Lab 6 Report

Instructor: Dr. Reza Sedaghat
TA: Leandra Budau
By: Haran Muruganantham
ID: 501166674

Table Of Contents

Introduction

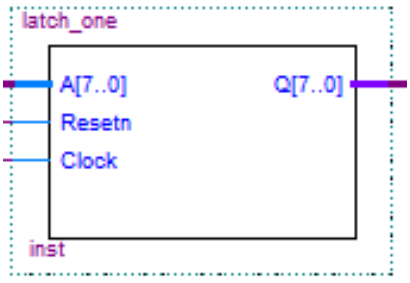
The goal of this lab is to put together a general processing unit using knowledge built up from previous labs and lectures. This lab can be split into four components. The first component is the latches, which are used to store memory. The second component is the decoder, which is used to convert data. The third component is the FSM, which will decide the pattern of how the functions are performed. The final component is the ALU, where all the instructions are held, and the actions are performed. When all four components are combined, the result is a functioning general processing unit. Finally, all the output data from the ALU will be presented on a seven-segment display for verification.

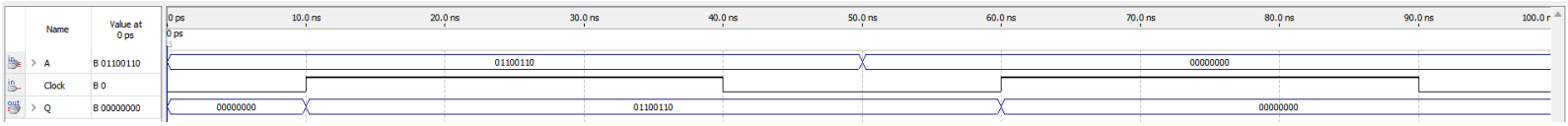
General Components

Latches

In this general processing unit, latches are constantly used as a temporary storage unit. The latch used in this lab is formally known as a D-latch since it is positively edge-triggered. This means the latches will hold the user's inputted A and B once a clock edge and keep the value until the next edge. In the case of the lab, since the values imputed are constant throughout the clock cycle, the latch outputs never change.

Input (Clock)	**Output (Q)**
0	Q(t-1)
1	Q(t)

