Department of Engineering

Course Number	COE328
Course Title	Digital Systems
Semester/Year	Fall 2023
Instructor	Arghavan Asad
Section No.	10
Group No.	XX
Submission Date	2023/12/04
Due Date	2023/12/04

Lab/Tut Assignment NO.	6

Assignment Title	**Design of a Simple General-Purpose Processor**

Name	Student ID	Signature*
Maheen Ashraf Qureshi	[redacted]	[redacted]

(Note: remove the first 4 digits from your student ID)

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 record under the Student Code of Academic Conduct, which can be found online at: *www.ryerson.ca/senate/current/pol60.pdf*

TABLE OF CONTENTS

2.0 Introduction

Within the design of a GPU, it is important to note that there are many components involved to make it work. Components in the design include using two Latches listed as Latch1 and Latch2 that function as storage units, a 4:16 Decoder and Finite State Machine that form a Control unit that outputs the Student ID and 16-bit ALU Operation Selection. Additionally, it has an Arithmetic Logic Unit that takes the outputs of the latches and clock to output the result in a Seven Segment Display Unit. This lab provides the listed components and their functions using Quartus 13.0 software, and utilizes an Altera board to create the final GPU.

3.0 Components

This section will focus on the storage units and the control unit of the GPU. Specifically, it will elaborate on Latch1, Latch2, the 4x16 Decoder and the Function State Machine.

3.1 Latch 1

The primary function of Latch1 is to hold the data given so that it may be used in the Arithmetic Logic Unit. In this case, Latch 1 holds the 8-bit input value of A, which in this case is equivalent to $(01110101)_2$, alongside the Clock. The design of this specific latch is made up of flip flops, and the output for A is the 8-bit value.

CLOCK	**RESET**	**A[0..7]**	**Q[0..7]**
0	1	0	0
1	0	0	0
0	0	1	0

1	0	1	0
0	0	0	0
1	0	0	0
0	0	1	0
1	0	1	1

Table 3.1.0 Latch1 Truth Table

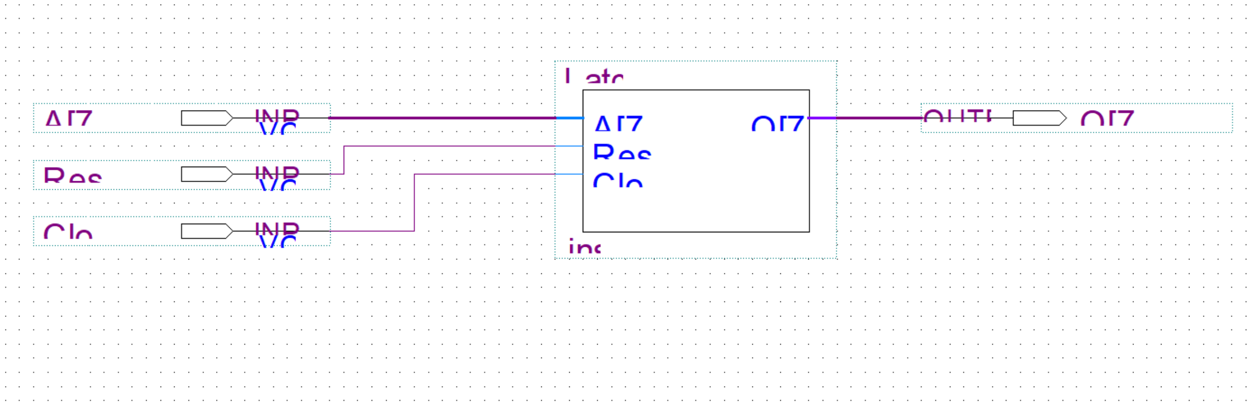
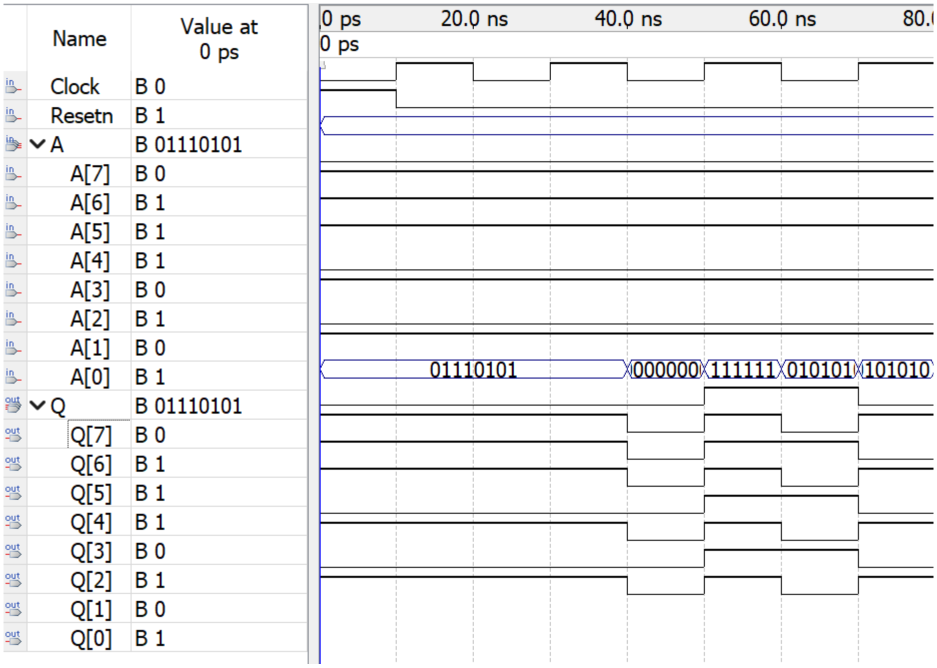


