Design of a Simple Central Processing Unit

Hilal Nasir

Dec 2024

Table of Contents

Introduction

This lab report includes the various topics and methods taught in the course to create a simple central processing unit (CPU) in VHDL and then applied to an Field Programmable Gate Arrays Board. This CPU allows for simple calculations to be made by executing instructions inputted. The components within the lab include latches, finite state machines (FSM), a 3:8 decoder, and arithmetic logic units (ALU). The latches allow for input data to be stored and then sent through the rest of the components. The FSM creates a sequence pattern and cycles through states. This state is received by the 3:8 decoder and translates to one of the 8 opcodes. These two components make up the control unit. The ALU performs arithmetic operations with the input and opcode which results in the output displayed by the 7 segment display. The final block diagram of the simple CPU can be seen in Figure 1.

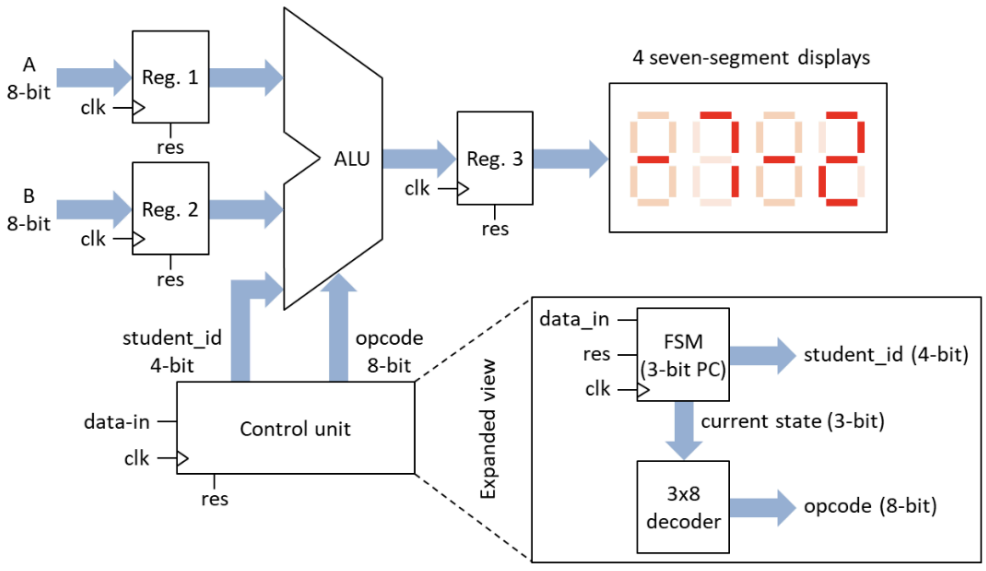

Figure 1: Simple CPU Block Diagram [1].

Components

Latches

The latches allow for input values to be stored. With every rising edge of the clock the latch reads its input value and updates its stored value. Latches are usually level-triggered, however in this case the latches take in the clock's trigger. The input of the latches are both A and B values from the student ID and meant two latches would be needed. Figure 2 is the block diagram of both latches used since the actual logic did not change between them but rather their input. Figure 3 & 4 is the waveform output of latch 1 and latch 2 respectively.

