Ryerson University

**Department of Electrical**
**& Computer Engineering**
Faculty of Engineering & Architectural Science

| | |
|---|---|
| **Course Title:** | Digital Systems |
| **Course Number:** | COE328 |
| **Semester/Year (e.g.F2016)** | F2020 |

| | |
|---|---|
| **Instructor:** | Dr. Sedaghat |

| | |
|---|---|
| *Assignment/Lab Number:* | Lab 6 |
| *Assignment/Lab Title:* | Design of a Simple General-Purpose Processor |

| | |
|---|---|
| *Submission Date:* | November 11, 2020 |
| *Due Date:* | November 20, 2020 |

| **Student LAST Name** | **Student FIRST Name** | **Student Number** | **Section** | **Signature*** |
|---|---|---|---|---|
| Patel | Manav | 500967756 | 1 | MPatel |
| | | | | |
| | | | | |

*By signing above, you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report 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: http://www.ryerson.ca/senate/current/pol60.pdf

# TABLE OF CONTENTS

# Introduction

In Lab 6: Design of a Simple General-Purpose Processor, and Arithmetic and Logic Unit (ALU) is to be designed in a VHDL environment on Quartus, and then implemented through Waveforms, to check for accurate results. The unit consists of several parts, that eventually would come together to produce a functional ALU. These parts include the control unit, the bus, the registers and the ALU core. This general-purpose processor will consist of two latches, a decoder unit, multiple SSEGs, a finite state machine, the ALU. The goal of this lab is to create a functional Simple General-Processor using the parts mentioned. Two input values, A and B, were also to be chosen. For this lab, A is chosen to be 77, and B is chosen to be 56

# Components

## Component 1: Latch 1

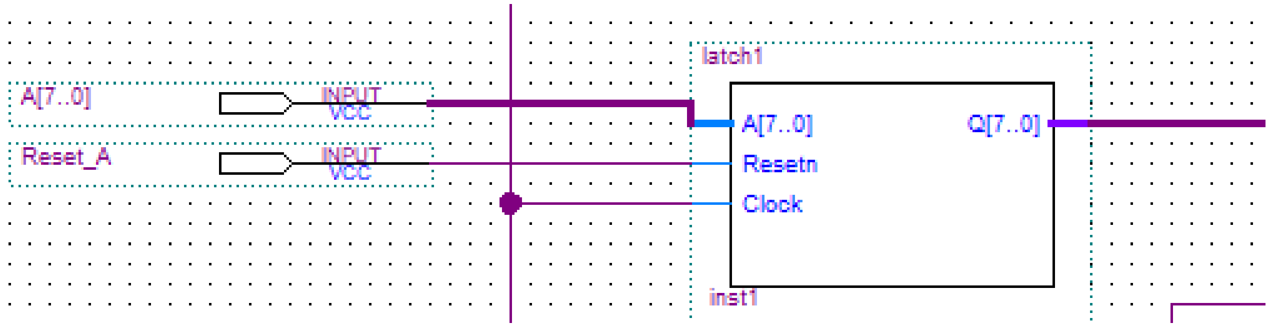
The first component of this general-purpose processor is the Latch. There are two latches that will be used in creating this processor. The latch stores the output of the processor so that the LED can be turned on in the SSEG. This latch is used for the first input value, A. For this part, A = 77. The following table is the truth table for latch 1:

Table 1: Truth Table for Latch 1

| S | R | $Q_a$ | $Q_b$ |
|---|---|---|---|
| 0 | 0 | Memory | |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Not Used | |

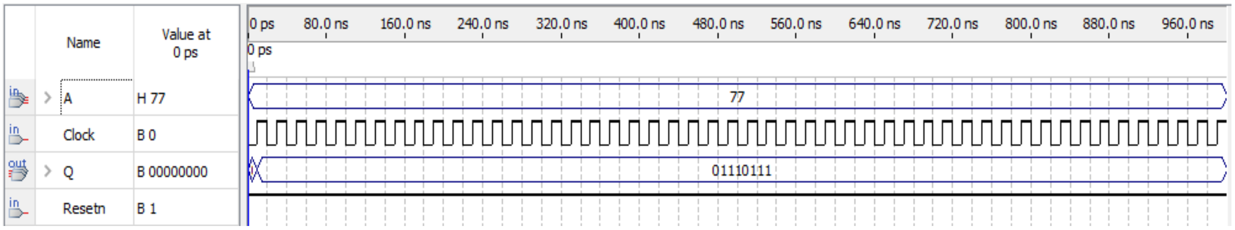
The following shows the block diagram for the first Latch.