Figure 3.1.0 Latch1 Block Diagram



Name	Value at 0 ps
Clock	B 0
Resetn	B 1
A	B 01110101
A[7]	B 0
A[6]	B 1
A[5]	B 1
A[4]	B 1
A[3]	B 0
A[2]	B 1
A[1]	B 0
A[0]	B 1
Q	B 01110101
Q[7]	B 0
Q[6]	B 1
Q[5]	B 1
Q[4]	B 1
Q[3]	B 0
Q[2]	B 1
Q[1]	B 0
Q[0]	B 1

Figure 3.1.1 Latch1 Waveform Diagram

3.2 Latch 2

Similarly to Latch1, Latch2 holds the 8-bit value B with 8 D-flip flops. With the clock as an input as well, its output is used alongside Latch1 for the ALU. In binary format, B = $(100100)_2$.

CLOCK	RESET	B[0..7]	Q[0..7]
0	1	100100	000000
1	0	100100	100100
0	0	100100	100100
1	0	100100	100100
0	0	100100	000000
1	0	100100	000000
0	0	100100	100100
1	0	100100	100100

Table 3.2.0 Latch2 Truth Table

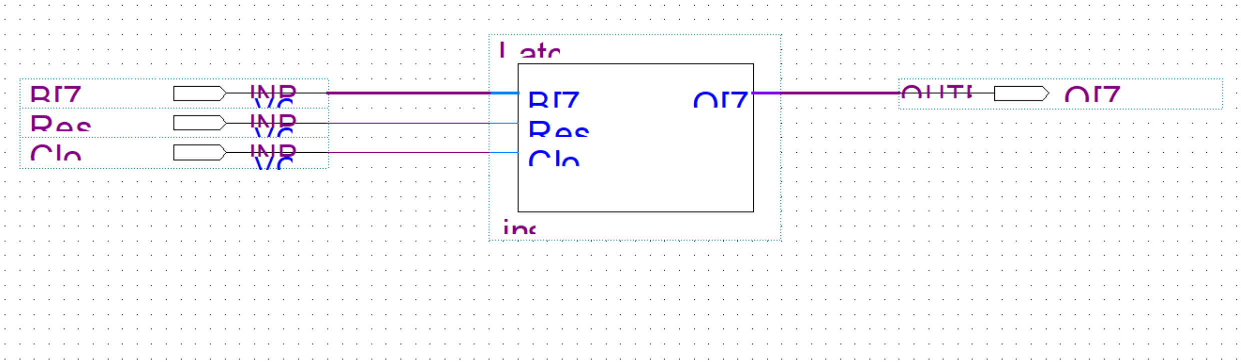
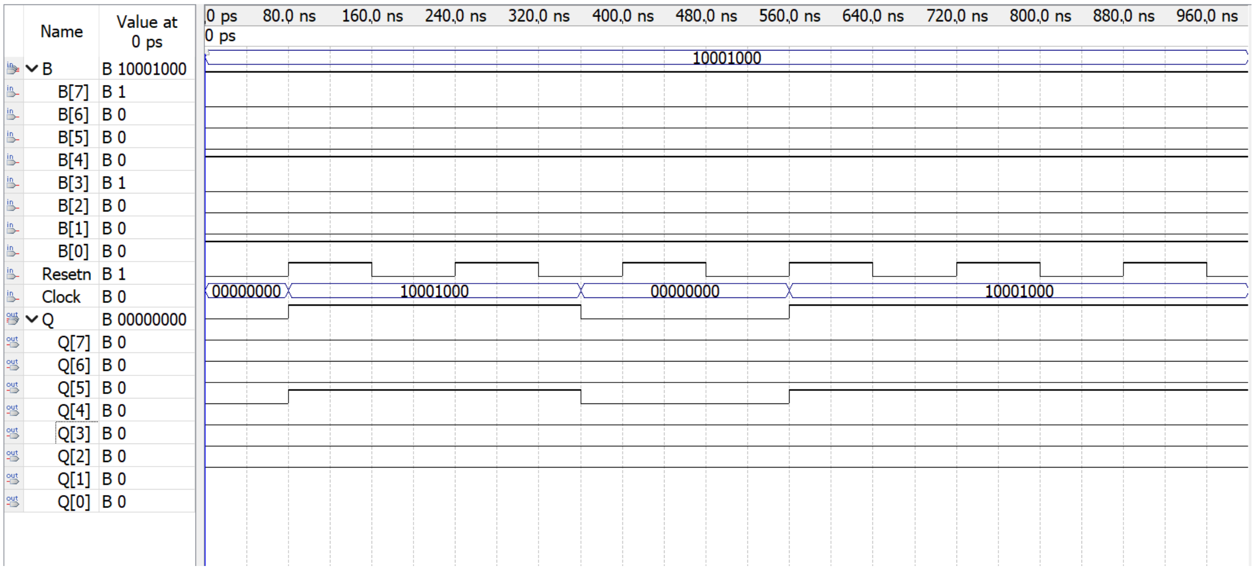


