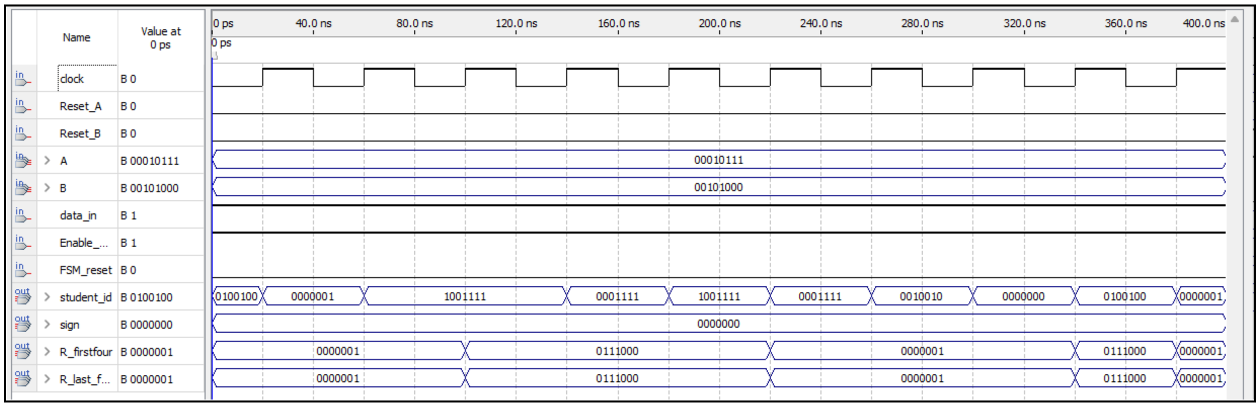
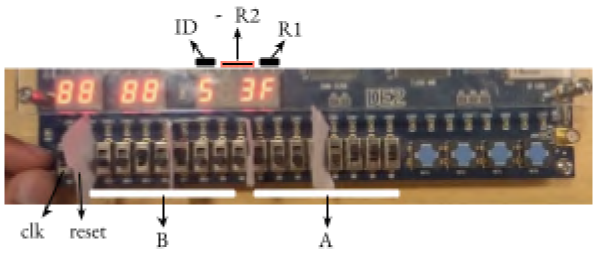
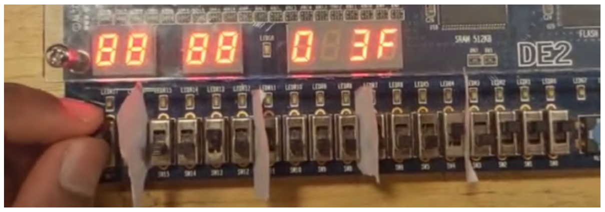
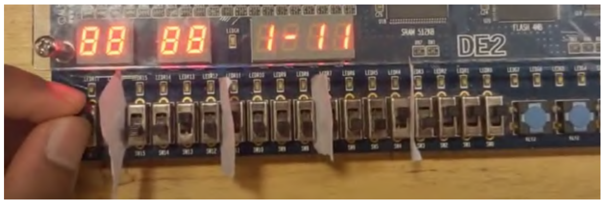
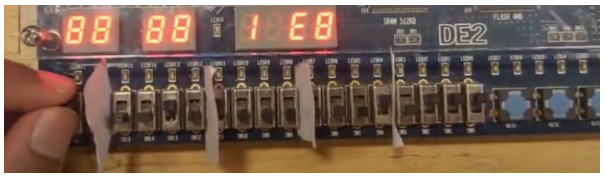
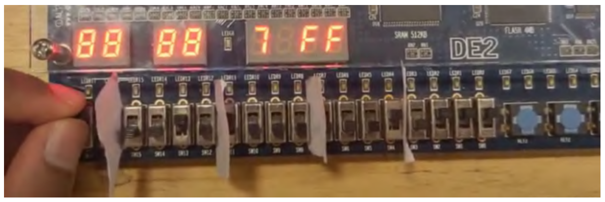


