Lab 6 - Design of a Simple Central Processing Unit

Wu, Tommy

COE328 - Digital Systems - Fall 2024

Professor: V. Geurkov; Teaching Assistant: M. Li; Section 7

December 12, 2024

Table of Contents

1. Introduction …………………………………… Page 2
2. Components …………………………………… Page 3
    - Latch1
    - Latch2
    - 4:16 Decoder
    - FSM
3. ALU_1 for Problem Set 1 ……………….. Page 10
4. ALU_2 for Problem Set 2 ……………….. Page 13
5. ALU_3 for Problem Set 3 ……………….. Page 15
6. Conclusion …………………………………… Page 18

DISCLAIMER

1.0 Introduction

This report presents the design and implementation of a simple central processing unit (CPU) in a VHDL environment. The project encompasses creating a functional arithmetic and logic unit (ALU), a finite state machine (FSM), and a 4:16 decoder. These components are integrated to execute basic operations and simulate a CPU using Quartus software. The CPU fetches instructions, processes inputs, and generates outputs. The design utilizes a systematic approach, incorporating components like registers, decoders, and ALU cores for efficient data processing. The manual-driven process ensures modularity and accuracy in implementation.

2.0 Components

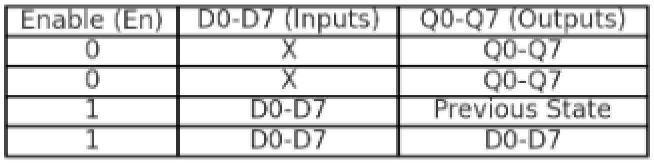
Latch1

The latch is a memory element used to store X-bit data based on an enable signal. It acts as a temporary storage component critical in sequential circuits. In the lab, it was able to store up to 8 bits of binary data, with a maximum decimal value of 255.

- **Truth Table:**

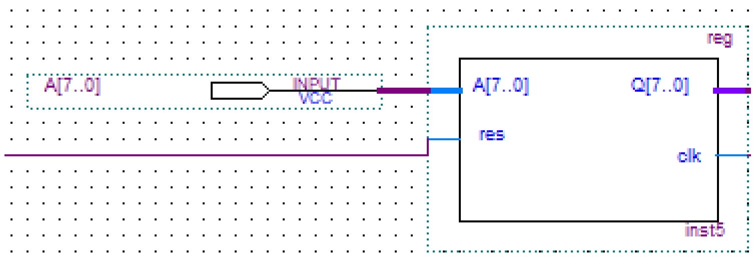
Throughout the entire Lab, Latch 1 = A = 35.

D0-D7 in binary is equal to 35 as decimal.

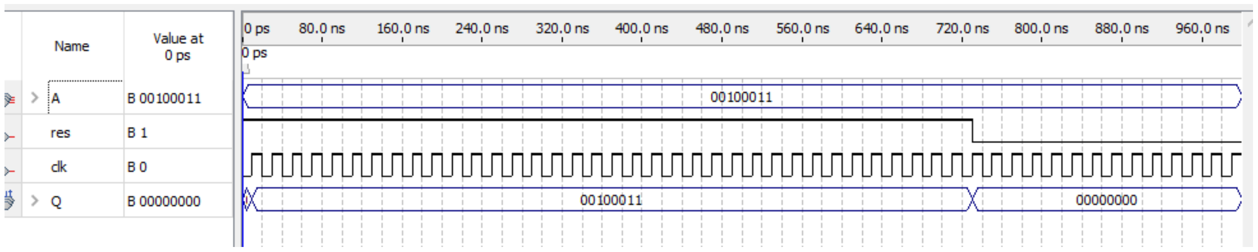


Enable (En)	D0-D7 (Inputs)	Q0-Q7 (Outputs)
0	X	Q0-Q7
0	X	Q0-Q7
1	D0-D7	Previous State
1	D0-D7	D0-D7

- **Circuit/Block Diagram:**



- Waveform File:



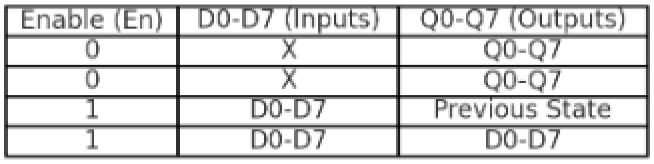
Latch2

Similar to Latch1, this component temporarily holds data, ensuring the correct flow of information during operations. The second latch is to hold another binary number to perform operations on between data in the two latches.