Figure 3.2.0 Latch2 Block Diagram



Name	Value at 0 ps
B	B 10001000
B[7]	B 1
B[6]	B 0
B[5]	B 0
B[4]	B 0
B[3]	B 1
B[2]	B 0
B[1]	B 0
B[0]	B 0
Resetn	B 1
Clock	B 0
Q	B 00000000
Q[7]	B 0
Q[6]	B 0
Q[5]	B 0
Q[4]	B 0
Q[3]	B 0
Q[2]	B 0
Q[1]	B 0
Q[0]	B 0

Figure 3.2.1 Latch2 Waveform

3.3 4x16 Decoder

The purpose of a decoder in a GPU is to translate instructions in the programming to signals that can be used by the machine. The first value, in this case 4 represents the number of inputs, while the second represents the outputs. While it is possible to create a 4x16 decoder using two 3x8 decoders (having inputs x,y and E respectively, this GPU consists of a simple 4x16 decoder instead, with the four inputs represented as w(from 0 to 3) and y as the output (going from 0 to 15).

W[0..3] Input	**Y[0..15] Output**
0000	1000000000000000
0001	0100000000000000
0010	0010000000000000
0011	0001000000000000
0100	0000100000000000
0101	0000010000000000
0110	0000001000000000

0111	0000000100000000
1000	0000000010000000
1001	0000000001000000
1010	0000000000100000
1011	0000000000010000
1100	0000000000001000
1101	0000000000000100
1110	0000000000000010
1111	0000000000000000

