Design of a Simple General-Purpose Processor Lab #6

COE 328

Section 11

Course instructor: Arghavan Asad

—

Mohammed Mansuri

50116024

December 4, 2023

# 

[**Introduction**](#_ipqb3k37jiie) **2**

[**Latch 1&2**](#_wd67i02y0f3x) **2**

[**4 to 16 Decoder**](#_s8abuljsuayn) **3**

[**Finite State Machine**](#_lj996ws2v4ct) **5**

[**ALU\_1**](#_k0re3u7bym39) **6**

[**ALU\_2**](#_ld7yy7q8e4i3) **8**

[**ALU\_3**](#_q4vfkcxdubom) **10**

[**Conclusion**](#_ui6xe6v2zghr) **11**

# 

# Introduction

The purpose of this lab is to create a simple functioning processor capable of handling various calculations with 8 bit binary numbers and then outputting the results. A number of components are needed in order to achieve this goal. First, 2 latches are required in order to read and store 2 8-bit binary numbers from the user, the latches must output the binary input of the user and output 0 if the reset signal is 0. Next a finite state machine is required in order to cycle through the user's student number. The FSM will output the student number and its current state on each rising edge of the clock. The FSM will also have a reset that will return the state and number to its original value, as well as a data in that when enabled allows the FSM to cycle through the values on the rising edge of the clock and when off will hold the FSM at the current state. Next the 4 to 16 decoder will take in the current state of the FSM and return the corresponding value while its enable value is 1 if not it will return 0. Finally the 3 different ALUS will take in the 8 bit binary input, perform various operations on them then output the result.

# 

# Latch 1&2

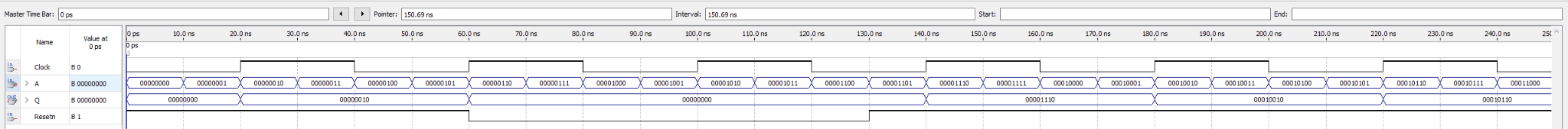
As previously mentioned the latch will read an 8 bit binary input from the user then output that value at the rising edge of the clock. The latch is a rising edge triggered clock therefore the value of latch is updated only on the rising edge of the clock and the last input of the user is stored otherwise. The latch's reset signal occurs when reset is 0 and causes the latch to output 0.

Latch Truth Table:

Assuming A is an arbitrary 8 bit binary number

| Input | Reset | Output |
| --- | --- | --- |
| A | 0 | 00000000 |
| A | 1 | A |

Latch Waveform:



# 4 to 16 Decoder

The 4 to 16 decoder takes the state values from the finite state machine and outputs the according value. The 4 to 16 decoder is made up of two 3 to 8 decoders. One of the inputs is linked to either decoder's enable signal meaning that as one turns on the other will turn off allowing for 16 outputs. The decoder has an enable signal which when on allows the decoder to output values and when off the decoder outputs 0.

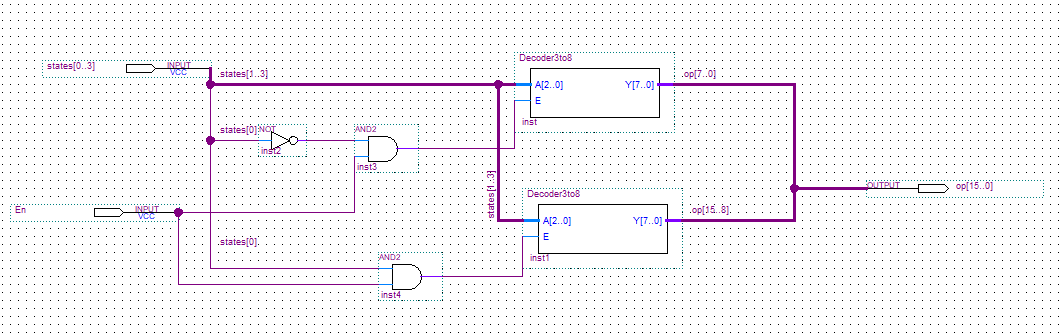
4 to 16 Decoder Truth Table:

E=enable

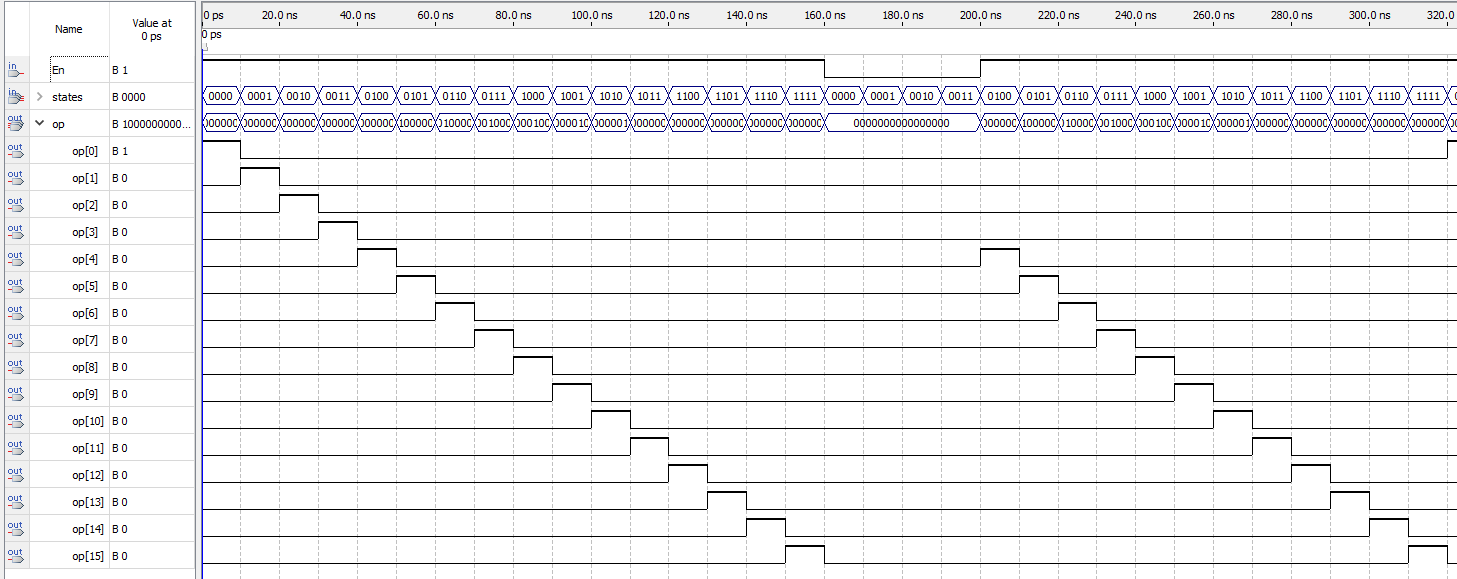
y=input

t=output

| E | y1 | y2 | y3 | y4 | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 | t13 | t14 | t15 | t16 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 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 |

Block diagram

4 to 16 Decoder Waveform



# Finite State Machine

The finite state machine will function as an up counter cycling through states 0 to 8. The FSM will begin at state 0 and continue to count up to 8 sequentially on each rising edge of the clock. The FSM will also print the student number corresponding to its current state, for example at state 0 the FSM must output 5. The FSM also has a data in which allows it to be stopped at a specific state, when data in is 0 the FSM stops cycling and remains at its current state and if data in is 1 the FSM continues cycling on the rising edge of the clock. The FSM also has a reset that when enabled returns the state back to state 0.