- Truth Table:

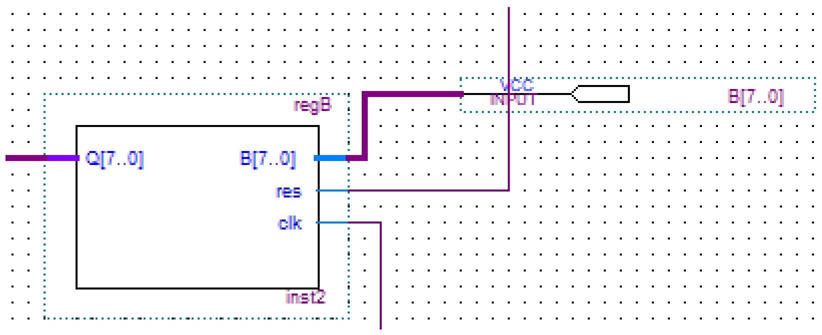
Throughout the entire Lab, Latch 2 = B = 76.

D0-D7 in binary is equal to 76 as decimal.



Enable (En)	D0-D7 (Inputs)	Q0-Q7 (Outputs)
0	X	Q0-Q7
0	X	Q0-Q7
1	D0-D7	Previous State
1	D0-D7	D0-D7

- Circuit/Block Diagram:



- Waveform File: Same as Latch 1 (just with different value)

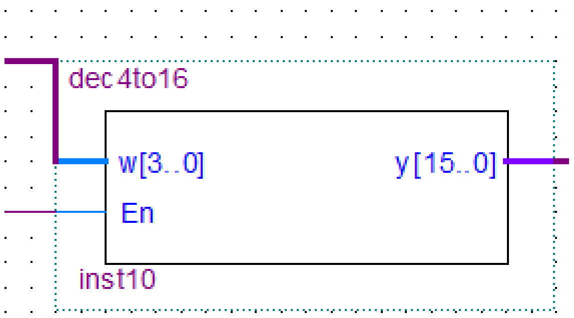
4:16 Decoder

The decoder converts the 3-bit FSM output into a 1-out-of-8 code to determine the ALU operation. It is essential for instruction selection in the CPU. Although only 8 codes are needed for the ALU, the 4:16 Decoder allows for better scalability.

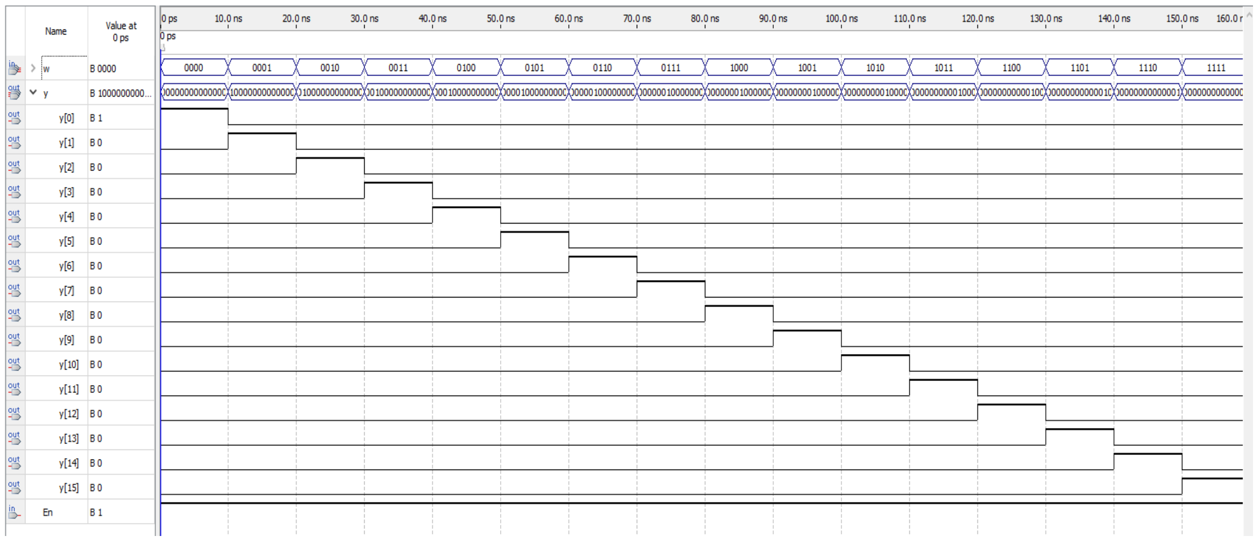
- Truth Table:

En	W3	W2	W1	W0	Cout
1	0	0	0	0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0	0	0	1	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0	0	1	0	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0	0	1	1	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1	0	1	0	0	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
1	0	1	0	1	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
1	0	1	1	0	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1	0	1	1	1	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1	1	0	0	0	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1	1	0	0	1	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1	1	0	1	0	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1	1	0	1	1	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1	1	1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
1	1	1	0	1	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1	1	1	1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1	1	1	1	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

- Circuit/Block Diagram:



- Waveform File:



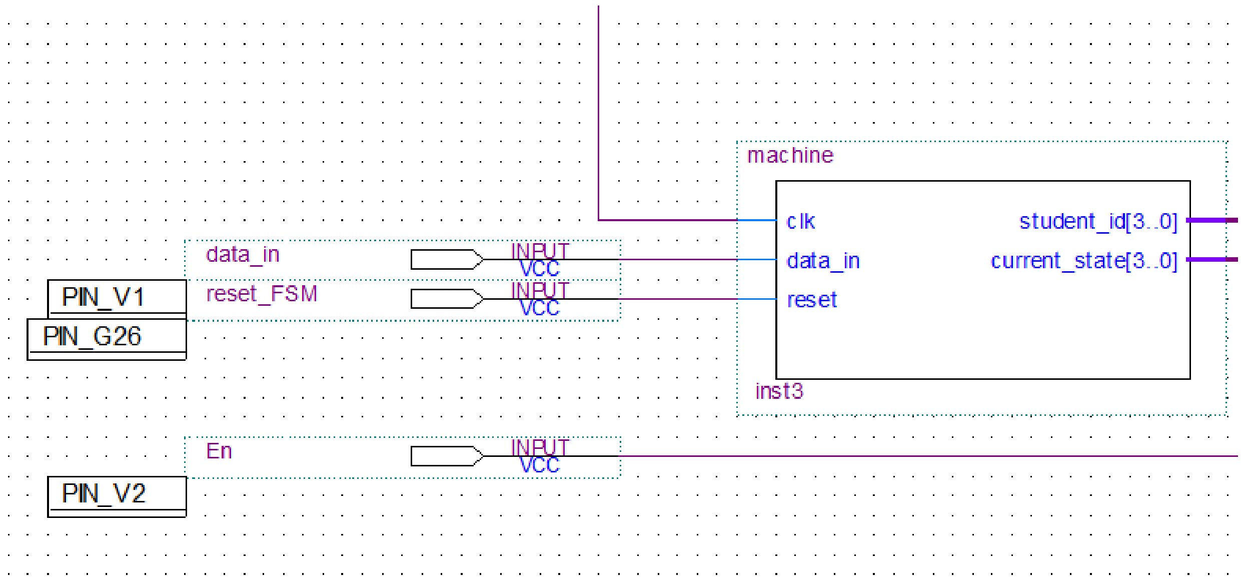
FSM

The FSM acts as a program counter, cycling through states to produce the control signals for the decoder. It synchronizes operations in the system. With each state starting from State 0, it will print out my student ID in the sequence, 0, 1, 2, 4, 3, 5, 7, 6. This excludes my first digit of my ID as the state machine only cycles 8 states.