Table 3.3.0 4x16 Truth Table

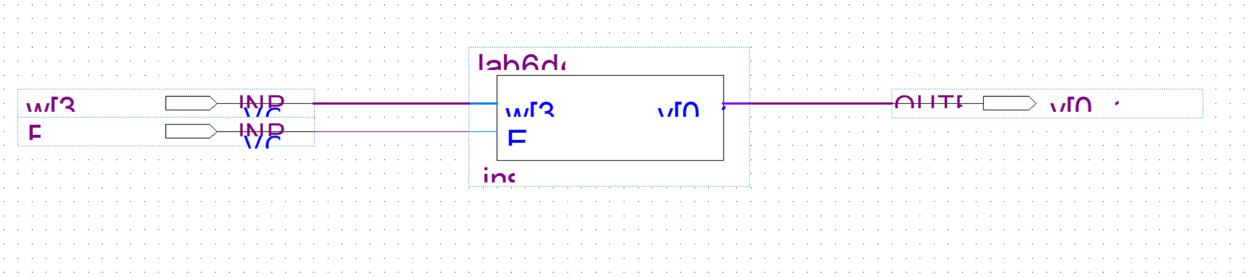


Figure 3.3.0 4x16 Block Diagram

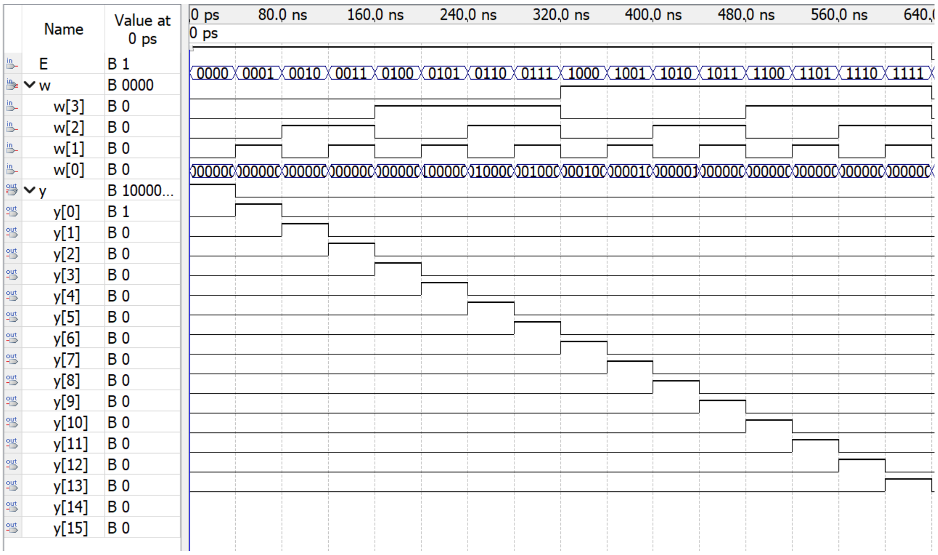


Figure 3.3.1 4x16 Waveform

3.4 Finite State Machine

The Finite State Machine is a powerful component of the GPU design. With a focus on translating between states using the inputs, they make up half of the control unit. In this case, it consists of the clock, input and reset. However, what makes it crucial to this GPU is that it has both the student id and current state in its output.

Current State	Next State	Student_id
0000	0001	0101
0001	0010	0000
0010	0011	0001
0011	0100	0001
0100	0101	1001
0101	0110	1010
0110	0111	1011
0111	1000	1010
1000	0000	1000

Table 3.4.0 Finite State Machine Truth Table

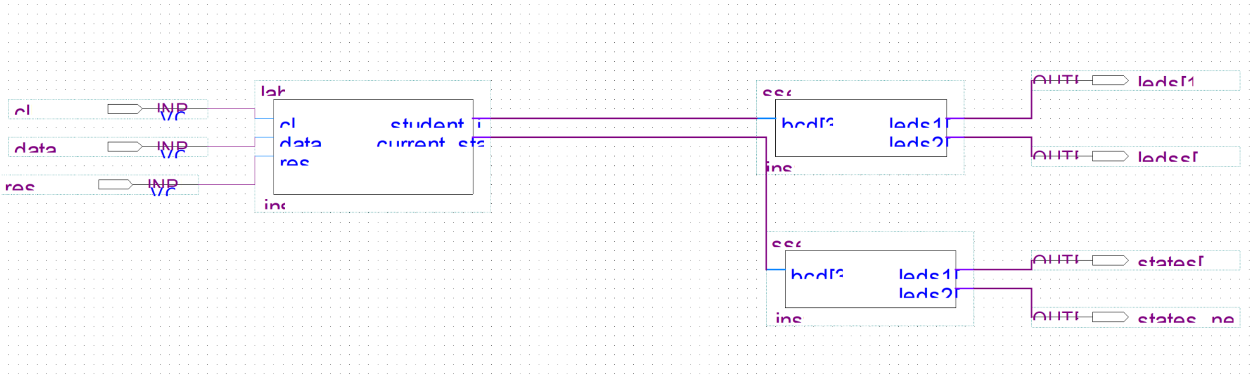
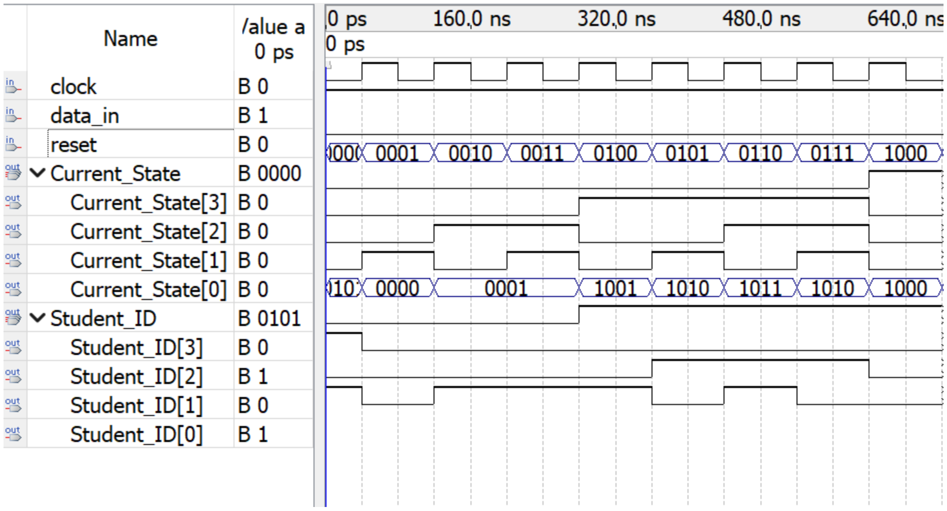