Figure 1: Block Diagram for Latch 1



The following shows the waveform for Latch 1.

Figure 2: Waveform for Latch 1



| Name | Value at 0 ps |
|---|---|
| A | H 77 |
| Clock | B 0 |
| Q | B 00000000 |
| Resetn | B 1 |

## Component 2: Latch 2

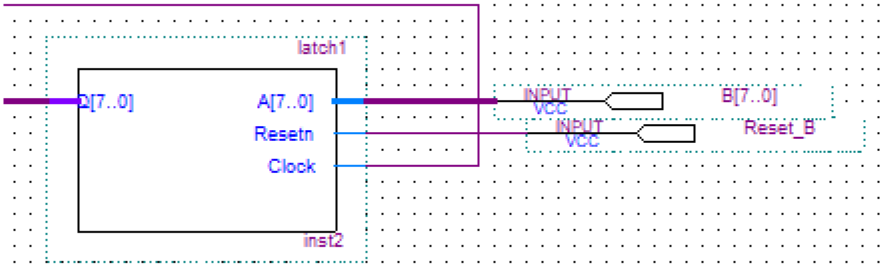
The second component of this general-purpose processor is another latch. This is the second latch used to create this processor. This latch is used for the second input value, B. For this part, B = 56. The following table is the truth table for latch 2:

Table 1: Truth Table for Latch 2:

| S | R | $Q_a$ | $Q_b$ |
|---|---|---|---|
| 0 | 0 | Memory | |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Not Used | |

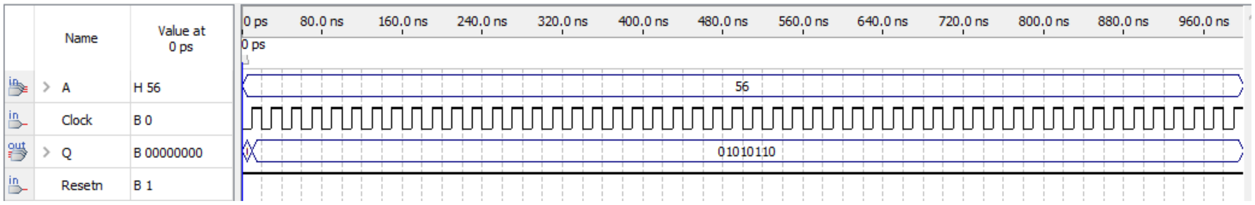
The following shows the Block Diagram for Latch 2:

Figure 3: Block Diagram for Latch 2



The following shows the waveform for Latch 2.

Figure 4: Waveform for Latch 2



| | Name | Value at 0 ps |
|---|---|---|
| in | A | H 56 |
| in | Clock | B 0 |
| out | Q | B 00000000 |
| in | Resetn | B 1 |

## Component 3: 4:16 Decoder

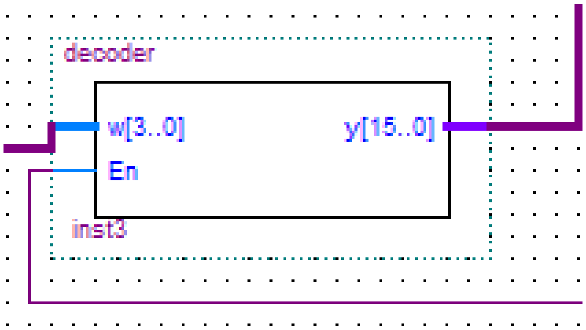
The third component of this general-purpose processor is the 4:16 decoder. The function of a decoder is to change a sort of instructions into signals. It is tasked to receive the signal from the FSM, decode it and send it to the ALU. The following is the truth table for a 4:16 decoder:

Table 3: Truth Table for 4:16 Decoder

| En | X4 | X3 | X2 | X1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

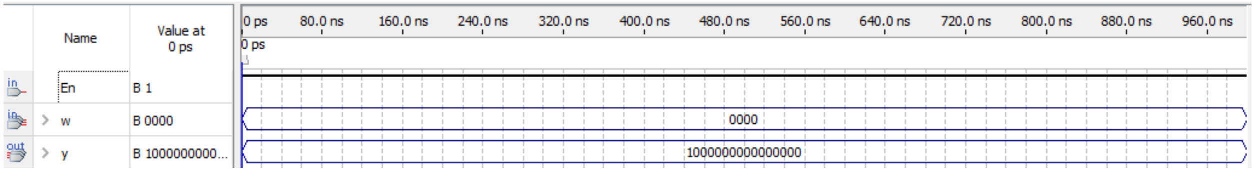
The following is the Block Diagram for the 4:16 Decoder:

Figure 5: 4:16 Decoder



The following is the waveform for the 4:16 Decoder:

Figure 6: Waveform for 4:16 Decoder



| Name | Value at 0 ps |
|---|---|
| En | B 1 |
| w | B 0000 |
| y | B 1000000000... |

## Component 4: Finite State Machine

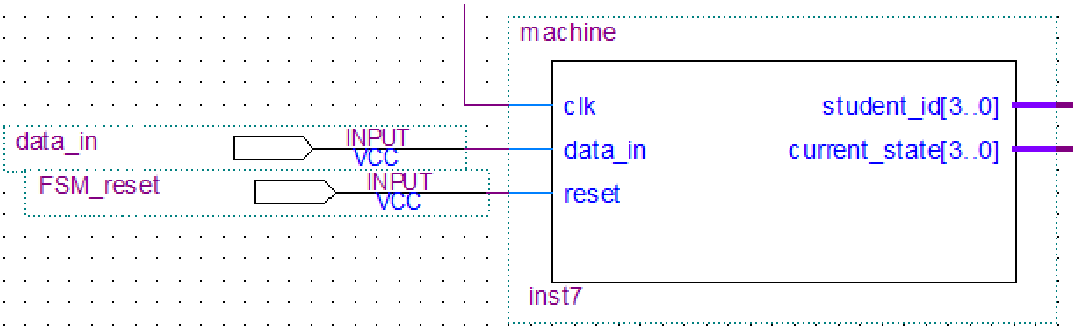
The fourth component of this general-purpose processor is the Finite State Machine (FSM). An FSM is a machine made of one or multiple states, where only one state can be active at a time. If it is on, it would go to the next state, else if it is off, it would stay in the same state or go to the previous state. The following is the state diagram for a Finite State Machine:

Table 4: State Diagram for Finite State Machine

| Present State | Next State | | Output |
|---|---|---|---|
| | w = 0 | w = 1 | z |
| A | A | B | 0 |
| B | A | C | 0 |
| C | A | C | 1 |

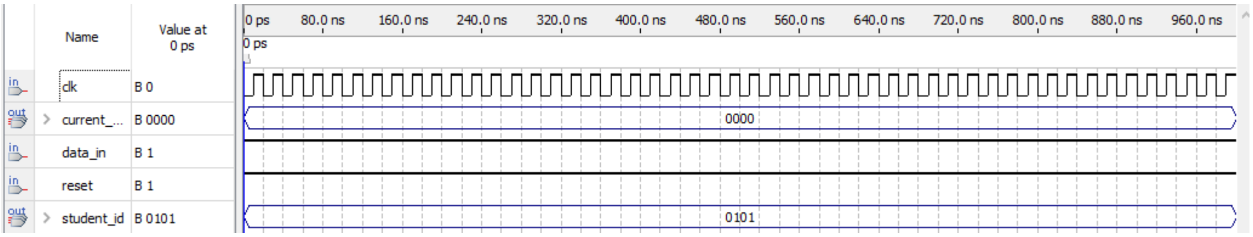
The following is the Block Diagram for a Finite State Machine:

Figure 7: Block Diagram for FSM



The following is the waveform for an FSM:

Figure 8: Waveform for FSM



| Name | Value at 0 ps |
|---|---|
| clk | B 0 |
| current_... | B 0000 |
| data_in | B 1 |
| reset | B 1 |
| student_id | B 0101 |

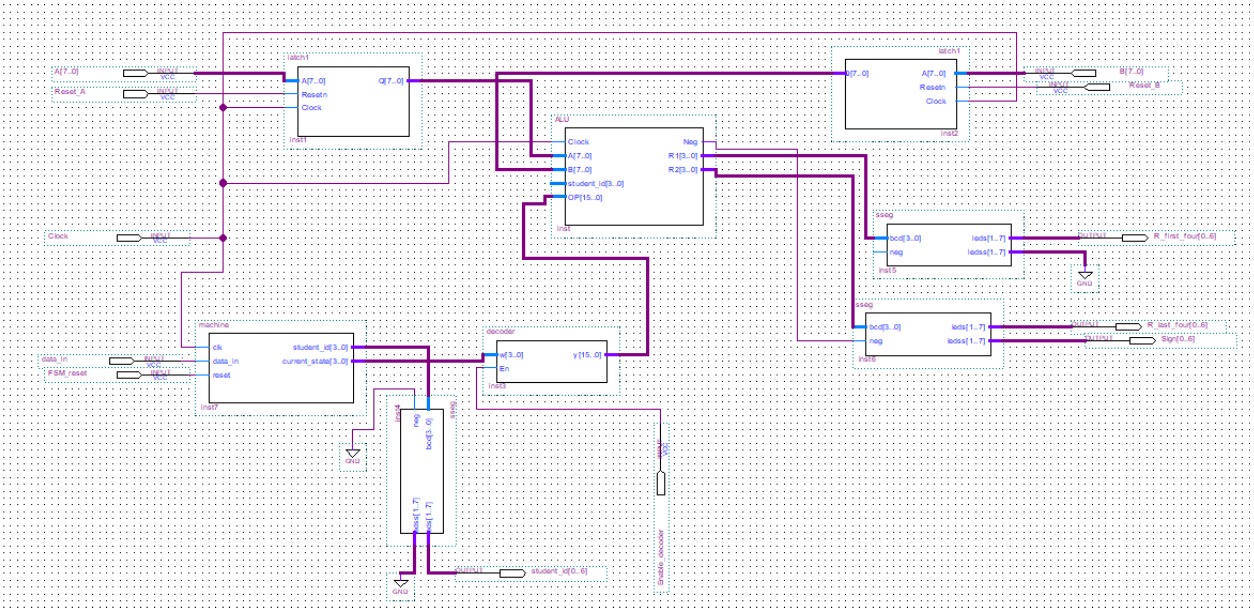
# Arithmetic and Logic Unit

## Problem Set 1

The problem is to initially implement the first design of the General Processor Unit. Thee FSM output will follow an up-counting pattern which will send a signal to the ALU core for the operation. This will go then to the Result, which shows the output. The FSM is assigned nine microcodes, where each microcode as a specific function it performs. At the output, all functions will be displayed.

The following shows the Block Schematic for Problem Set 1:

Figure 9: Block Schematic for Problem 1



Here, the FSM is where the states are shown. If data_in for each number ins the student id is ‘0’, it will stay in the same state. Else if data_in is ‘1’, it will go onto the next state. Also, each number is assigned to a specific state. For example, when student_id is equal to ‘0101’, current_state would be ‘0000’. The latches will store the output of the processor for the initial A and B inputs, and output it through the SSEG, where the LEDs will be on or off. The ALU is where the instructions to the microcodes will be processed. For instance, if the microcode is ‘0000000000000001’, it will sum A and B, and will output the result. Finally, the SSEGs will receive information from the ALU, and will provide an output by turning on or off for each input.