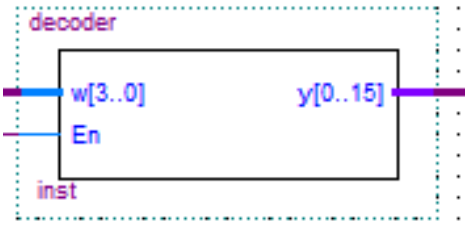


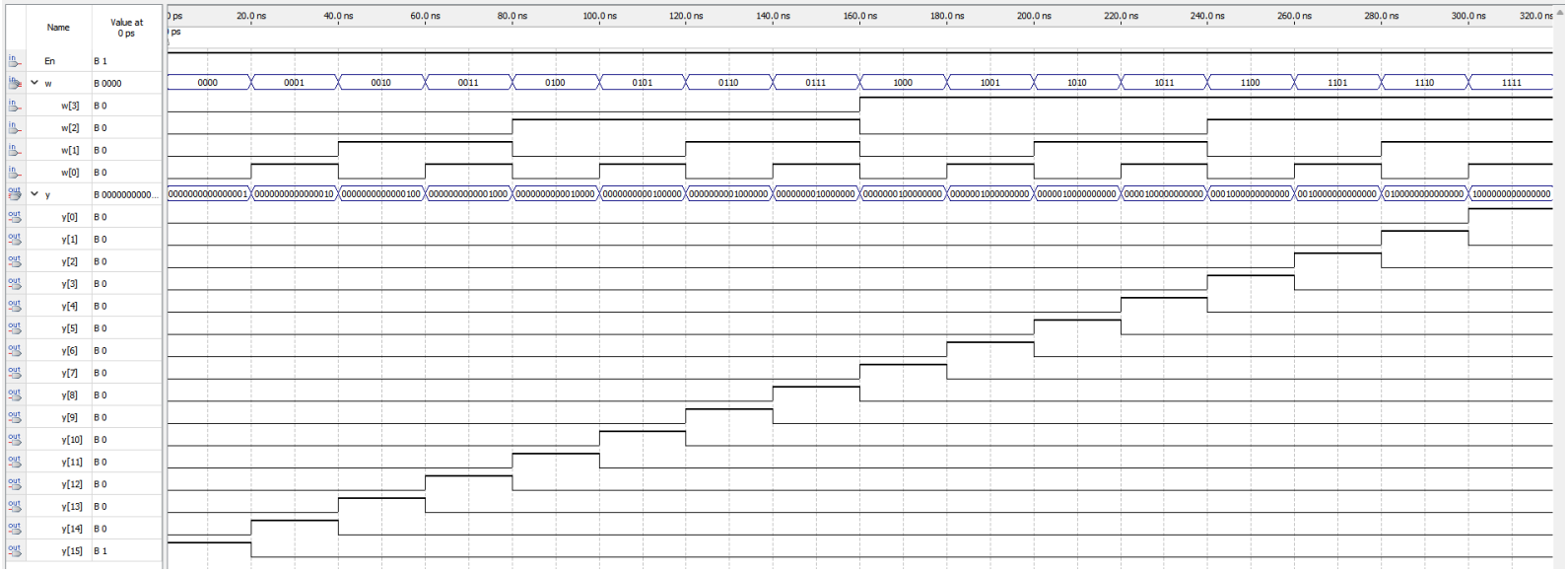


Decoder

In this general processing unit, a decoder outputs a 16-bit microcode given a 4-bit value. This decoder is formally known as a one hot 4 to 16 decoder. A simple way of seeing how this decoder works is by taking the decimal value of the 4-bit input and only turning on that bit position for the 16-bit output. For example, input 0101, which is five in decimal, would yield 0000000000100000 as the microcode output. Also, this lab only utilizes 9 out of the 16 microcode outputs since the FSM counts from 0 to 8. In the circuit for this lab, the decoder is combined with the FSM in the control unit.

Input (w)	**Output (y)**
0000	0000000000000001
0001	0000000000000010
0010	0000000000000100
0011	0000000000001000
0100	0000000000010000
0101	0000000000100000
0110	0000000001000000
0111	0000000010000000
1000	0000000100000000
1001	0000001000000000
1010	0000010000000000
1011	0000100000000000
1100	0001000000000000
1101	0010000000000000
1110	0100000000000000
1111	1000000000000000

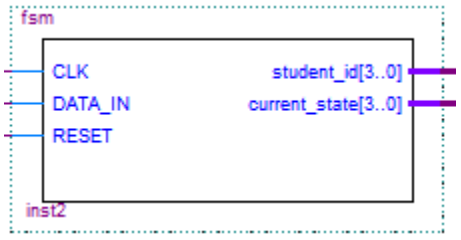


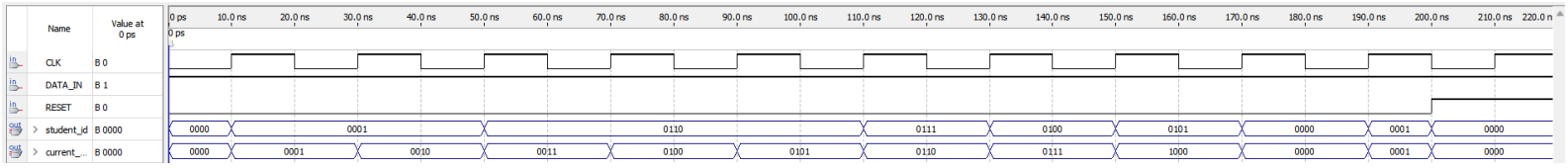


Finite State Machine

In this general processing unit, a finite state machine, commonly known as an FSM, controls the ALU function. The FSM used in this lab is a Moore-type machine that outputs based only on its current state, and it will go up one state for every clock edge, starting at 0, going to 8, and then restarting at 0. This FSM will output the current state value as a 4-bit binary value and the student ID as a 4-bit binary value. Furthermore, the current state will be sent to the decoder's 4-bit input. In the circuit for this lab, the FSM is combined with the decoder in the control unit.