Figure 3.4.0 Finite State Machine Block Diagram



Name	/alue a 0 ps
clock	B 0
data_in	B 1
reset	B 0
Current_State	B 0000
Current_State[3]	B 0
Current_State[2]	B 0
Current_State[1]	B 0
Current_State[0]	B 0
Student_ID	B 0101
Student_ID[3]	B 0
Student_ID[2]	B 1
Student_ID[1]	B 0
Student_ID[0]	B 1

Figure 3.4.1 Finite State Machine Waveform

4.0 ALU_1 for Problem Set 1 of the Lab 6 procedure

The first component design consists of a regular GPU structure. Including all of the components mentioned earlier, the most important part of this would be the Arithmetic Logic Unit, given the name “ALUCore". As the ALU is where all calculations and arithmetic operations occur, the output (result) is connected to two Seven Segment Displays and represents the 8-bit output of any calculations done using the previously mentioned A and B values. The primary inputs would be the A and B values coming from Latch1 and Latch2.

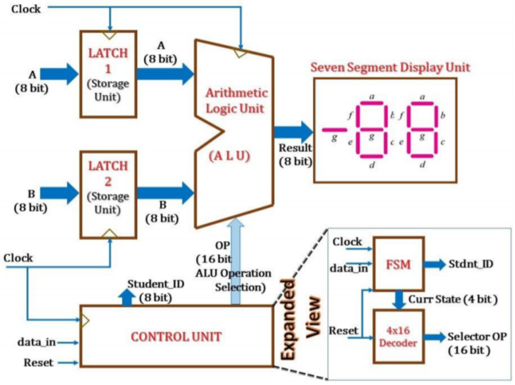


Figure 4.0.0 Block Diagram of the GPU [1]

Function #	Microcode	Boolean Operation	Output2	Output16
1	1000000000000000	Sum(A,B)	11111101	FD
2	0100000000000000	Diff(A,B)	11111101	FD
3	0010000000000000	NOR A	10001010	8A
4	0001000000000000	A NAND B	11111110	FE
5	0000100000000000	A NOR B	11111111	FF
6	0000010000000000	A AND B	00000000	00
7	0000001000000000	A XOR B	11111101	FD
8	0000000100000000	A OR B	11111101	FD
9	0000000010000000	A XNOR B	00000010	02

Table 4.0.0 Table of Microcodes for ALUCore [1]