Figure #10: Waveform of Block Diagram for problem set 1 (outputs show the SSEG output)

Boolean Operation/Function	SSEG output (R2&R1) when: A = 0001 0111 (1 7) B = 0010 1000 (2 8)
sum(*A*, *B*)	
diff(*A*, *B*)	
$\overline{A}$	
$\overline{A \cdot B}$	
$\overline{A + B}$	

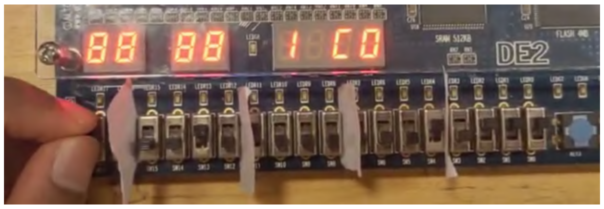
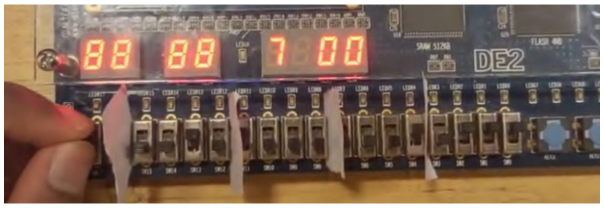
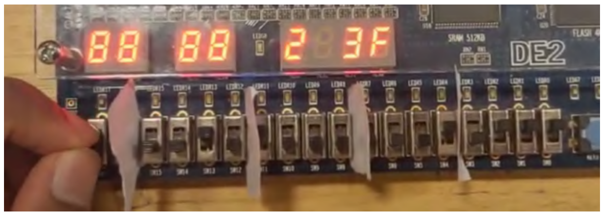
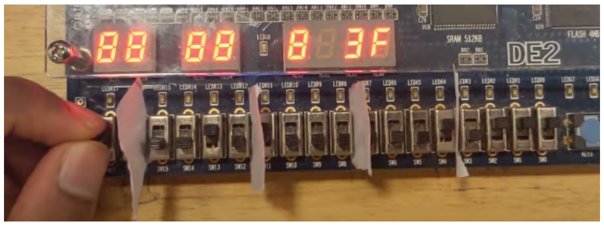
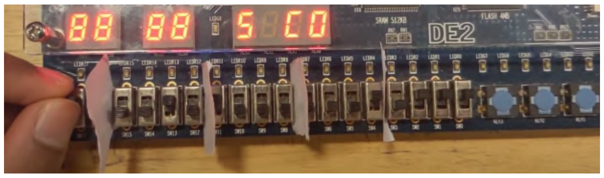
$A \cdot B$	
$A \oplus B$	
$A + B$	
$\overline{A \oplus B}$	 

Table #5: In-lab display results for problem set 1

Note that the student number is always one state ahead, or the output is remaining one state behind, a possible reason for this may be due to the fact that every time it is clocked, w is equal to 1, which means the state is always jumping ahead, rather than remaining the same as it should going from function 1 to 2. This issue is also present in ALU 2.

ALU 2 for Problem Set 2 (assigned 'd')

Purpose and Design:

The purpose of the ALU in this problem set is the same as problem set 1. The ALU functions as a part of the GPU (general processor unit), handling boolean operations, and outputting the results to a display. The given boolean operations are executed between the 8-bit values of A and B. In this case, the operations are simply slightly different.

Purpose of Inputs and Outputs:

The purpose of the inputs and outputs of ALU 2 are unchanged from the ALU of problem set 1, which is on page 7.