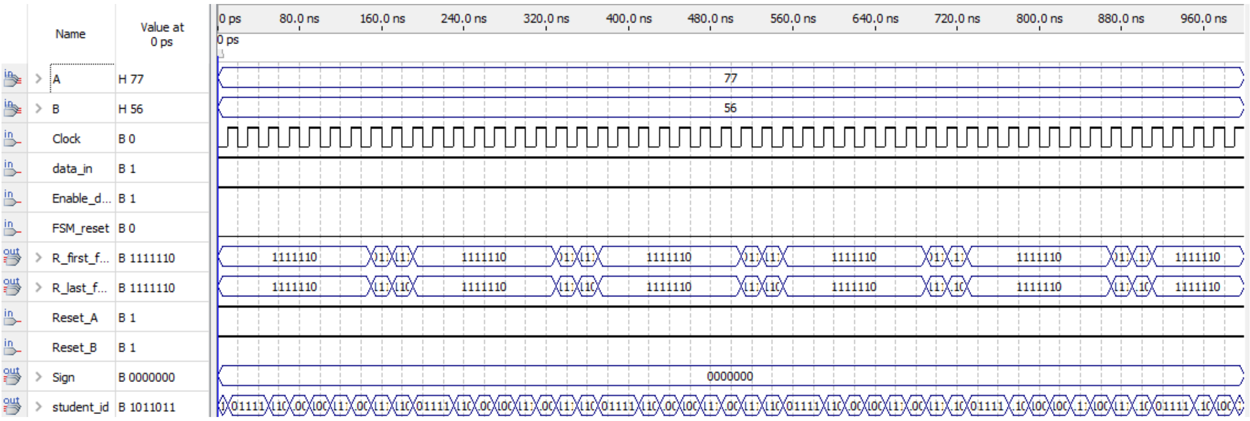
The following is the table for the microcodes for Problem Set 1:

Table 5: ALU Core Operations for Problem 1 with Output Code

| Function # | Microcode | Output Code |
|---|---|---|
| 1 | 0000000000000001 | Result <= Reg1 + Reg2; |
| 2 | 0000000000000010 | Result <= Reg1 - Reg2;<br>if Reg2 > Reg1 Then<br>Neg <= '1';<br>End if; |
| 3 | 0000000000000100 | Result <= (NOT(Reg1)); |
| 4 | 0000000000001000 | Result <= (NOT(A OR B)); |
| 5 | 0000000000010000 | Result <= (NOT(A AND B)); |
| 6 | 0000000000100000 | Result <= (A OR B); |
| 7 | 0000000001000000 | Result <= (A XOR B); |
| 8 | 0000000010000000 | Result <= (A AND B); |
| 9 | 0000000100000000 | Result <= (NOT (A XOR B)); |
|  | others | 00000000 |

The following is the waveform for Problem Set 1:

Figure 10: Waveform for Problem Set 1



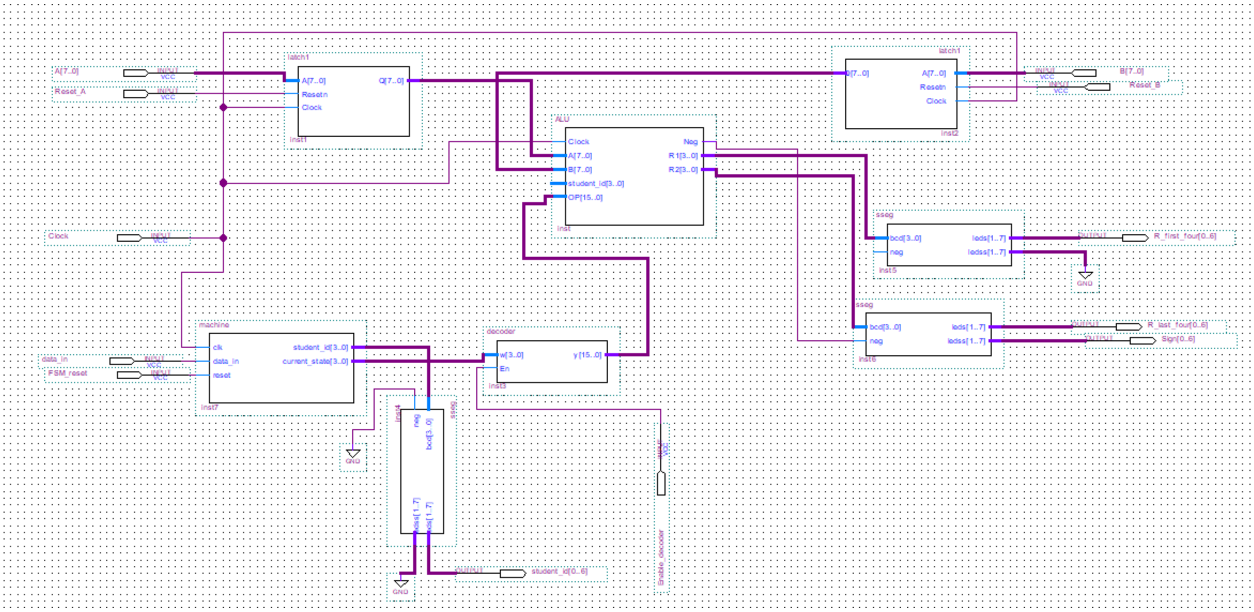
| Name | Value at 0 ps |
|---|---|
| A | H 77 |
| B | H 56 |
| Clock | B 0 |
| data_in | B 1 |
| Enable_d... | B 1 |
| FSM_reset | B 0 |
| R_first_f... | B 1111110 |
| R_last_f... | B 1111110 |
| Reset_A | B 1 |
| Reset_B | B 1 |
| Sign | B 0000000 |
| student_id | B 1011011 |

## Problem Set 2

Theis problem is to modify the core of the ALU based on operations and functions provided. The Block Diagram is the same as the block diagram for Problem Set 2, and therefore will perform in the same manner.

The following shows the Block Schematic for Problem Set 1:

Figure 11: Block Schematic for Problem 2



Other than the ALU, all components will function the same way as Problem Set 1, with the same set of instructions given. The ALU is where the instructions to the microcodes will be different in this Problem Set. It will follow the instructions to the functions from Modified ALU core operations. Option C is selected for this portion of the lab. For instance, if the microcode is ‘0000000000000001’, it will produce the difference between A and B. The SSEGs will receive information from the ALU and will provide an output by turning on or off for each input.

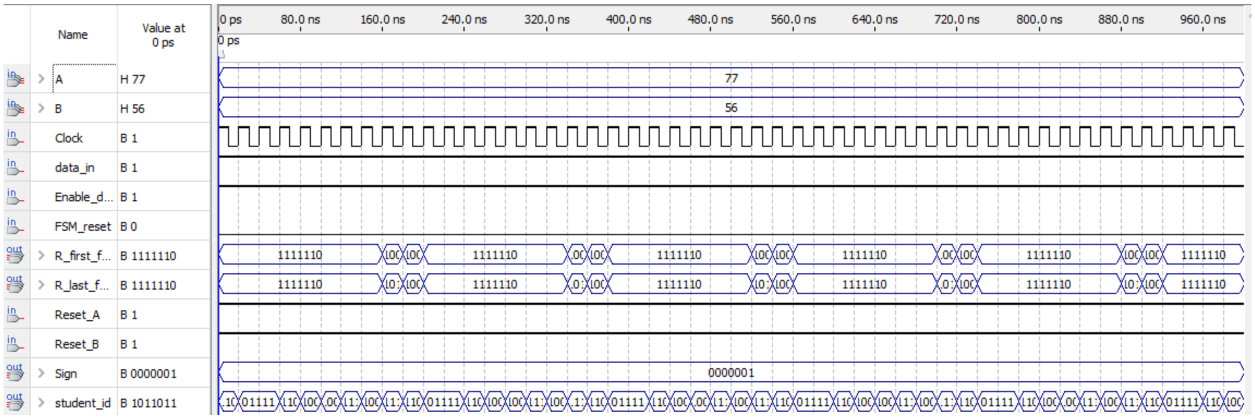
The following is the table for the microcodes for Problem Set 2:

Table 6: ALU Core Operations for Problem 2 with Output Code

| Function # | Microcode | Output Code |
|---|---|---|
| 1 | 0000000000000001 | if ( A < B )<br>then<br>neg <= '1';<br>Result <= B - A;<br>else<br>neg <= '0';<br>Result <= A - B;<br>end if; |
| 2 | 0000000000000010 | neg <= '0';<br>Result <= (1 + not(B)); |
| 3 | 0000000000000100 | neg <= '0' ;<br>temp(0) <= B(0);<br>temp(1) <= B(1);<br>temp(2) <= B(2);<br>temp(3) <= B(3);<br>temp(4) <= A(4);<br>temp(5) <= A(5);<br>temp(6) <= A(6);<br>temp(7) <= A(7);<br>Result <= temp; |
| 4 | 0000000000001000 | neg <= '0' ;<br>Result <= null ; |
| 5 | 0000000000010000 | neg <= '0';<br>Result <= B - "0101" ; |
| 6 | 0000000000100000 | neg <= '0' ;<br>temp(0) <= A(7);<br>temp(1) <= A(6);<br>temp(2) <= A(5);<br>temp(3) <= A(4);<br>temp(4) <= A(3);<br>temp(5) <= A(2);<br>temp(6) <= A(1);<br>temp(7) <= A(0);<br>Result <= temp; |
| 7 | 0000000001000000 | neg <= '0' ;<br>Result <= B sll 3 ; |
| 8 | 0000000010000000 | neg <= '0';<br>Result <= A - "011" ; |
| 9 | 0000000100000000 | neg <= '0' ;<br>Result <= not(B); |
|  | others | 00000000 |

The following is the waveform for Problem Set 1:

Figure 12: Waveform for Problem Set 2



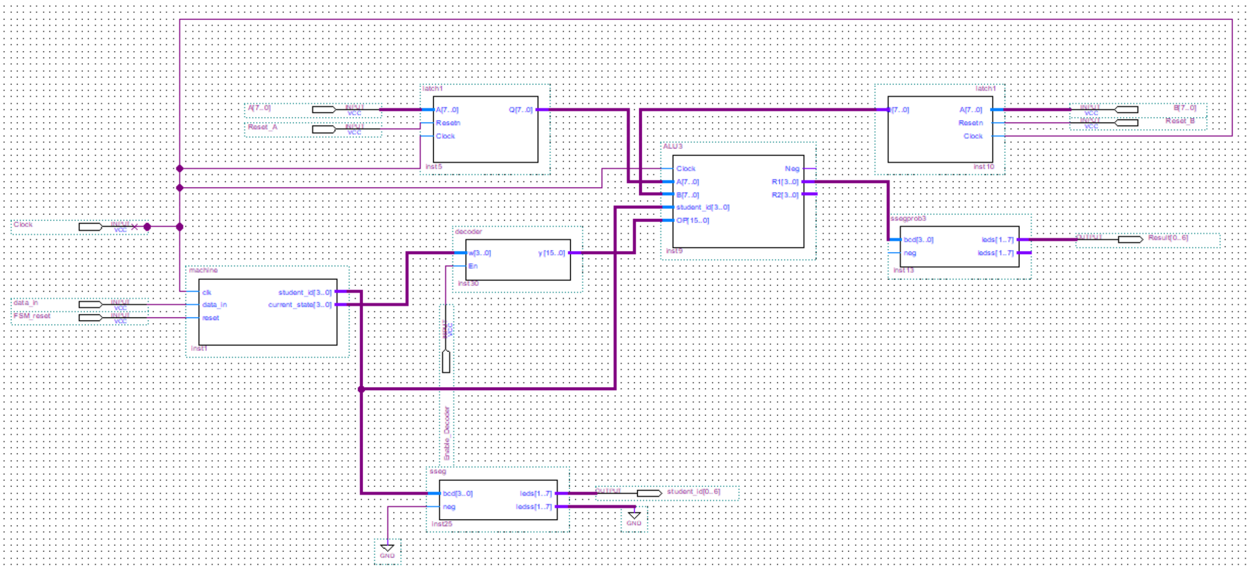
| Name | Value at 0 ps |
|---|---|
| A | H 77 |
| B | H 56 |
| Clock | B 1 |
| data_in | B 1 |
| Enable_d... | B 1 |
| FSM_reset | B 0 |
| R_first_f... | B 1111110 |
| R_last_f... | B 1111110 |
| Reset_A | B 1 |
| Reset_B | B 1 |
| Sign | B 0000001 |
| student_id | B 1011011 |