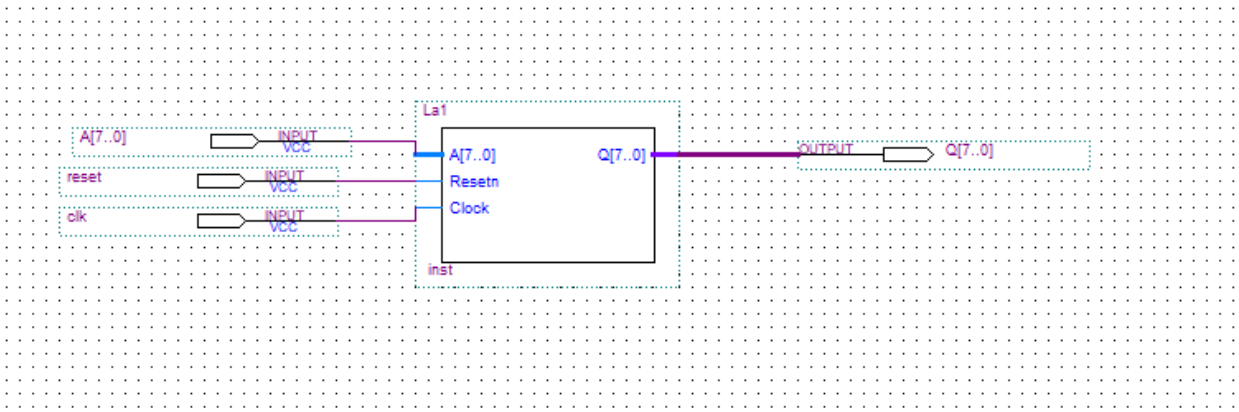

Figure 2: Block diagram of Latches

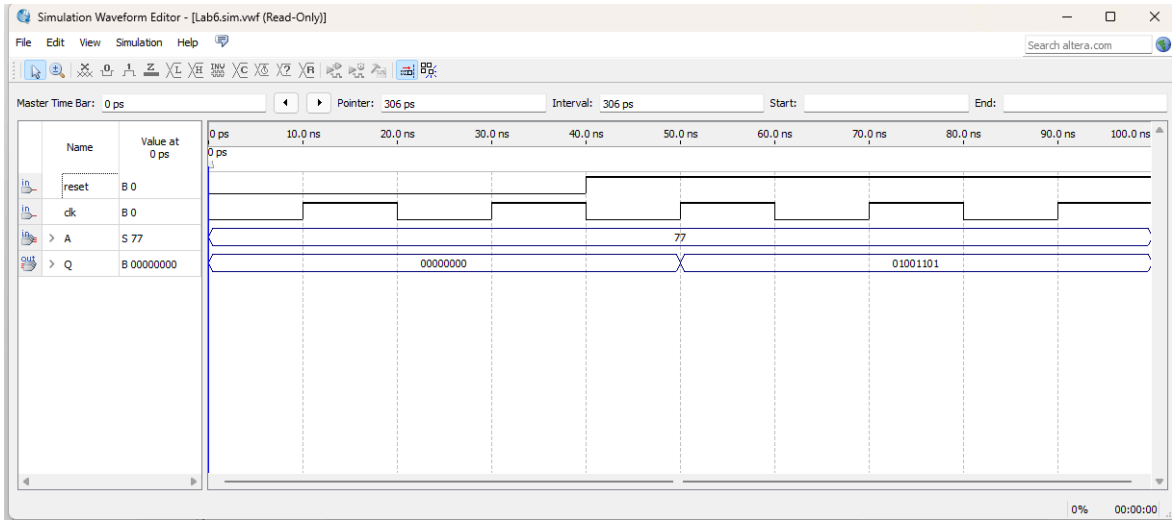

Figure 3: Waveform of Latch 1

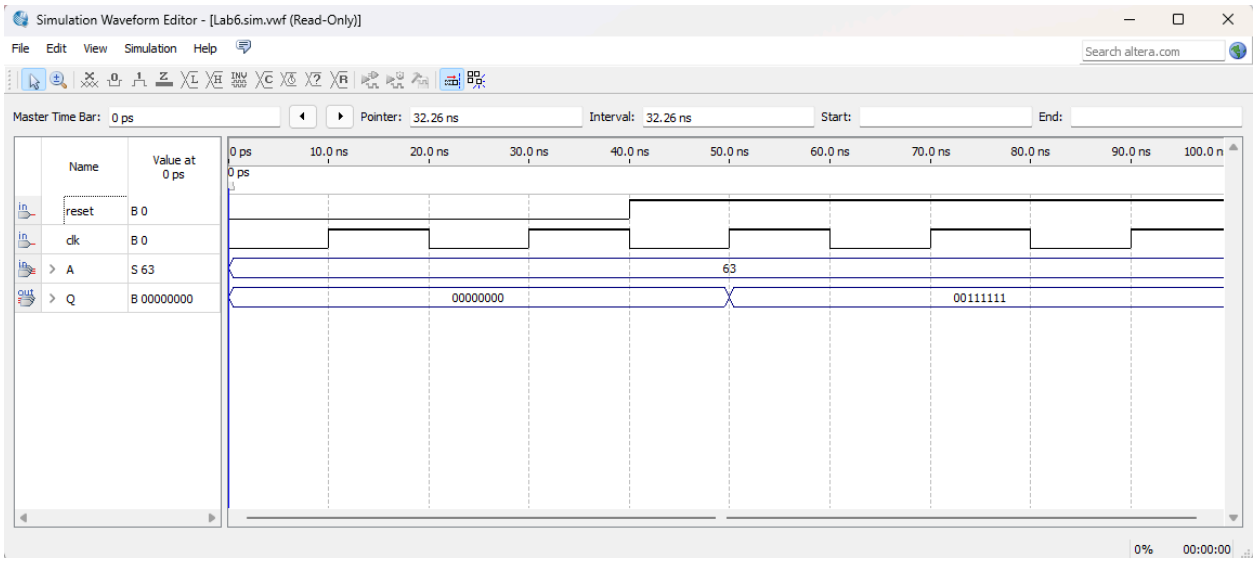

Figure 3: Waveform of Latch 2

D	CLK	Q	Q'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Table 1: Truth Table for the Latches

Finite State Machine (FSM)

The FSM allows for a controlled output activation based on previous states. From the clock input the FSM cycles through 8 states which allows for certain outputs to be active in certain states. The block diagram and waveform can be seen below.