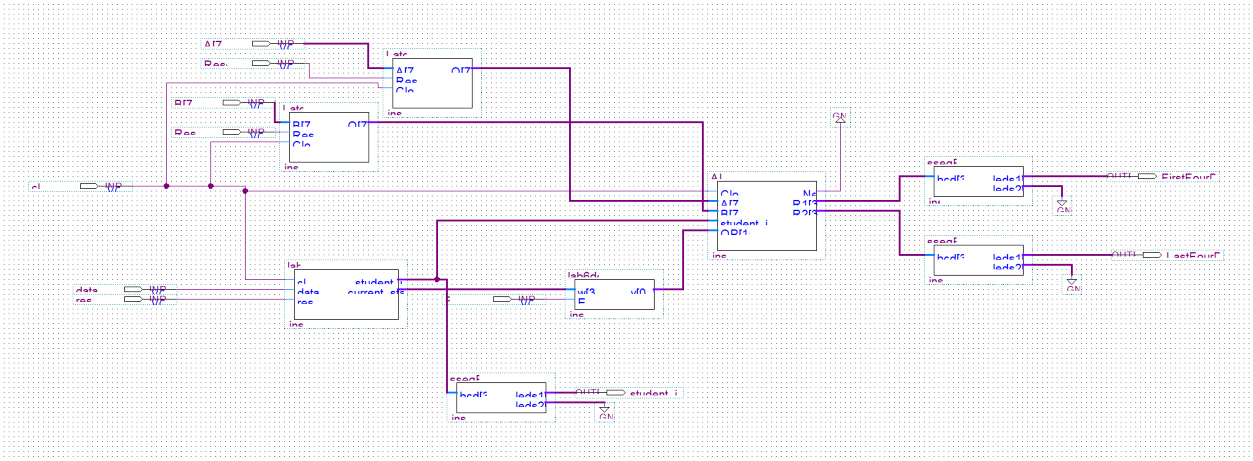


Figure 4.0.1 Block Diagram of Problem Set 1

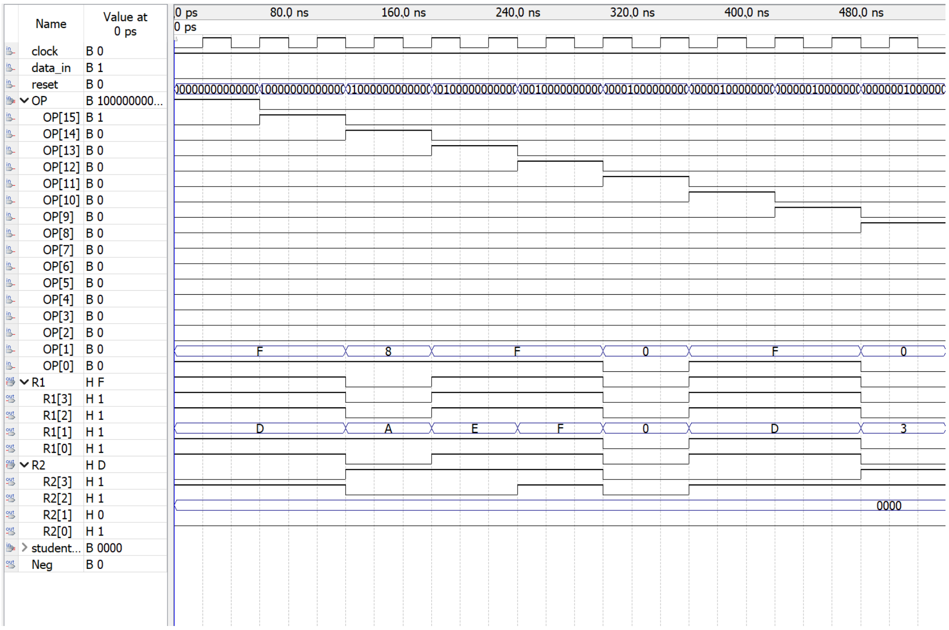


Figure 4.0.2 Problem 1 Waveform

5.0 ALU_2 for Problem Set 2 of the Lab 6 procedure

Problem 2 for this lab further explores modifying the ALU Core component, and requires following a set of functions. Once followed, the output in the Seven Segment Display Unit differs from the first lab and outputs all of the highest values. Once again, the primary inputs are those from Latch1 and Latch2; A and B respectively.

Function	Output2	Output16
Increment A by 2	01110111	77
Shift B 2 Bits Right, Input Bit = 0	00100010	22
Shift A 4 Bits Right, Input Bit = 1	00000111	07
Min(A,B)	01110101	75
Rotate A to the Right by 2 Bits	01011101	5D

Invert B’s Bit Significance Order	00010001	11
A XOR B	11111101	FD
Sum(A,B) Then Subtract by 4	11111001	F9
High Bits of Output	11111111	FF