Function #	Microcode of the Operation	Boolean Operation/Function	Expected SSEG output (R2&R1) when: A = 0001 0111 (1 7) B = 0010 1000 (2 8)
1	0000000000000001	Shift A to right by two bits, input bit = 1 (SHR)	C5
2	0000000000000010	Produce the difference of A and B and then increment by 4	-0D
3	0000000000000100	Find the greater value of A and B and produce the	28

		results (Max(A,B))	
4	0000000000001000	Swap the upper 4 bits of A by the lower 4 bits of B	17
5	0000000000010000	Increment A by 1	18
6	0000000000100000	Produce the result of ANDing A and B	00
7	0000000001000000	Invert the upper four bits of A	87
8	0000000010000000	Rotate B to left by 3 bits (ROL)	41
9	0000000100000000	Show null on the output	00

Table #6: Microcode for ALU 2 (assigned 'd') [1]

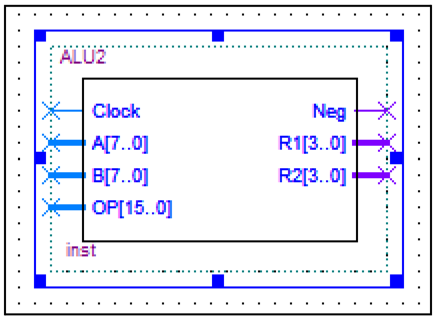


Figure #11: Symbol file of ALU 2

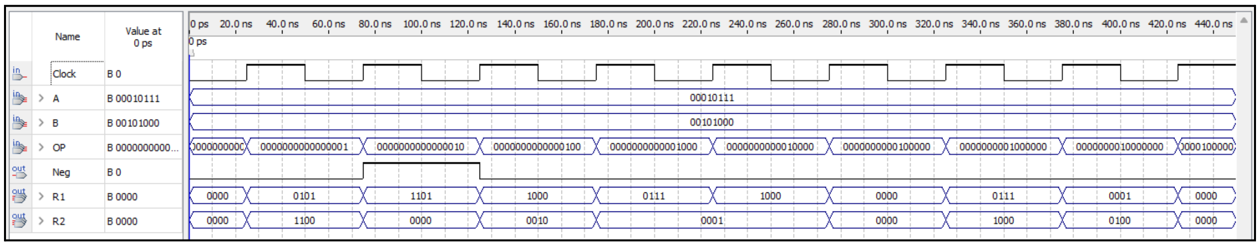

Name
Value at 0 ps
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
180.0 ns
200.0 ns
220.0 ns
240.0 ns
260.0 ns
280.0 ns
300.0 ns
320.0 ns
340.0 ns
360.0 ns
380.0 ns
400.0 ns
420.0 ns
440.0 ns
Clock B 0
A B 00010111 00010111
B B 00101000 00101000
OP B 0000000000... 000000000 0000000000000001 0000000000000010 0000000000000100 0000000000001000 0000000000010000 0000000000100000 0000000001000000 0000000010000000 000100000
Neg B 0
R1 B 0000 0000 0101 1101 1000 0111 1000 0000 0111 0001 0000
R2 B 0000 0000 1100 0000 0010 0001 0000 1000 0100 0000