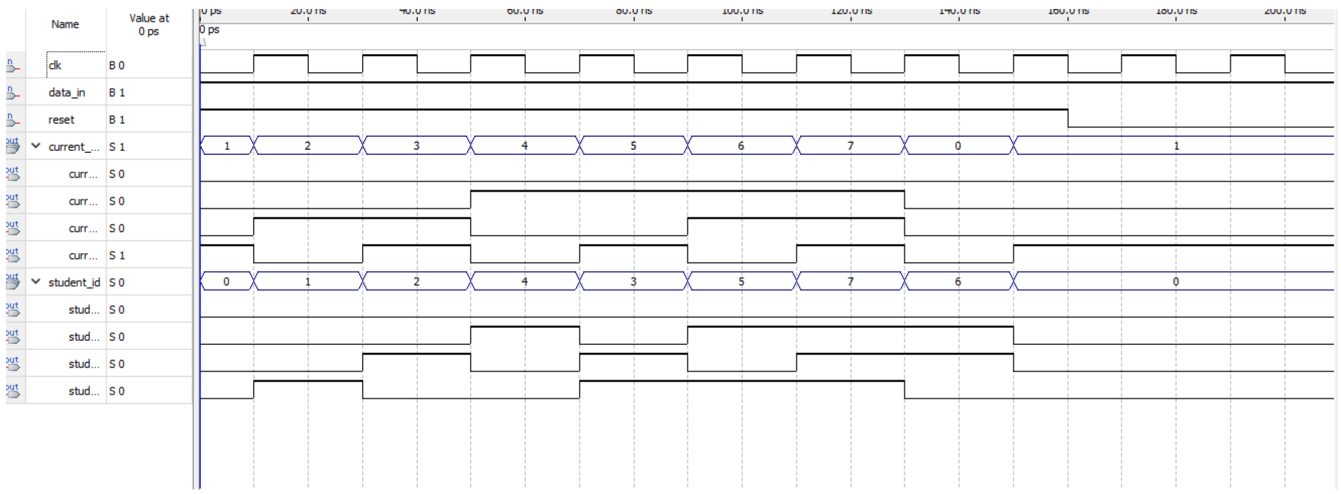
- Truth Table:

Present State (y3y2y1y0)	Next State (w=0)	Next State (w=1)	Cout (w=0)	Cout (w=1)
0000	0000	0001	0000	0000
0001	0001	0010	0001	0001
0010	0010	0011	0010	0010
0011	0011	0100	0100	0100
0100	0100	0101	0011	0011
0101	0101	0110	0101	0101
0110	0110	0111	0111	0111
0111	0111	0000	0110	0110

- Circuit/Block Diagram:



- Waveform File:



3.0 ALU_1 for Problem Set 1

The Arithmetic and Logic Unit (ALU) is a key component of the CPU, responsible for performing all arithmetic and logical operations. For this lab, the ALU had five inputs—A, B, Clock, student_id, and OP—and three outputs—Neg, R1, and R2. The objective of this task was to modify the ALU so that it performs the correct Boolean operation on the 8-bit values A and B, based on the assigned microcode, and produces the correct result. To achieve this, the ALU uses the 16-bit value of the "current state" from the FSM as a selector for the operation to be performed between A and B.