Table 5.0.0 Table of Microcodes for Problem 2

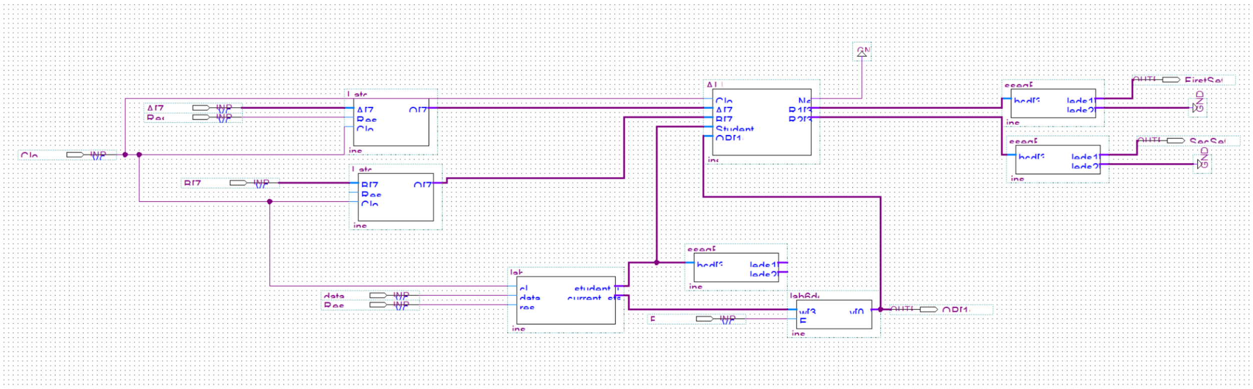


Figure 5.0.0 Block Diagram of Problem Set 2

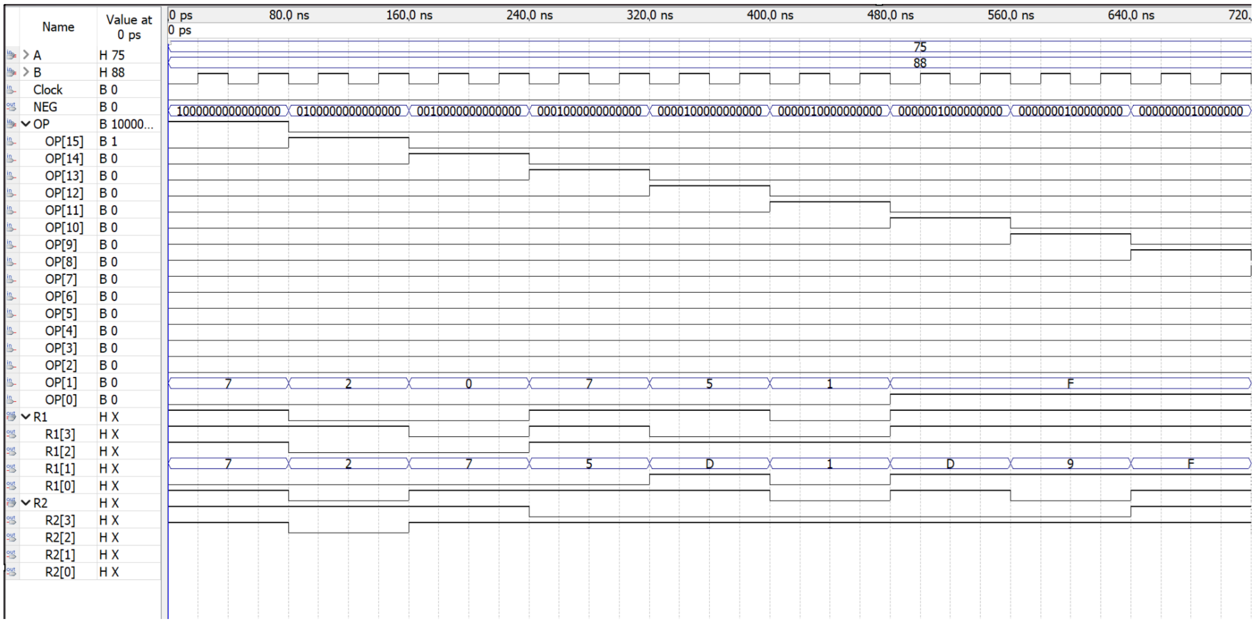


Figure 5.0.1 Problem 2 Waveform

6.0 ALU_3 for Problem Set 3 of the Lab 6 procedure

The final problem for the lab consists of modifying both the ALU and Seven Segment from the first section of the lab to conform with the required instruction. In this case, 'y' is to be output if the student ID output from FSM is odd, and 'n' would be output otherwise. Hence, the required inputs would be A, B, the 8-bit Student ID and the OP, which is a 16-bit ALU Operation Selection. Take note that the N value is displayed as 0010101 on the Seven Segment, while Y is displayed as 0111011.

Function	Output2	Output16	Even: 0010101 (N) Odd: 0111011 (Y)
Increment A by 2	01110111	77	0111011
Shift B 2 Bits Right, Input Bit = 0	00100010	22	0010101
Shift A 4 Bits Right, Input Bit = 1	00000111	07	0111011
Min(A,B)	01110101	75	0111011
Rotate A to the Right by 2 Bits	01011101	5D	0111011
Invert B's Bit Significance Order	00010001	11	0010101
A XOR B	11111101	FD	0010101
Sum(A,B) Then Subtract by 4	11111001	F9	0111011
High Bits of Output	11111111	FF	0111011