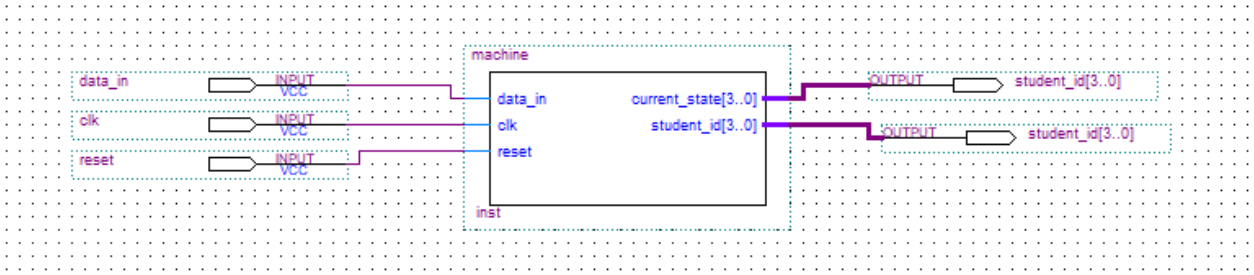

Figure 4: Block Diagram for FSM

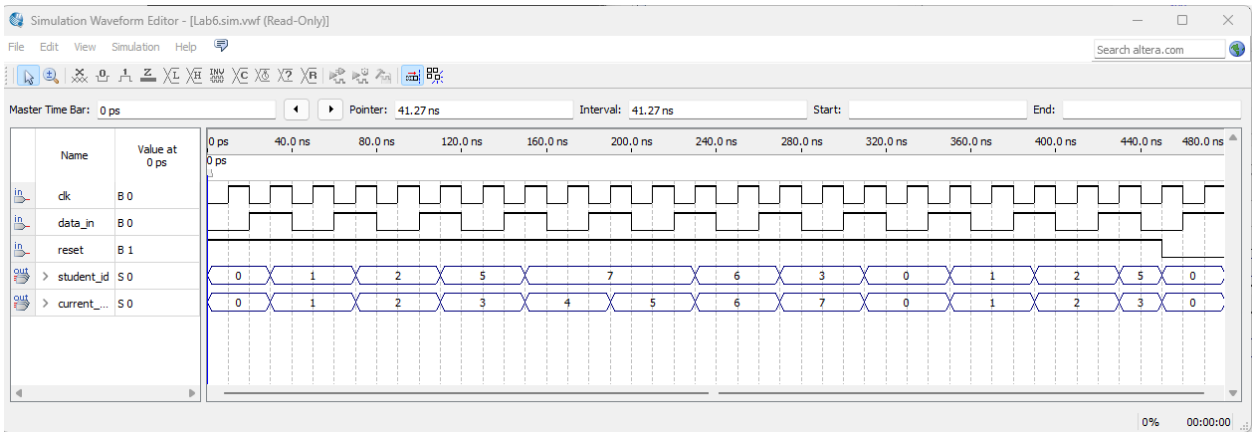

Figure 5: Waveform for FSM

Current State	Reset (R)	data_in	Next State	Active Output (D)
s0/d2	0	1	s1/d3	d2
s1/d3	0	1	s3/d5	d3
s2/d4	0	1	s5/d7	d4
s3/d5	0	1	s7/d9	d5
s4/d6	0	1	s3/d5	d6
s5/d7	0	1	s7/d9	d7
s6/d8	0	1	s5/d7	d8
s7/d9	0	1	s1/d3	d9

Table 2: Truth table for FSM

3:8 Decoder

Based on the 3 bit binary input, converted from the current state from FSM, the decoder will select 1 of 8 outputs, in this case opcodes. It essentially decodes the 3 bit input into 1 output line that is active one at a time. The 3 bit address accesses an output stored in a specific memory location based on that address. Below are the block diagrams, waveform outputs and truth tables for the decoder used.