Present State	Next State (w = 0)	Next State (w = 1)	Output
0000	0000	0001	0101
0001	0000	0010	0000
0010	0000	0011	0001
0011	0000	0100	0001
0100	0000	0101	0110
0101	0000	0110	0110
0110	0000	0111	0110
0111	0000	1000	0111
1000	0000	0000	0100





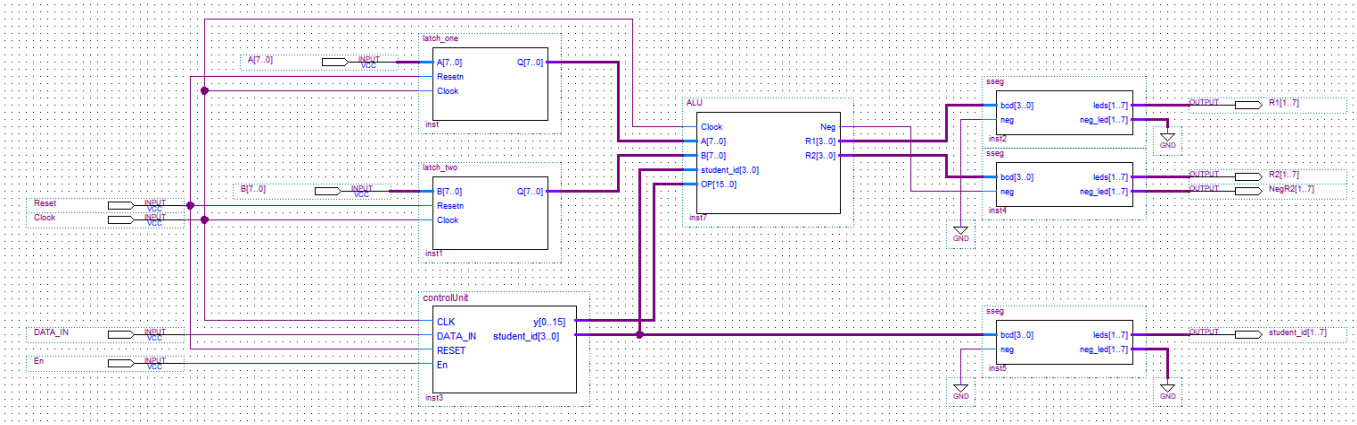
Arithmetic Logic Units

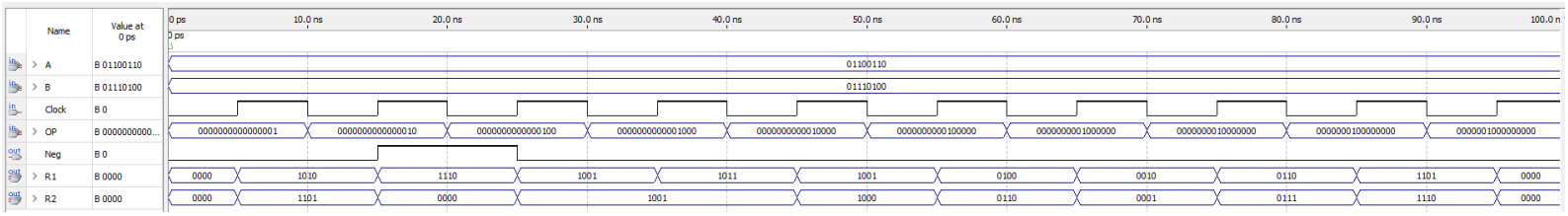
Unit 1

The first ALU will perform the actions in the first problem set to the input A and B values. The ALU will take in the 16-bit microcode output from the decoder and use it to call the corresponding function. Since all the elements are on a common clock, once the clock edges, the function will perform, and then the result will be outputted. The ALU will output two 4-bit binary values, which will then be converted to a hexadecimal value. Also, the ALU will output the sign of the calculated value.

Function Number	**Microcode**	**Function**
1	0000000000000001	sum(A, B)
2	0000000000000010	diff(A, B)
3	0000000000000100	NOT A
4	0000000000001000	A NAND B
5	0000000000010000	A NOR B
6	0000000000100000	A AND B
7	0000000001000000	A XOR B
8	0000000010000000	A OR B
9	0000000100000000	A XNOR B

Function **A** = $(66)_{16}$ **& B** = $(74)_{16}$	**Binary Output**	**Hexadecimal Output**
sum(A, B)	11011010	DA
diff(A, B)	-00001110	-0E
NOT A	10011001	99
A NAND B	10011011	9B
A NOR B	10001001	89
A AND B	01100100	64
A XOR B	00010010	12
A OR B	01110110	76
A XNOR B	11101101	ED