Table 6.0.0 Table of Microcodes for Problem 3

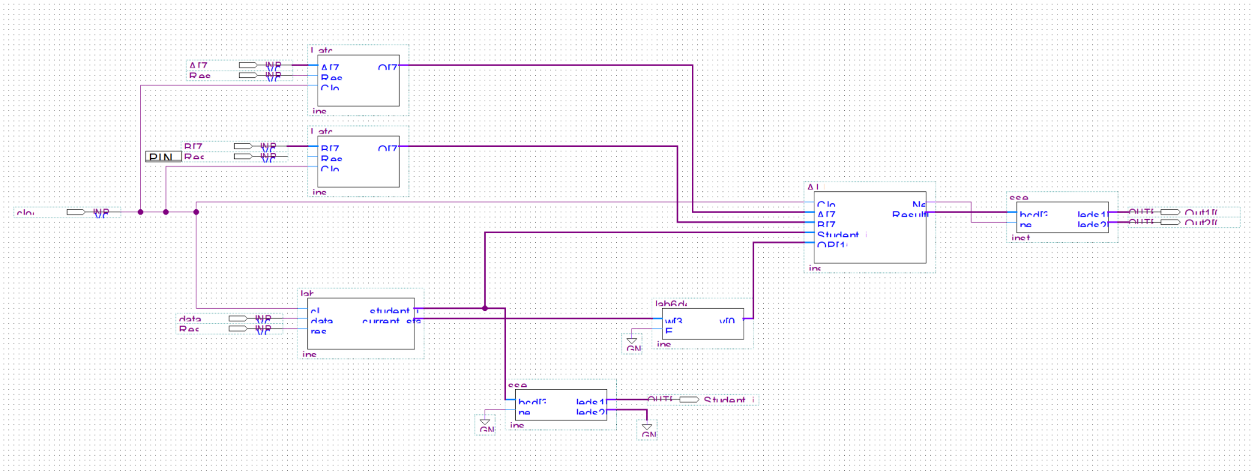
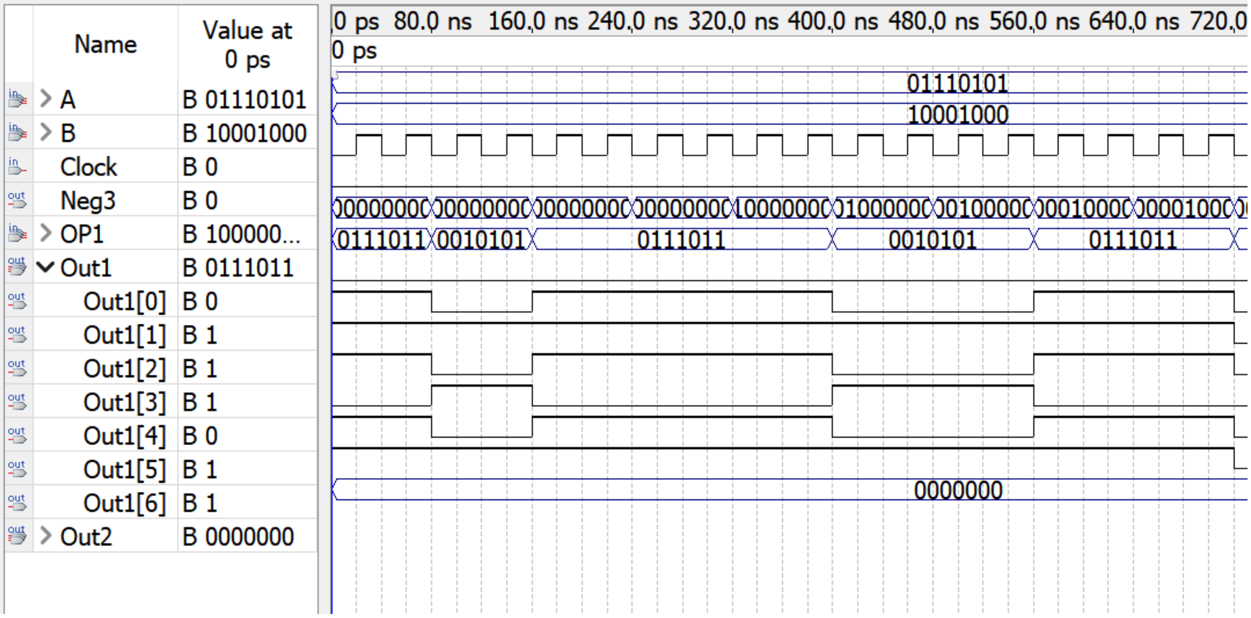


Figure 6.0.0 Block Diagram of Problem Set 3



Name	Value at 0 ps
A	B 01110101
B	B 10001000
Clock	B 0
Neg3	B 0
OP1	B 100000...
Out1	B 0111011
Out1[0]	B 0
Out1[1]	B 1
Out1[2]	B 1
Out1[3]	B 1
Out1[4]	B 0
Out1[5]	B 1
Out1[6]	B 1
Out2	B 0000000

Figure 6.0.1 Problem 3 Waveform

7.0 Conclusion

In conclusion, this lab designs and provides in-depth analysis of numerous aspects of a GPU. Utilizing the respective Storage Units, Arithmetic Logic Unit, Control Unit and Seven Segment Display Unit, these four crucial components create the general architecture. By first creating the VHDL code, then making a block diagram final and finally creating the waveform and implementing hardware, it concisely establishes the fundamentals of one of many systems that make up a computer.

8.0 References

[1] COE/BME 328 – Digital Systems. *Design of a Simple General-Purpose Processor* (Lab 6, pp. 1–13).