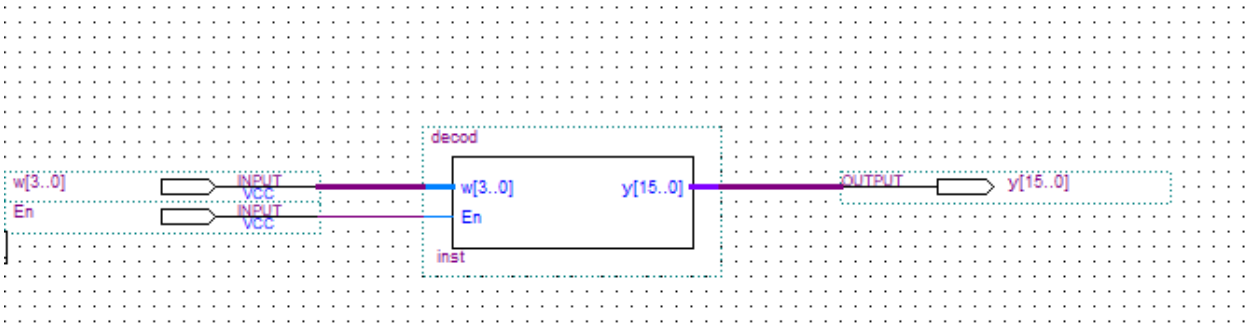

Figure 7: Block Diagram for 3:8 Decoder

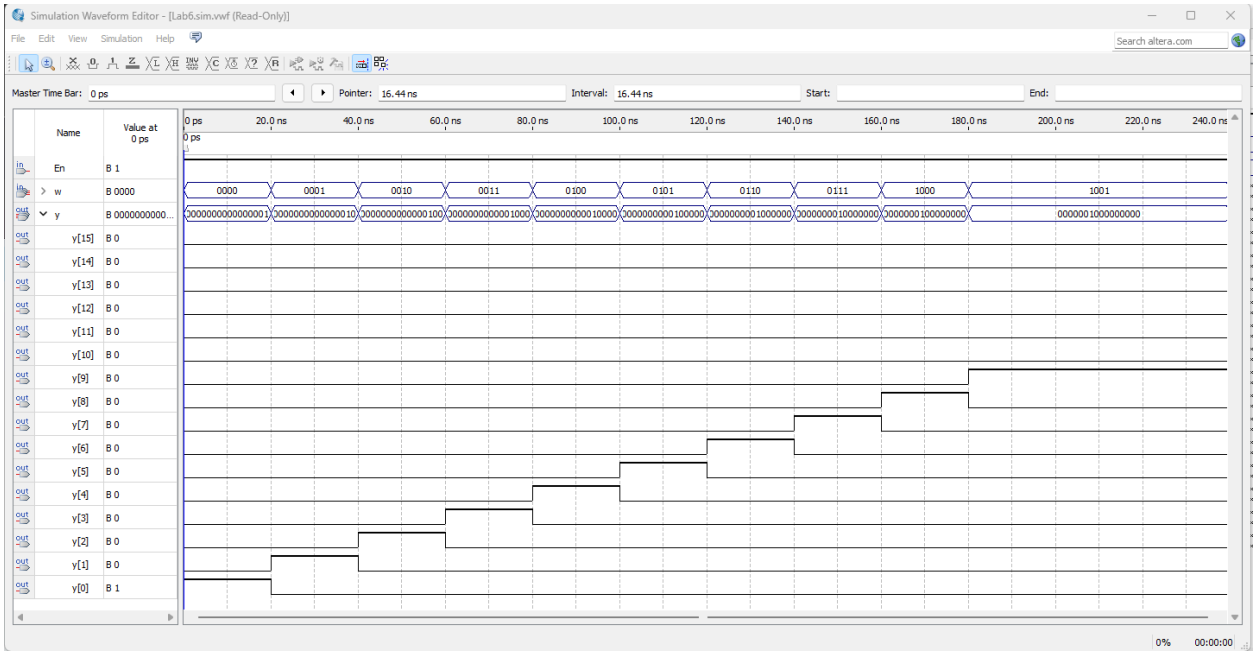

Figure 8: Waveform for 3:8 Decoder

x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Table 3: Truth Table for 3:8 Decoder

Problem 1 Arithmetic logic unit (ALU)

The ALU performs arithmetic and logical operations based on the 8-bit opcode input. The inputs include A and B, the opcode and the clock and outputs R1 and R2 as seen in Figure 9. The opcode selects which operation to perform on A and B and the result is both R1 and R2 which are the two numbers separately. This output is displayed on the 7-segment display. The clock input is needed so the ALU is synchronized with the control unit.