FSM Truth Table

| State | Data In=0  Next state | Data In=1  Next State |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 1 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 4 |
| 4 | 4 | 5 |
| 5 | 5 | 6 |
| 6 | 6 | 7 |
| 7 | 7 | 8 |
| 8 | 8 | 0 |

FSM Waveform

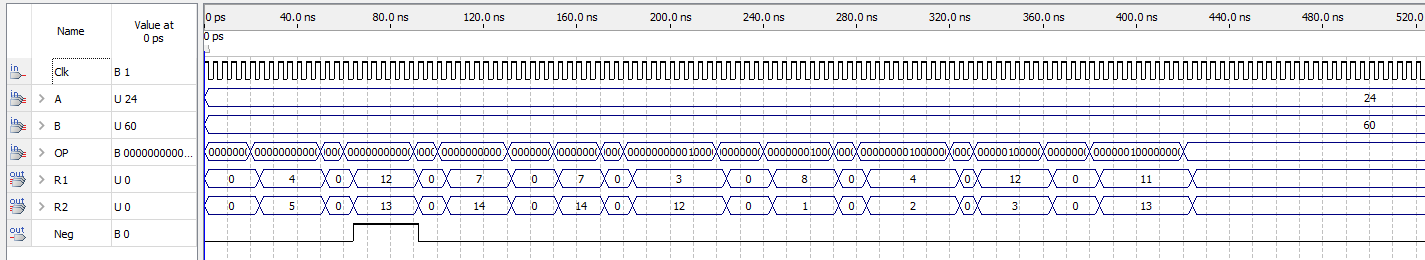
# ALU\_1

The ALU is designed to take 2 8 bit binary numbers and a microcode from a decoder and perform the appropriate operation on the bits and then output the result. The ALU performs these operations on each rising edge of the clock and will produce an 8 bit binary result. Combining all the previous inputs and the alu together allows for 2 8 bit binary numbers to be input through the latch while the FSM provides 8 different states that the decoder will then convert into microcodes for the ALU operations. The inputs required for the alu to function are the clock input which is used to synchronize the outputs, the data in for the FSM and the latch can be connected to 1 input as well as an input for the enable on the 4 to 16 decoder and finally the FSM reset. The outputs required are two 4 bit vectors for the result of ALU, one four bit vector for the states from the FSM and one four bit vector for the student id from states.