Figure #12: Waveform for ALU 2

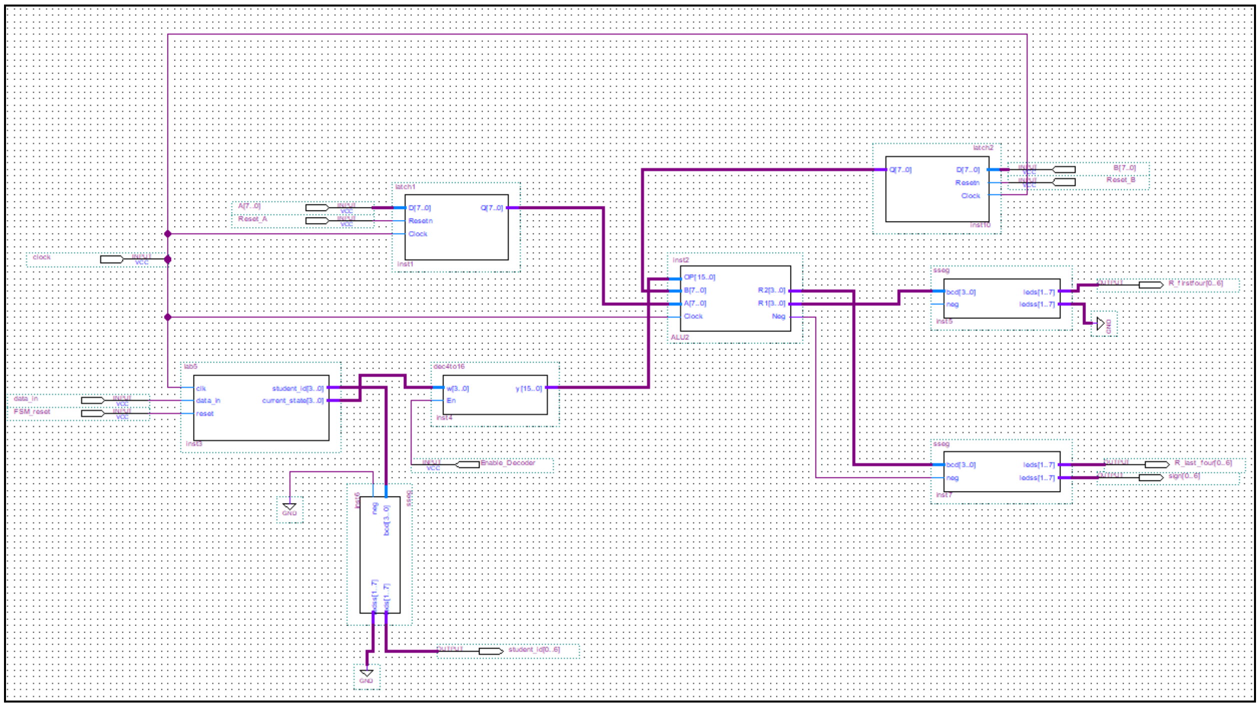


Figure #13: Block Diagram for problem set 2

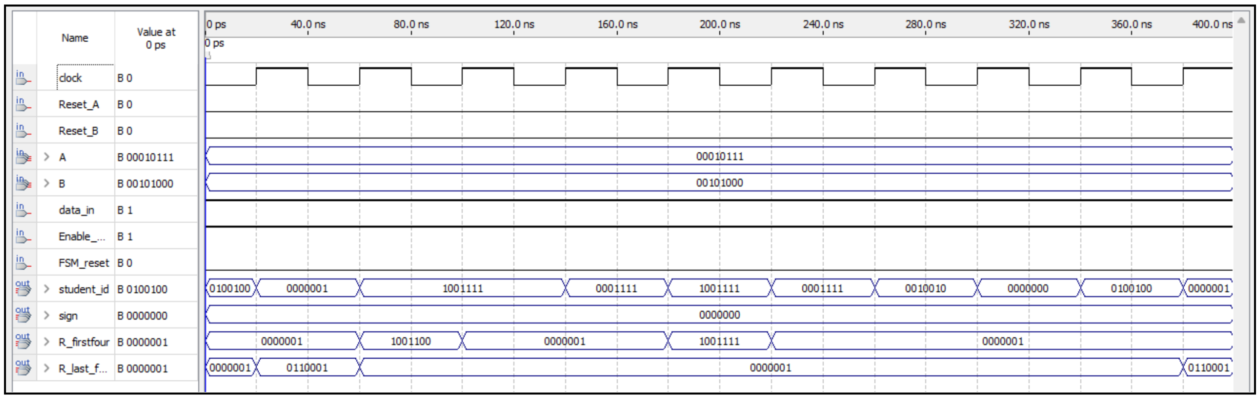


Figure #14: Waveform of Block Diagram for problem set 2 (outputs show the SSEG output)

Boolean Operation/Function	SSEG output (R2&R1) when:
	A = 0001 0111 (1 7)
	B = 0010 1000 (2 8)