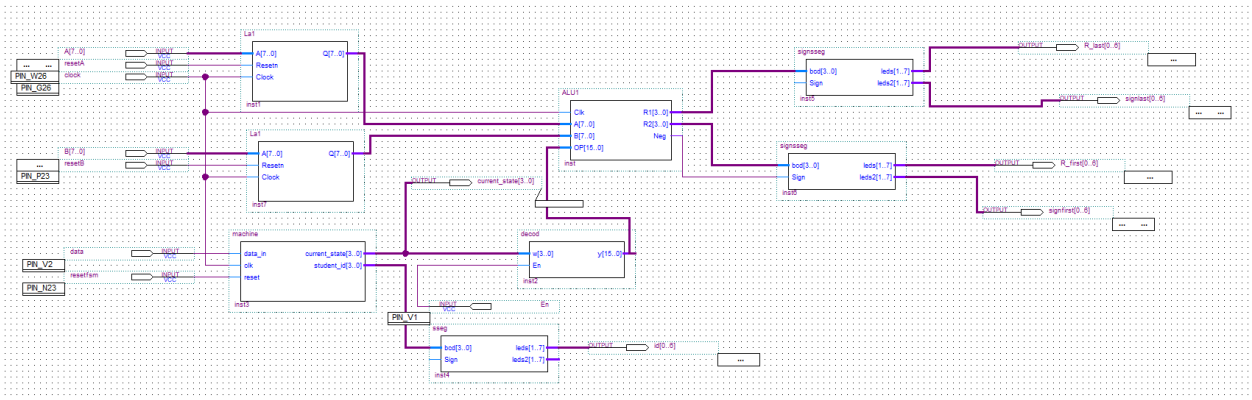

Figure 9: Complete Schematic design of Problem 1

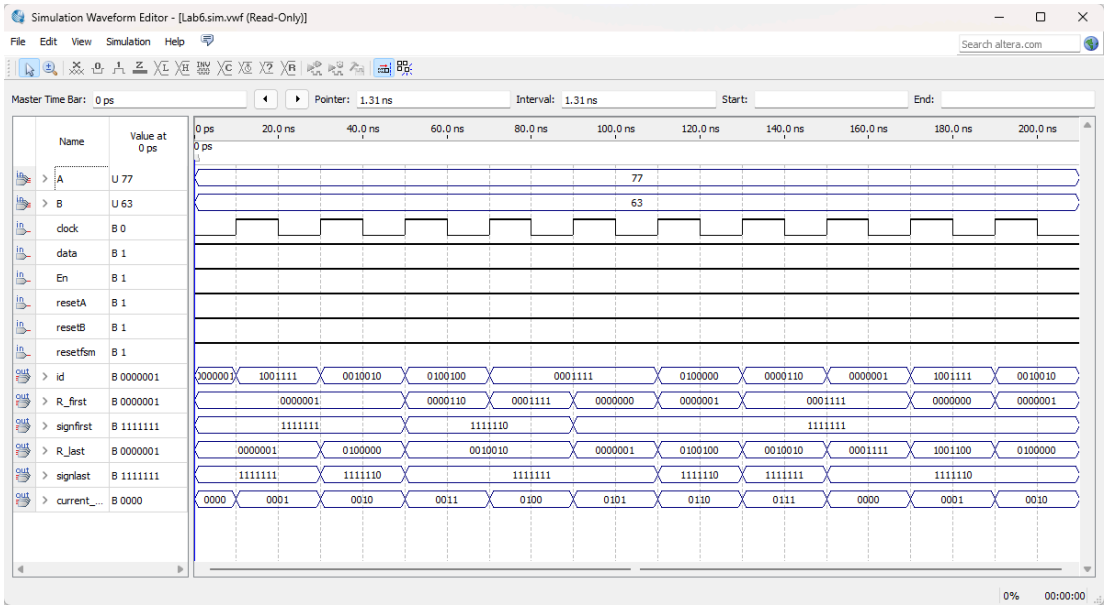

Figure 10: Waveform file using FSM Student ID

Function #	**Opcode**	**Function**	**Output**
1	00000001	sum(A,B)	01111111
2	00000010	dif(A,B)	010001100
3	00000100	$\overline{A}$	00001110
4	00001000	$A \cdot \overline{B}$	10110010
5	00010000	$A + \overline{B}$	00110010
6	00100000	$A \cdot B$	10111111
7	01000000	$A \oplus B$	01110010
8	10000000	A + B	01111111

Table 4: Opcodes and Output generated by ALU1

Problem 2 (ALU)

For problem two not much was changed for the rest of the circuit other than the ALU which was modified to allow for different operations to be done based on Table 5. The output negative was also added in case the output led to a negative number.