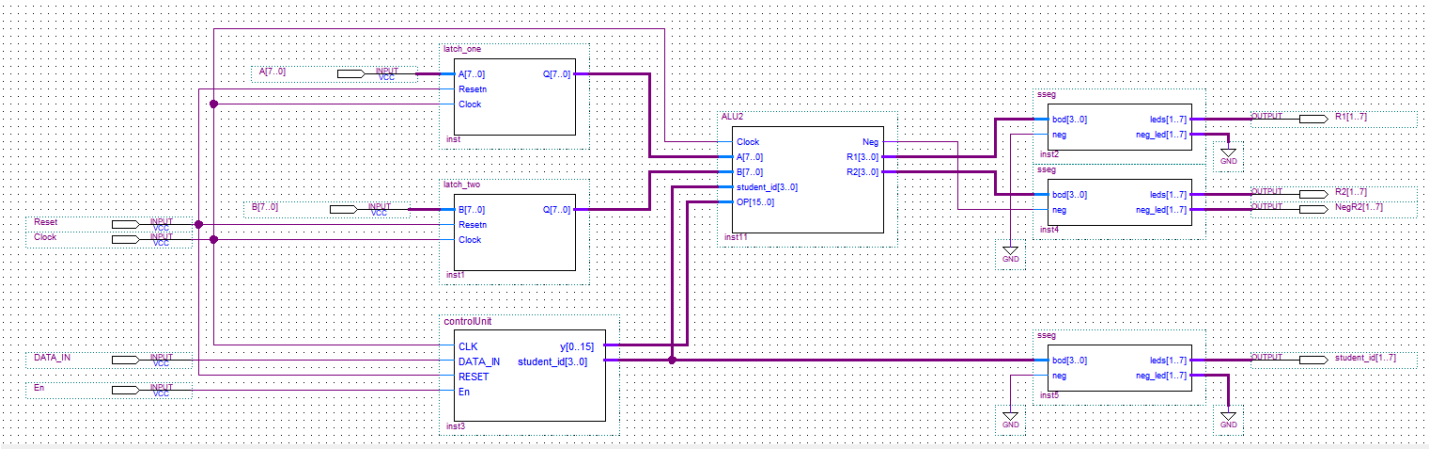


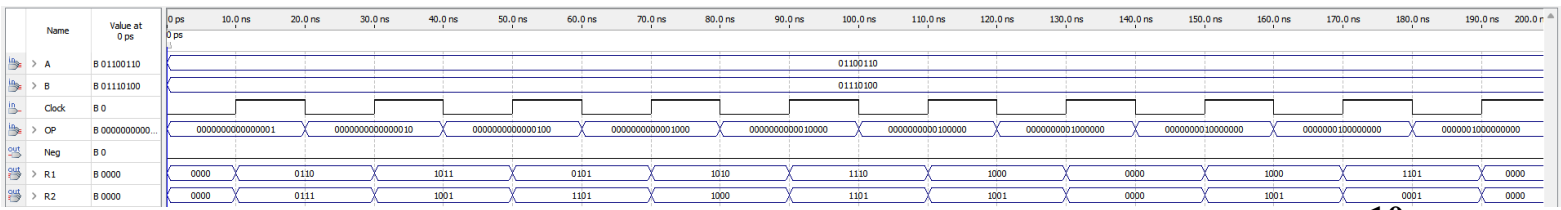
Unit 2

The second ALU will perform the actions in set E for the second problem. This ALU works the same way as the first ALU, with the only difference being the instructions contained in the ALU. The nine functions being performed are based on the assigned problem set.

Function Number	**Microcode**	**Function**
1	0000000000000001	Replace the odd bits of A with the odd bits of B
2	0000000000000010	Produce the results of NANDing A and B
3	0000000000000100	Calculate the summation of A and B and decrease it by 5
4	0000000000001000	Produce the 2's complement of B
5	0000000000010000	Invert the even bits of B
6	0000000000100000	Shift A to the left by 2 bits, input bit = 1 (SHL)
7	0000000001000000	Produce null on the output
8	0000000010000000	Produce the 2's complement of A
9	0000000100000000	Rotate B to right by 2 bits (ROR)

Function **A** = $(66)_{16}$ **& B** = $(74)_{16}$	**Binary Output**	**Hexadecimal Output**
Replace the odd bits of A with the odd bits of B	01110110	76
Produce the results of NANDing A and B	10011011	9A
Calculate the summation of A and B and decrease it by 5	11010101	D5
Produce the 2's complement of B	10001010	8A
Invert the even bits of B	11011110	DE
Shift A to the left by 2 bits, input bit = 1 (SHL)	10011000	98
Produce null on the output	00000000	00
Produce the 2's complement of A	10011000	98
Rotate B to right by 2 bits (ROR)	00011101	1D