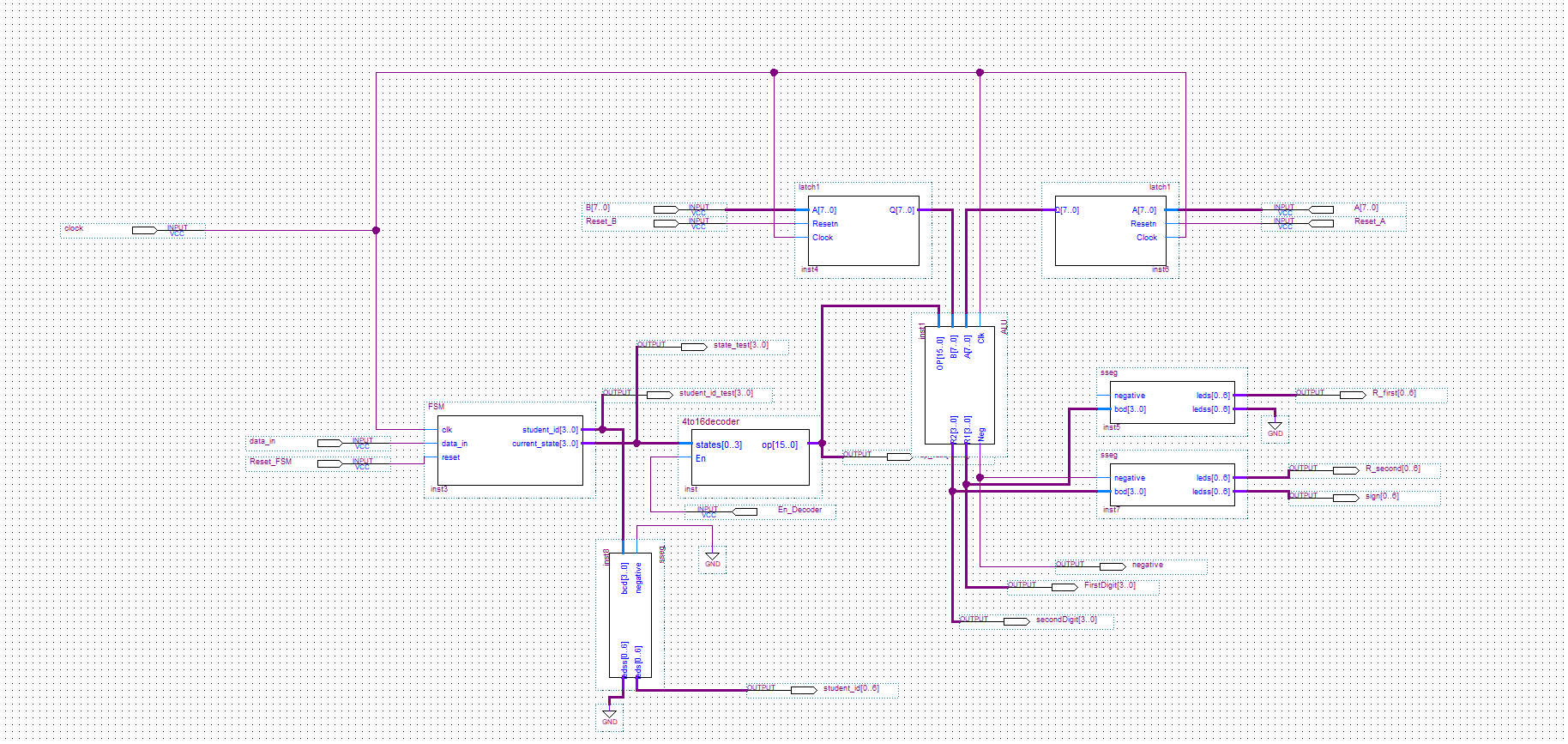
ALU\_1 Microcodes

| Code | Output |
| --- | --- |
| 0000000000000001 | sum(A,B) |
| 0000000000000010 | diff(A,B) |
| 0000000000000100 | NOT A |
| 0000000000001000 | A NAND B |
| 0000000000010000 | A NOR B |
| 0000000000100000 | A AND B |
| 0000000001000000 | A XOR B |
| 0000000010000000 | A OR B |
| 0000000100000000 | A XNOR B |

ALU\_1 Waveform



ALU\_1 Block Diagram:

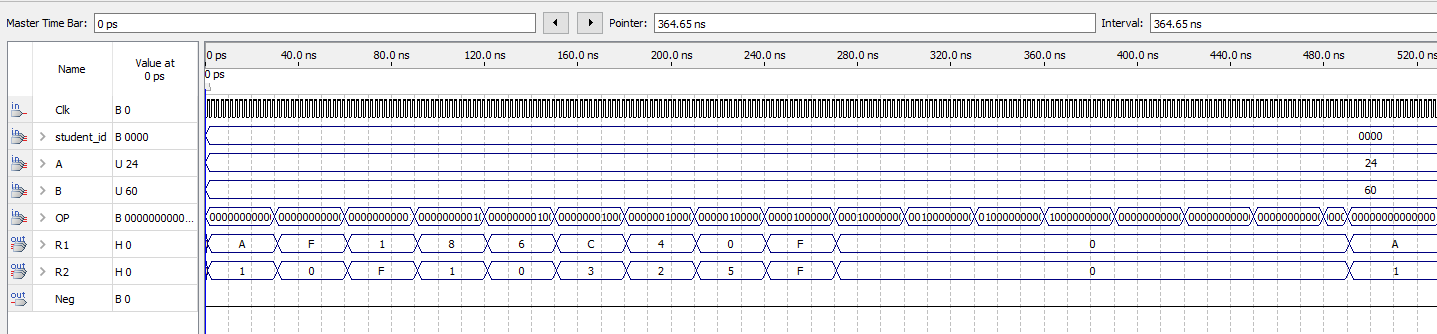


# ALU\_2

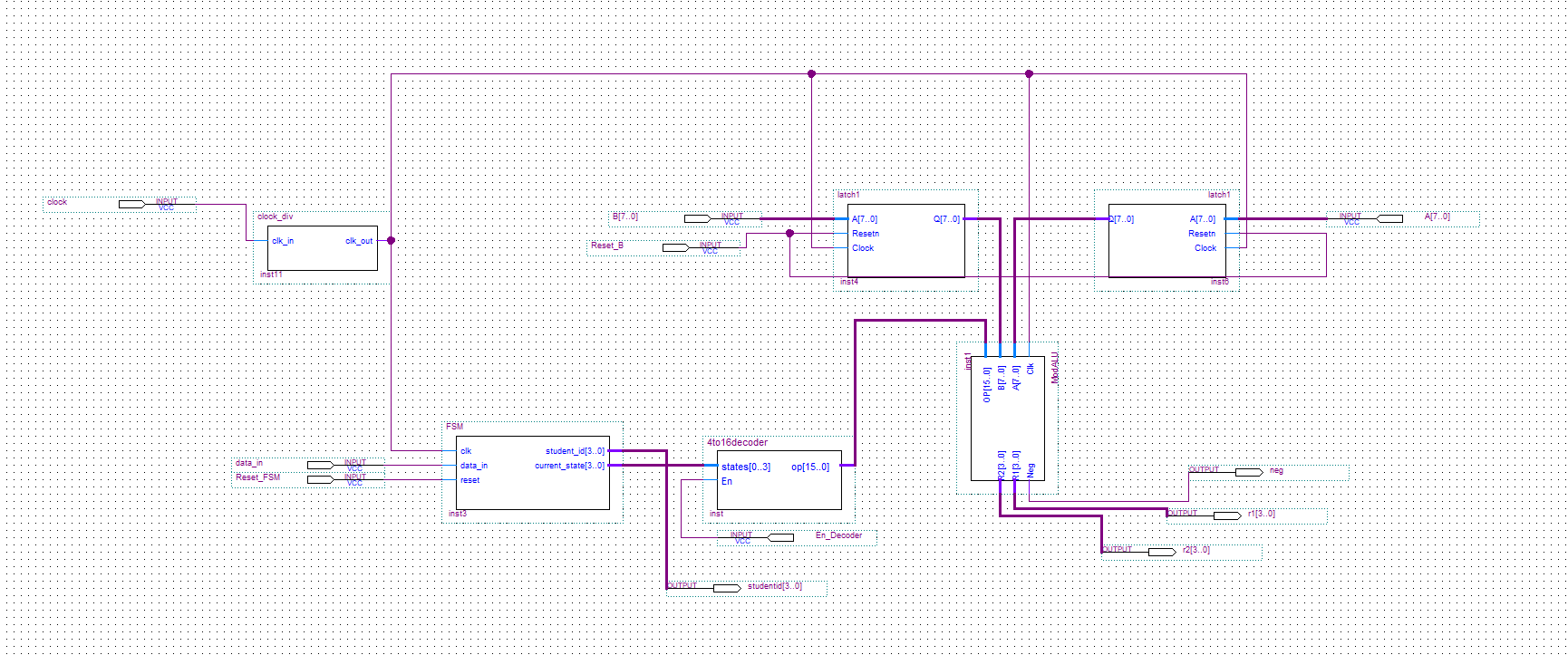
The second ALU follows much of the same implementation as the first ALU, both ALU’s also share all of the same inputs and outputs. The key distinction between them is in the operations they perform. The second ALU is programmed to perform 9 different functions than the first. Similarly all the same switches and key inputs and outputs will remain the same.