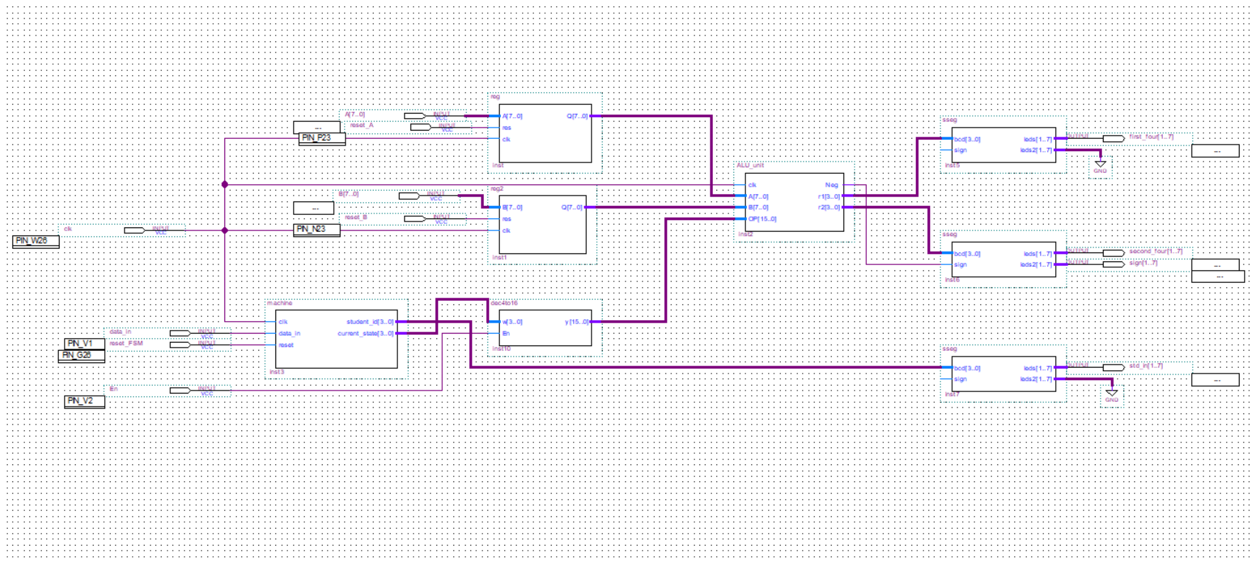
- Inputs for Problem Set 1
    - A[0.7] = Latch 1 for 8 bit value
    - B[0.7] = Latch 2 for 8 bit value
    - Reset_A/B = Reset Values for Latch 1, 2
    - FSM_Reset = Count Reset for State Machine
    - Clk = Toggle to Change Cycle for all components
    - En = Enable 4-16 Decoder
    - Data In = Machine will either store or retain this data based on the enable signal.
- Outputs for Problem Set 1
    - First_Four = first 4 bits of 8 bit output value from ALU
    - Second_Four = last 4 bits of 8 bit output value from ALU
    - Sign = Outputs a dash sign (--) if the output value is negative

- Std_in = Cycles through the last 8 digits of my student ID for each cycle of the FSM

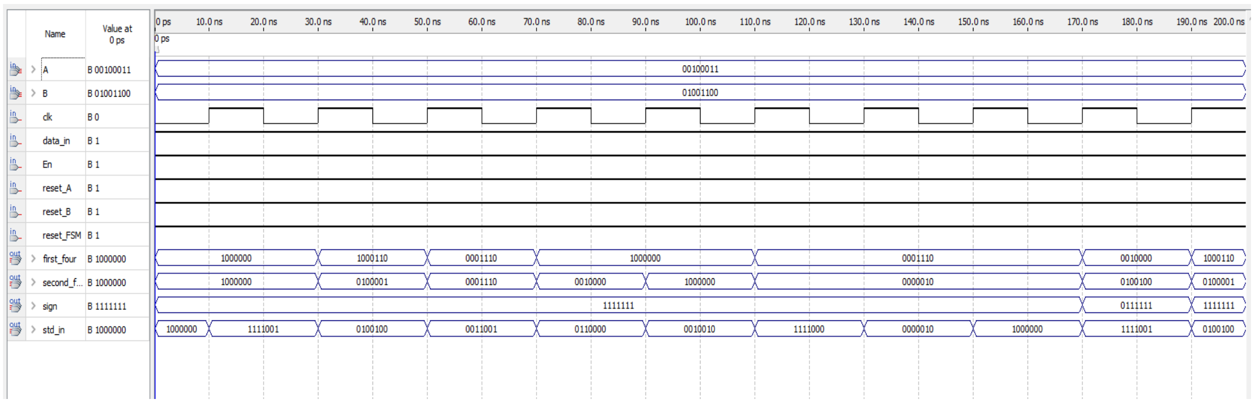
- Table of Microcodes:

Function #	Opcode	Function
1	00000001	$sum(\boldsymbol{A},\boldsymbol{B})$
2	00000010	$dif(\boldsymbol{A},\boldsymbol{B})$
3	00000100	$\overline{\boldsymbol{A}}$
4	00001000	$\overline{\boldsymbol{A}\cdot\boldsymbol{B}}$
5	00010000	$\overline{\boldsymbol{A}+\boldsymbol{B}}$
6	00100000	$\boldsymbol{A}\cdot\boldsymbol{B}$
7	01000000	$\boldsymbol{A}\oplus\boldsymbol{B}$
8	10000000	$\boldsymbol{A}+\boldsymbol{B}$

- Screenshot of Block Schematic File (BDF):



- Complete Waveform File:



4.0 ALU_2 for Problem Set 2

Problem Set 2 is identical to Problem Set 1, with a replaced ALU Unit with new functions. Refer to the Microcodes Table below where I was assigned Function E.