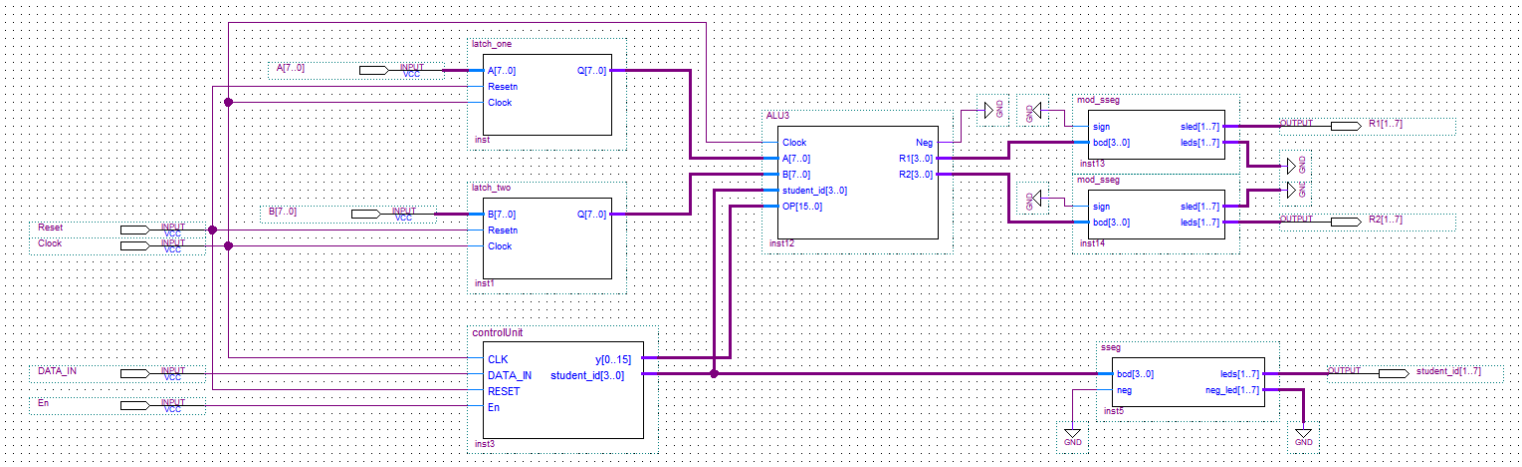


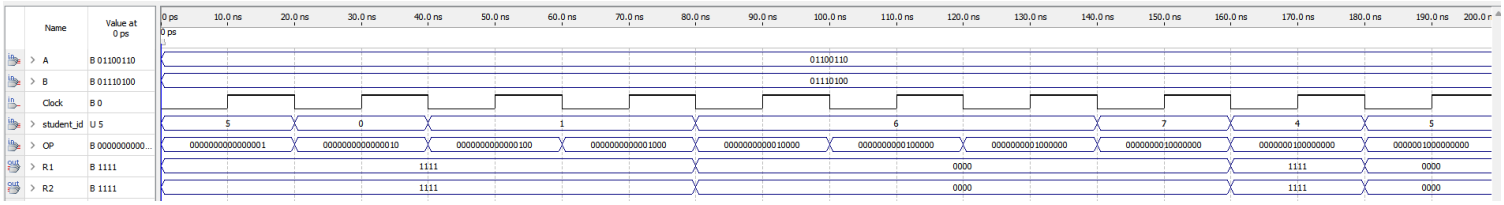


Unit 3

The third ALU unit is very different from the first and second since it only performs a single operation. The operation is a comparison operation to check if any of the digits in A are greater than the values of the student ID. If the comparison is valid, the output will be 1111; if it is false, it will be 0000. Also, this ALU requires a modified version of the seven-segment display code so that the output from the ALU will be converted to a Y for true and an N for false.

Student Number	**Condition**
5	Y
0	Y
1	Y
1	Y
6	N
6	N
6	N
7	N
4	Y





Conclusion

In conclusion, this lab put together various elements learned in previous labs and lectures using various techniques in VHDL. This lab showcased how to use latches as registers and memory. It showed how to use an FSM and decoder to create a control unit. Finally, it represented how an ALU performs actions given certain microcodes.