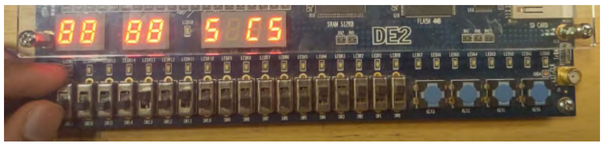
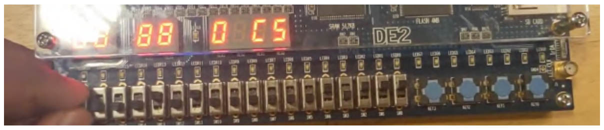
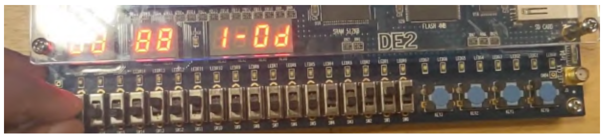
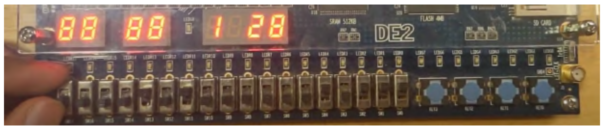
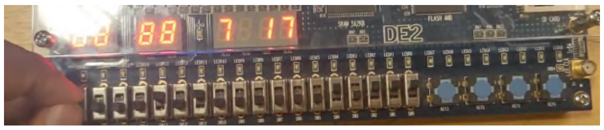
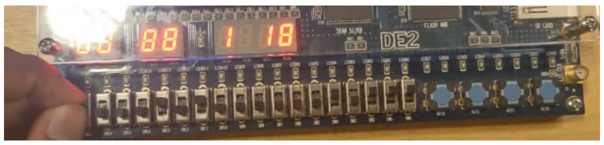
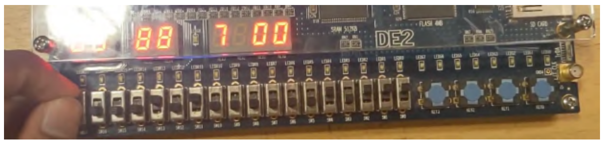
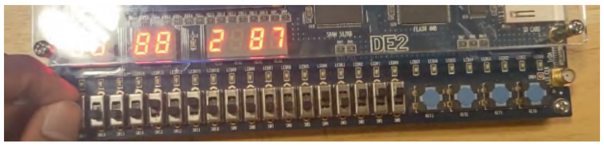
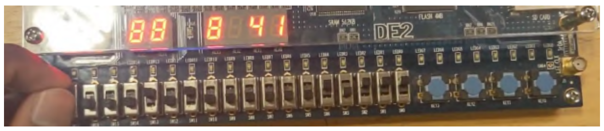
Shift A to right by two bits, input bit = 1 (SHR)	
Produce the difference of A and B and then increment by 4	
Find the greater value of A and B and produce the results (Max(A,B))	
Swap the upper 4 bits of A by the lower 4 bits of B	
Increment A by 1	
Produce the result of ANDing A and B	
Invert the upper four bits of A	
Rotate B to left by 3 bits (ROL)	
Show null on the output	

Table #7: In-lab display results for problem set 2

ALU 3 for Problem Set 3 (assigned d)

Purpose and Design:

The ALU component, in this case, was to be modified so it checks each digit of the student ID, going though the digits with a positive edge triggered clock, to see if it is in even parity or not. So if the first digit is '5' for instance, it would be 0101 in binary, which is already even, therefore the result should display a 'y' (no parity bit needed to be added). To achieve this, the input 'student_id' was added to the ASU, which is connected to the FSM, as mentioned before the FSM outputs each digit of my student ID one by one. The ASU is now coded in VHDL to read and determine if the digit exhibits even parity, and now there is only one output needed, 'student_ouput', to represent this data. This data was translated into a form that can be displayed on a 7-segment display, inside a new SSEG component simply named SSEG2. A new SSEG component was required since the previous one could only display from 0 to 10, or A to F, but in this case, specifically 'y' and 'n' were to be displayed.