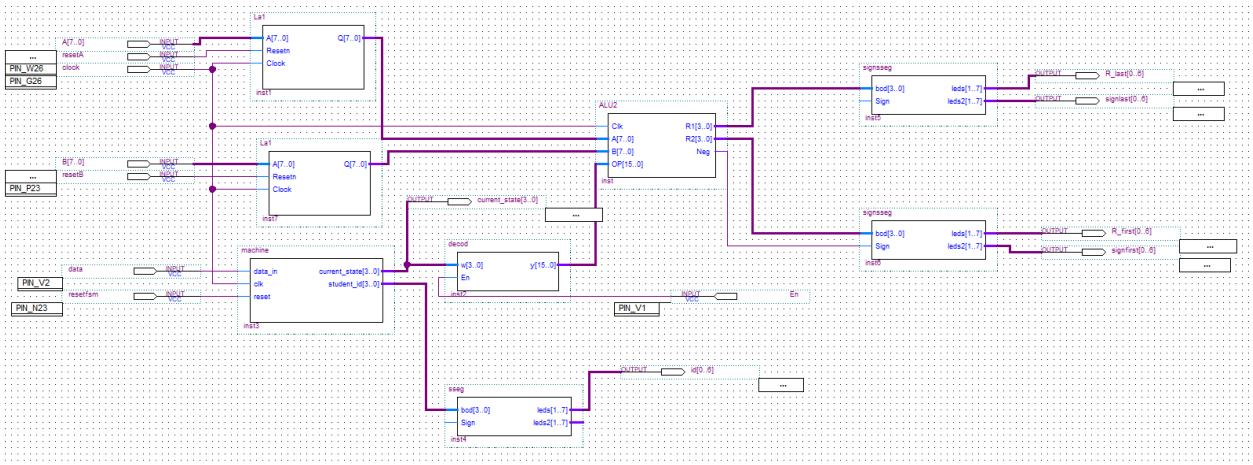

Figure 11: Complete Schematic design of Problem 1

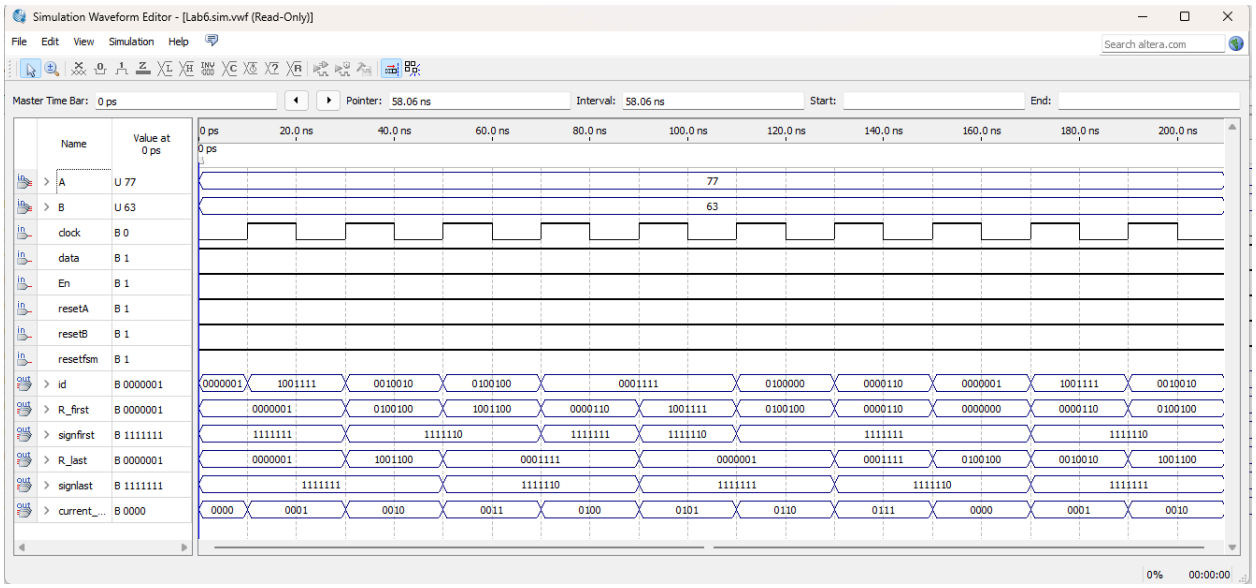

Figure 12: ALU2 Waveform file using FSM Student ID

Function #	Operation / Function	Output
1	Invert the bit-significance order of A	10110010
2	Shift A to left by 4 bits, input bit = 1 (SHL)	11010100
3	Invert upper four bits of B	11001111
4	Find the smaller value of A and B and produce the results (Min(A,B))	00111111
5	Calculate the summation of A and B and increase it by 4	10010000
6	Increment A by 3	01010000
7	Replace the even bits of A with even bits of B	00111111
8	Produce the result of XNORing A and B	10001101

Table 5: Operations and Output generated by ALU2

Problem 3 (ALU)

Problem three required the same steps as before however only one operation was performed in the ALU. This logic operation was to check if the binary of the student number had an odd parity. If so, output ‘y’ on the 7-segment display and if not out ‘n’. Again, the inputs were the same but the output was now one digit, R, which was inputted into a new 7-segment code to display the letters.