ALU\_2 Microcodes

| Code | Output |
| --- | --- |
| 0000000000000001 | Increment A by 2 |
| 0000000000000010 | Shift B 2 bit right input bit = 0 |
| 0000000000000100 | Shift A 4 bits right input bit = 1 |
| 0000000000001000 | Min(A,B) |
| 0000000000010000 | Rotate A 2 bits right |
| 0000000000100000 | Invert bit significance of B |
| 0000000001000000 | A XOR B |
| 0000000010000000 | sum(A,B)-4 |
| 0000000100000000 | Produce all high bits |

ALU\_2 Waveform

ALU\_2 Block Diagram



# 

# 

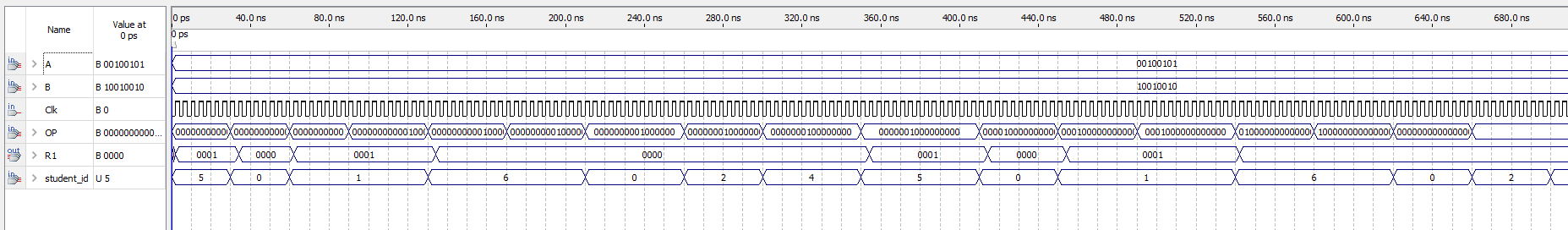
# ALU\_3

The third ALU shares several features with the previous ALUS such as its clock, however its difference is that instead of two 8 bit numbers the ALU takes the student number from the FSM then performs a calculation to determine whether the number is odd or even and outputs a 4 bit value based on the result. The ALU does not require any additional input pins and instead takes the clock input, student id from the FSM as well the latch outputs from the second and first ALU as well as the decoder output from the original ALU. The third ALU outputs 1 4 bit binary number which is displayed on the sseg.