Purpose of Inputs and Outputs:

Although 'A', 'B', and 'OP' are present, they are redundant and do not serve any purpose. 'Clock' as expected is present to provide a positive edge trigger to activate the components. The input 'student_id' is connected to the output of the FSM which is present so that the ALU is able to read each digit of the student ID (4-bit) coming from the FSM and decide if it is even or not. The output 'student_output' outputs either 0001 or 0000. 0001 represents a 'y' and 0000 represents an 'n', this was translated inside SSEG2 and displayed as either 'y' or 'n'. Similar to ALU 1 and 2, outside of the ALU, another 7-segment display (SSEG) is connected to the FSM in order to show the student ID.

Student #	Expected SSEG output
5	y
0	y
1	n
1	n
7	7
1	1
7	7
2	2
8	8

Table #8: Microcode for ALU 3 (assigned 'd') [1]

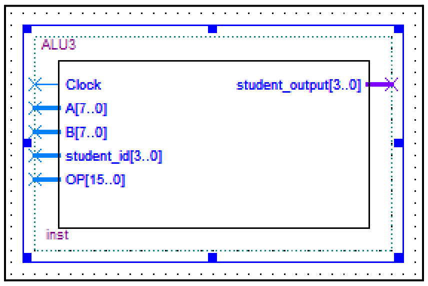


Figure #15: Symbol file of ALU 3

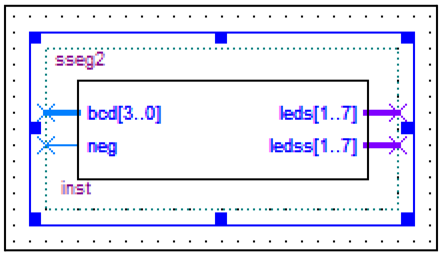


Figure #16: Symbol file of SSEG2

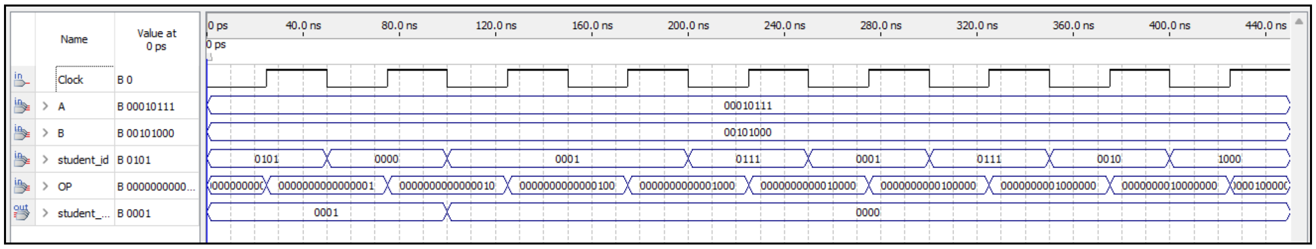


Figure #17: Waveform for ALU 3

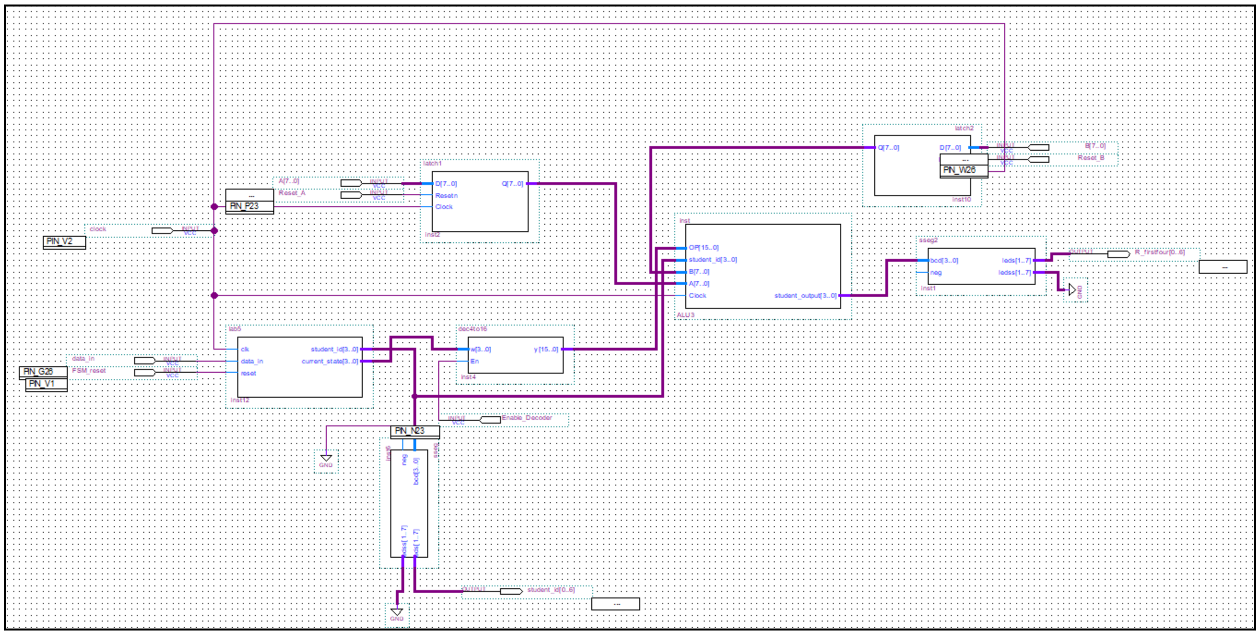


Figure #18: Block Diagram for problem set 3

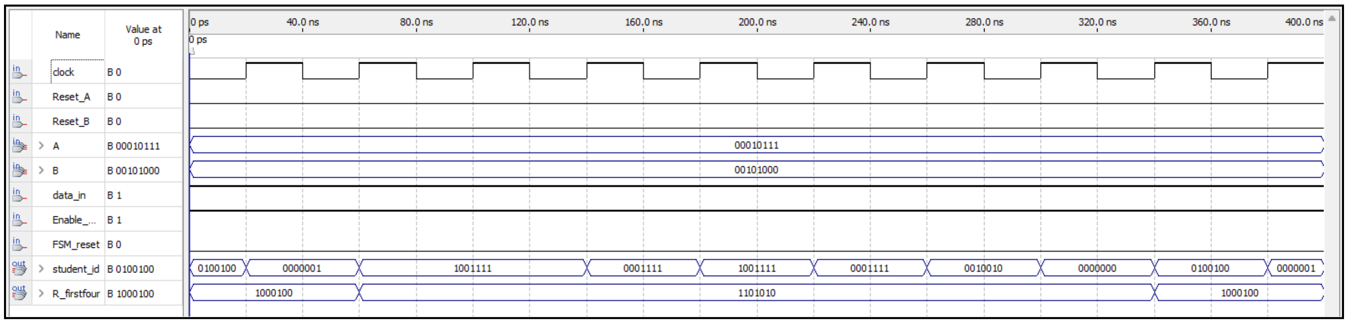
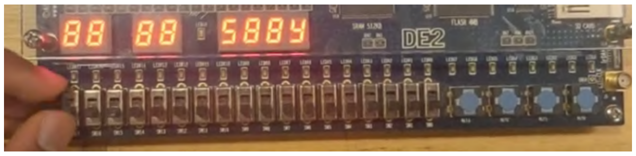
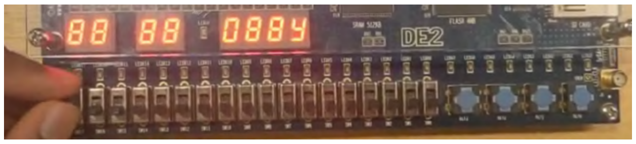
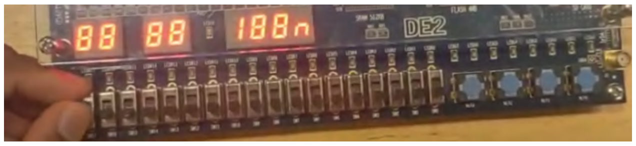
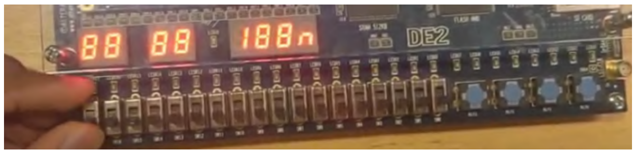
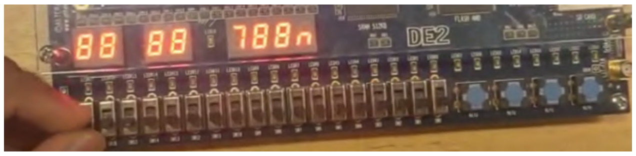
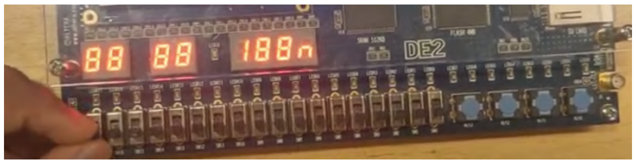
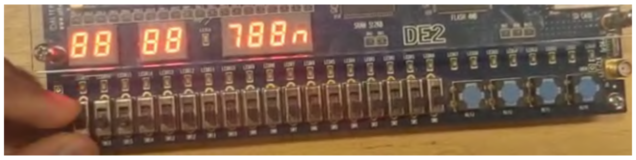