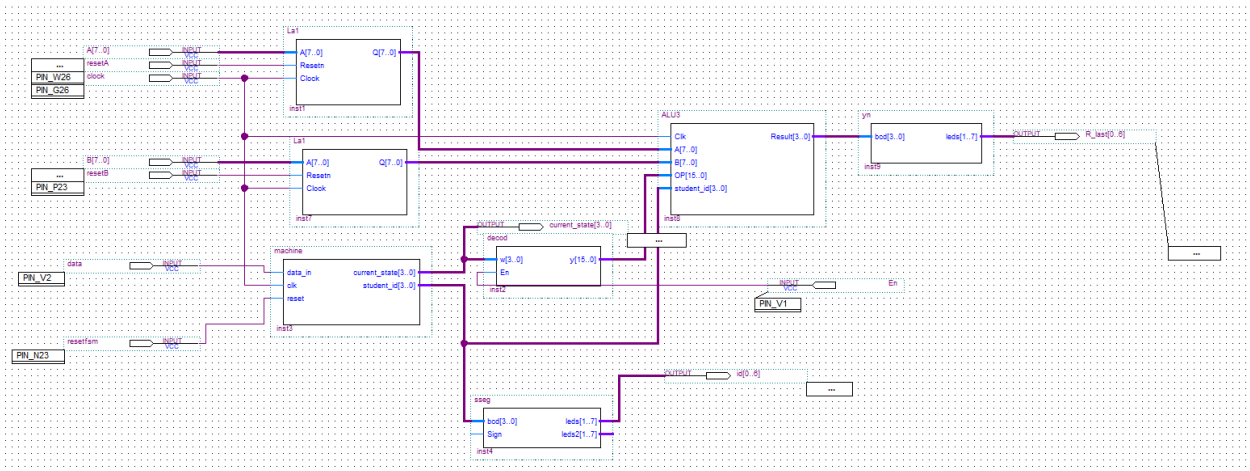

Figure 13: Complete Schematic design of Problem 3

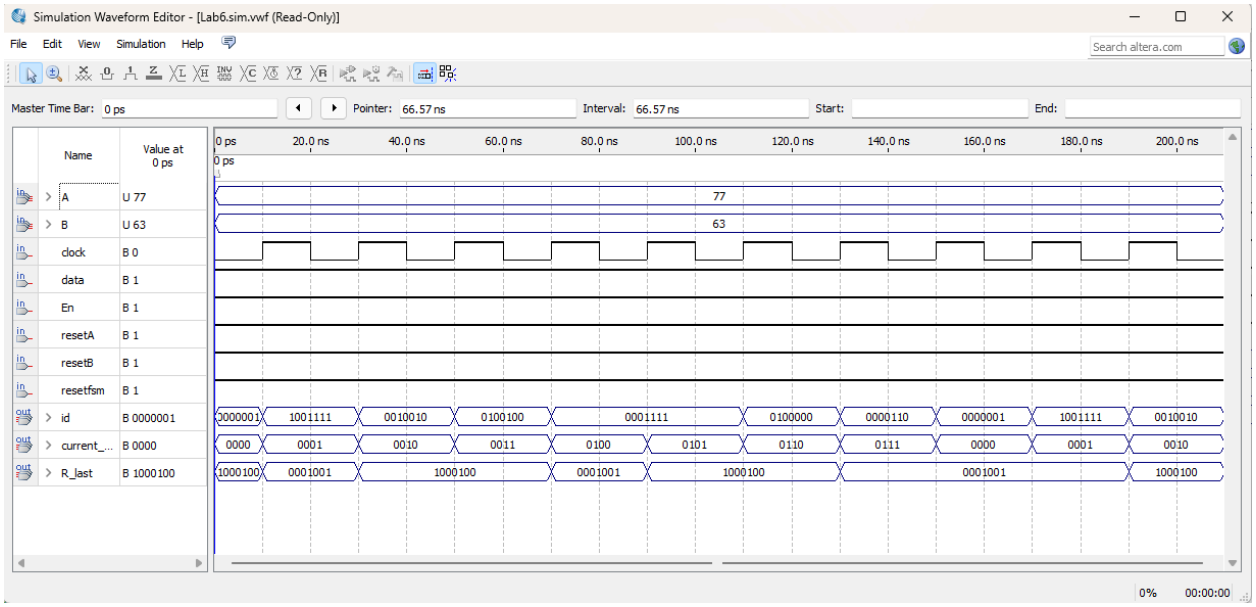

Figure 14: Waveform output of ALU 3

Input (Student #)	Input (Binary)	Output
0	0000	n
1	0001	y
2	0010	y
5	0101	n
7	0111	y
7	0111	y
6	0110	n
3	0011	n

Table 6: Output generated by ALU3 Based on Odd Parity

Conclusion

To conclude, this lab allowed for a better understanding of ALUs and other major components of a CPU. The arithmetic and logical operations done by the ALU and controlled by the FSM and decoder gave insight into computational tasks done by computers in the real world. And the latches helped with learning how memory is stored within such computers. All methods taught in the course were wrapped into this lab making it a great way to connect theoretical concepts and practical applications.