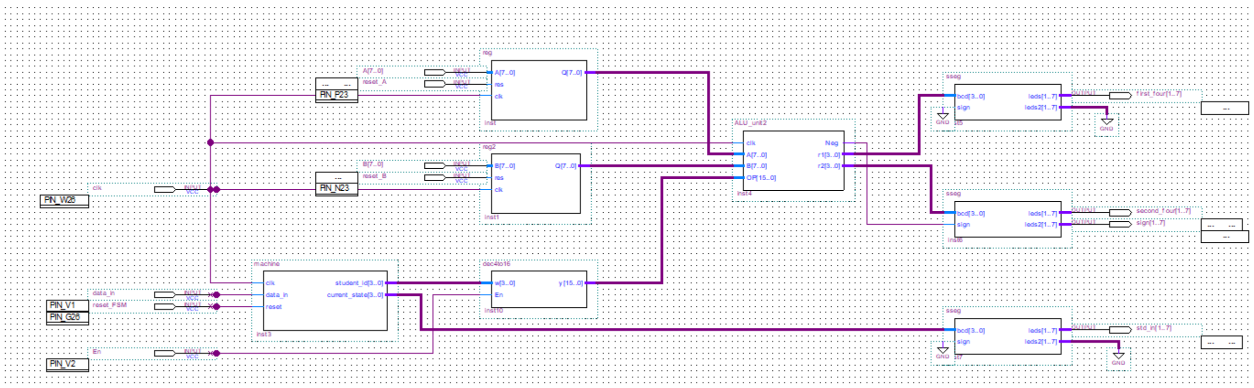
- Inputs for Problem Set 2
  - A[0.7] = Latch 1 for 8 bit value
  - B[0.7] = Latch 2 for 8 bit value
  - Reset_A/B = Reset Values for Latch 1, 2
  - FSM_Reset = Count Reset for State Machine
  - Clk = Toggle to Change Cycle for all components
  - En = Enable 4-16 Decoder
  - Data In = Machine will either store or retain this data based on the enable signal.
- Outputs for Problem Set 2
  - First_Four = first 4 bits of 8 bit output value from ALU
  - Second_Four = last 4 bits of 8 bit output value from ALU
  - Sign = Outputs a dash sign (--) if the output value is negative
  - Std_in = Cycles through the last 8 digits of my student ID for each cycle of the FSM

- Table of Microcodes:

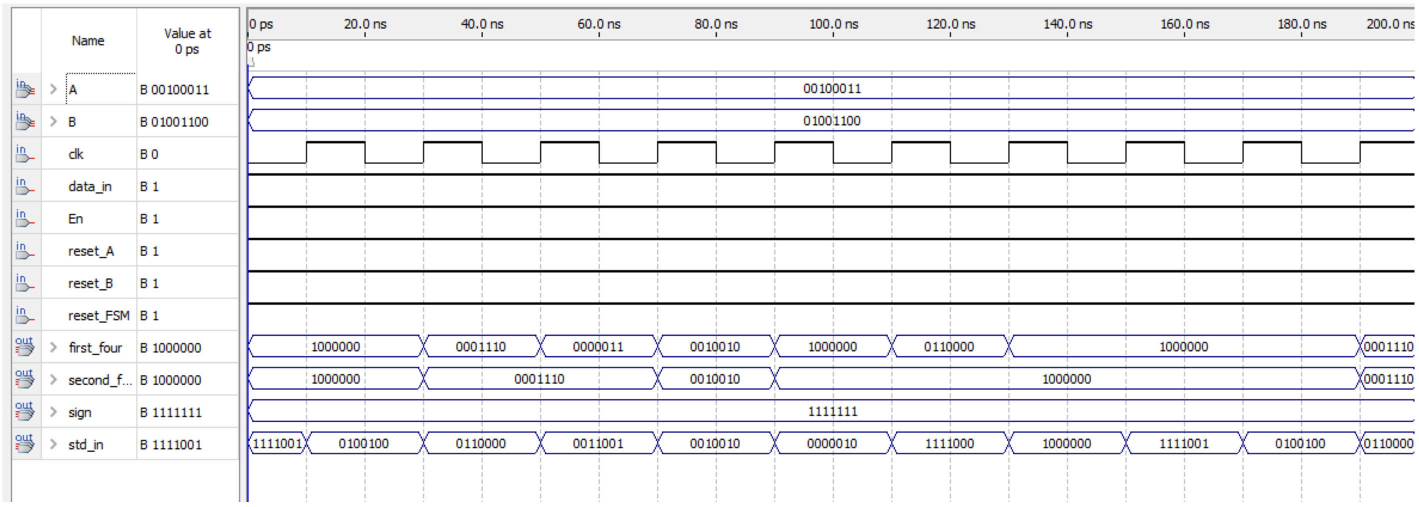
e)

Function #	Operation / Function
1	Replace the odd bits of **A** with odd bits of **B**
2	Produce the result of NANDing **A** and **B**
3	Calculate the summation of **A** and **B** and decrease it by 5
4	Produce the 2's complement of **B**
5	Invert the even bits of **B**
6	Shift **A** to left by 2 bits, input bit = 1 (SHL)
7	Produce null on the output
8	Produce 2's complement of **A**

- Screenshot of Block Schematic File (BDF):



- Complete Waveform File:



Name	Value at 0 ps
A	B 00100011
B	B 01001100
clk	B 0
data_in	B 1
En	B 1
reset_A	B 1
reset_B	B 1
reset_FSM	B 1
first_four	B 1000000
second_f...	B 1000000
sign	B 1111111
std_in	B 1111001

5.0 ALU_3 for Problem Set 3

The task for Problem Set 3 was to modify the ALU so that it could compare the two digits of input "A" with a student number over 9 clock cycles. If at least one of the digits of A was greater than the student number assigned to the current FSM state during a clock cycle, the FPGA board would output 'y' on one of the SSEGs. If neither digit matched, the output should be 'n'. To accomplish this, an input for "student_id" was added to the ALU, and conditional statements were incorporated into the ALU code. Rather than performing the boolean operation assigned to the current FSM state between A and B, the ALU now simply compares the "student_id" input with the digits of input A.

- Inputs for Problem Set 3
    - A[0.7] = Latch 1 for 8 bit value
    - Reset_A = Reset Values for Latch 1
    - FSM_Reset = Count Reset for State Machine
    - Clk = Toggle to Change Cycle for all components
    - En = Enable 4-16 Decoder
    - Data In = Machine will either store or retain this data based on the enable signal.
- Outputs for Problem Set 3
    - Second_Four = displays “Y” or “N” from ALU
    - Std_in = Cycles through the last 8 digits of my student ID for each cycle of the FSM

- Table of Microcodes:

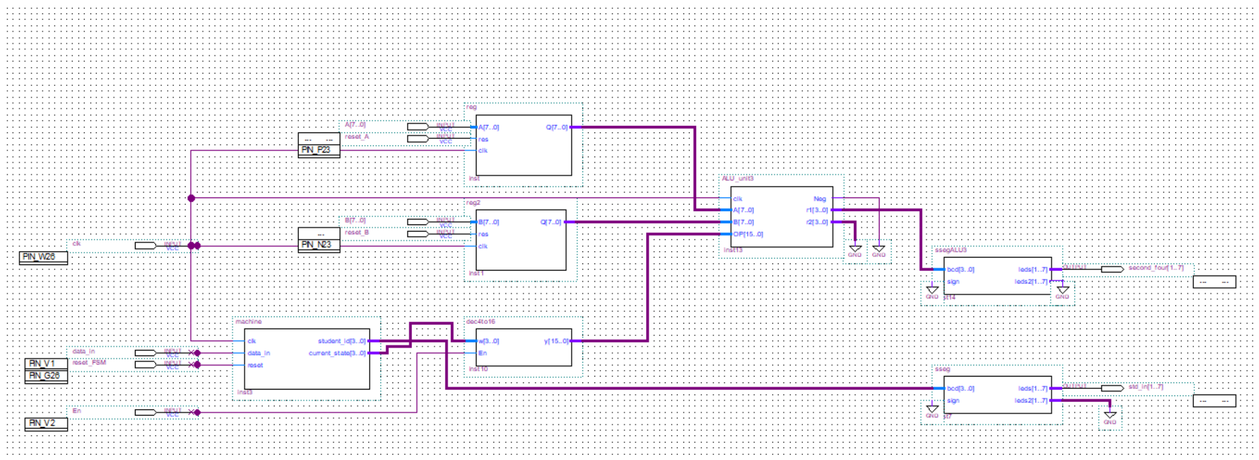
Prompt:

e) For each opcode submitted to the ALU, display 'y' if one of the 2 digits of A is greater than the **student_id** signal value and 'n' otherwise

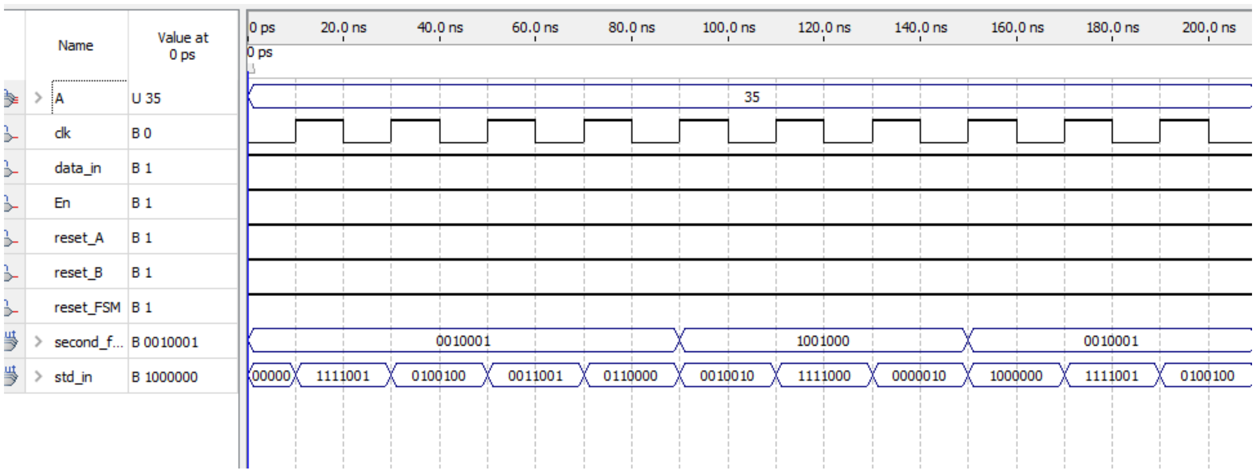
A = 35, use <u>A2 = 5</u> to compare with student_ID

Function #	Operation / Function
1	Compare 5 > 0 = **Y**
2	Compare 5 > 1 = **Y**
3	Compare 5 > 2 = **Y**
4	Compare 5 > 4 = **Y**
5	Compare 5 > 3 = **Y**
6	Compare 5 > 5 = **N**
7	Compare 5 > 7 = **N**
8	Compare 5 > 6 = **N**

- Screenshot of Block Schematic File (BDF):



- Complete Waveform File:



6.0 Conclusion

The goal of this lab was to work through a series of problem sets by designing and implementing an ALU core capable of taking inputs "A" and "B" to perform various boolean operations and modifications, then displaying the results on an FPGA board. As the final lab, it served as a comprehensive assessment of everything covered throughout the semester. The ALU incorporates elements from earlier labs, such as the SSEG display, FSM, and 4:16 decoder, while also integrating new components like the gated-D latch. Successfully completing this lab required a solid understanding of how each individual component functions, how they interact with one another, and the specific role each component plays in solving the problem sets.