Figure #19: Waveform of Block Diagram for problem set 3 (outputs show the SSEG output)

In-lab the display seemed to be flipped in this case, so whereas originally 0111011 represented 'y' in a 7-segment display, this had to be flipped to 1000100 to work.

Function #	Microcode of the Operation	SSEG output when: A = 0001 0111 (1 7) B = 0010 1000 (2 8)
1	0000000000000001	
2	0000000000000010	
3	0000000000000100	
4	0000000000001000	
5	0000000000010000	
6	0000000000100000	
7	0000000001000000	

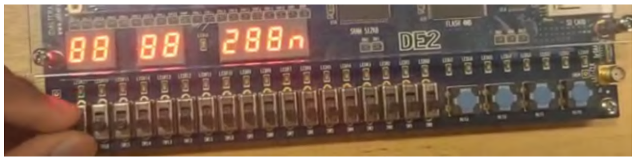
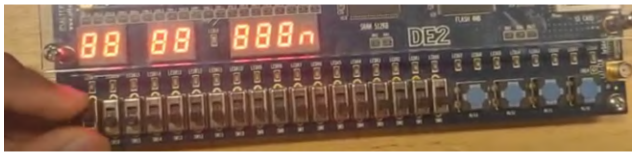
8	0000000010000000	
9	0000000100000000	

Table #9: In-lab display results for problem set 3

<u>Conclusion</u>

Overall, aside from the small hiccup noted under Table #5, lab 6 was successfully completed and with it, we learned how to design a simple general-purpose processor. This milestone requires a good understanding of the knowledge in previous labs and the components we learned about in this course, such as latches, finite-state machines, and decoders. Furthermore, this highlights how we were able to apply our knowledge and combine what we know to create something larger like this processor. We have also gained more experience with Quartus II and learned about many of its features such as waveforms and block diagrams, and how it can be used to verify certain results or how it can be used to combine components and thus carry out much larger functions. We have also gained much more expertise in VHDL coding, which is essential for future success.

References

[1] Department of Electrical and Computer Engineering, “Lab 6: Design of a Simple General-Purpose Processor ,” 2023. https://www.ecb.torontomu.ca/~courses/coe328/Lab6_Manual.pdf