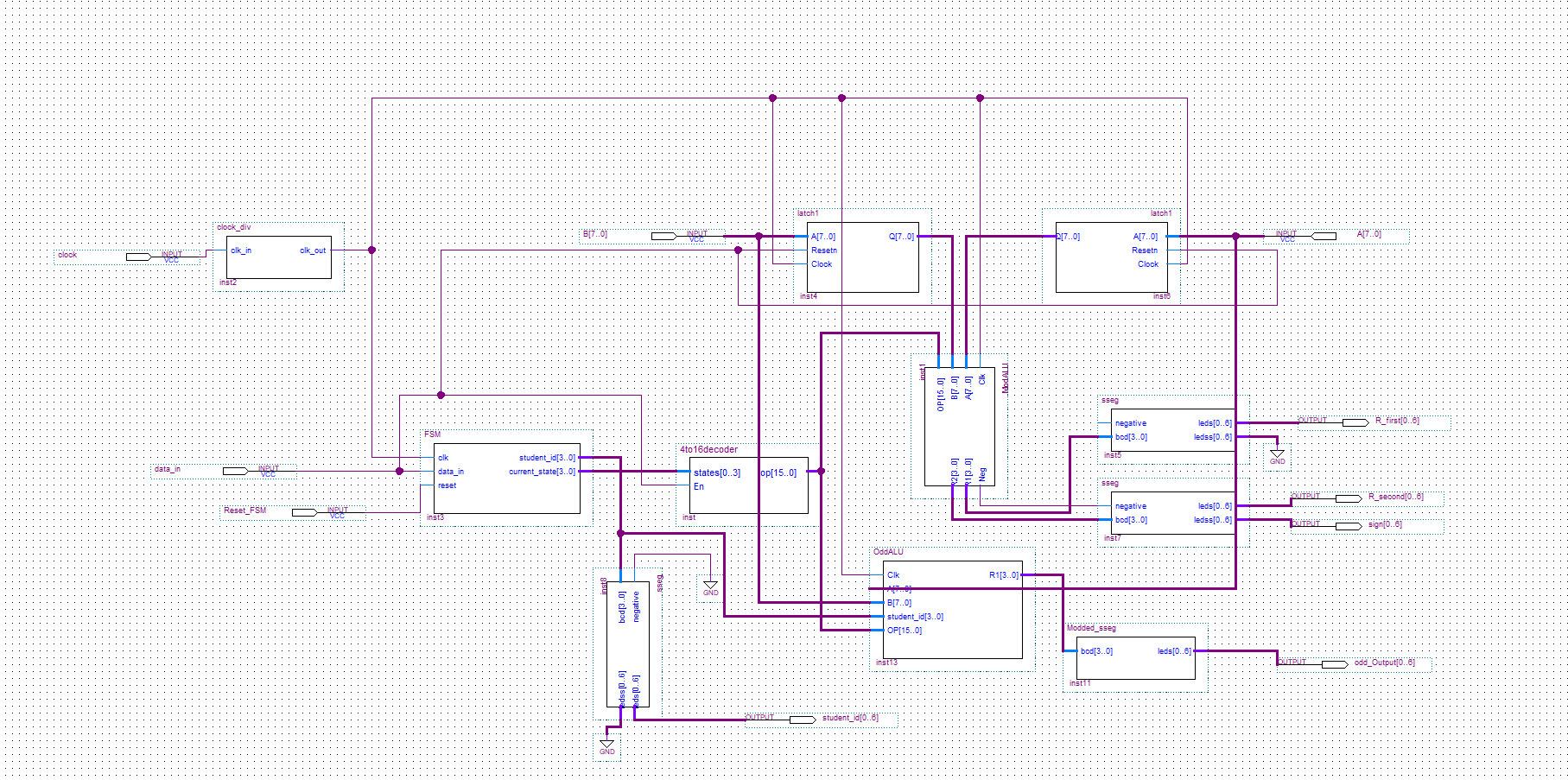
ALU\_3 Microcodes:

| Code | Student Number | Output |
| --- | --- | --- |
| 0000000000000001 | 5 | 0001 (odd) |
| 0000000000000010 | 0 | 0000 (even) |
| 0000000000000100 | 1 | 0001 (odd) |
| 0000000000001000 | 1 | 0001 (odd) |
| 0000000000010000 | 6 | 0000 (even) |
| 0000000000100000 | 6 | 0000 (even) |
| 0000000001000000 | 0 | 0000 (even) |
| 0000000010000000 | 2 | 0000 (even) |
| 0000000100000000 | 4 | 0000 (even) |

ALU\_3 Waveform



ALU\_3 Block Diagram



# 

# Conclusion

In conclusion our lab successfully designed and implemented an 8 bit general purpose processor. The latches stored and output the binary number from the user. The finite state machine cycles through all states with their respective student numbers and the 4 to 16 decoder outputs a microcode for its subsequent state for all 3 ALU’s. The ALU’s were able to perform 8 bit binary operations and output the correct values while the third ALU was able to determine if the student number passed was odd or even. The block diagram and waveforms all demonstrated the correct output values as well as the correct arguments from the ALU. In summary this lab highlighted the significance of each element required to create a function processing unit.