## Problem Set 3

This problem is to use the student_id output from the FSM component of the control unit and implement functionalities that are described.

The following shows the Block Schematic for Problem Set 3:

Figure 13: Block Schematic for Problem 3



Here, the student_id output from the FSM will be used. Choosing option B for this part of the lab, the task is to display ‘y’ for each student_id output if it is even and display ‘n’ if the output is odd, for each microcode instruction. The ALU will take all instructions for each microcode and have outputs based on what each function is asking. For example, for the microcode “0000000000000001’, if the first value of the student number is even, the SSEG will display Y, in binary. Else if the first value is odd, the SSEG will display N, in binary. These same set of instructions were repeated for all eight functions. This will continue for all nine functions.

The following is the table for the microcodes for Problem Set 3:

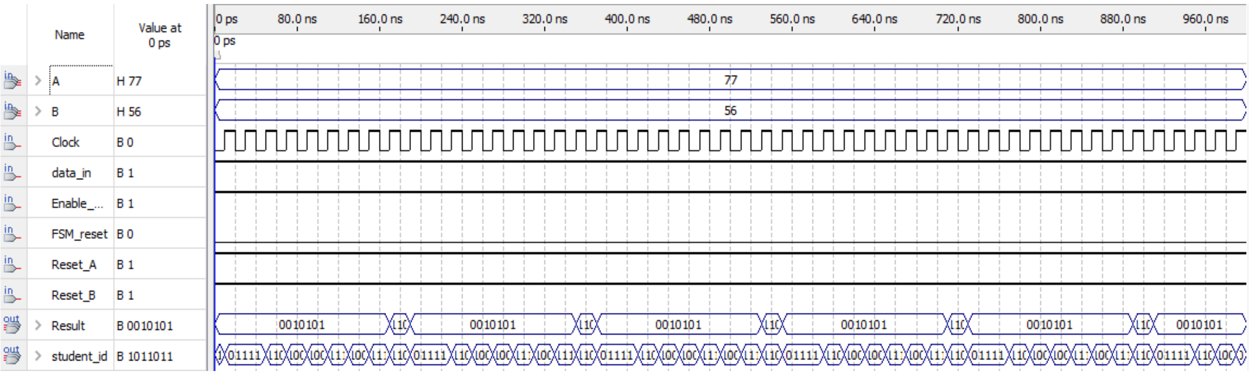
Table 7: ALU Core Operations for Problem 3 with Output Code

| Function # | Microcode | Output Code |
|---|---|---|
| 1 | 0000000000000001 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 2 | 0000000000000010 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 3 | 0000000000000100 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 4 | 0000000000001000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 5 | 0000000000010000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 6 | 0000000000100000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 7 | 0000000001000000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |

| | | |
|---|---|---|
| 8 | 0000000010000000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| 9 | 0000000100000000 | if ( student_id mod 2 = 0 )<br>then<br>Result <= "1111";<br>else<br>Result <= "0000";<br>end if; |
| | others | 00000000 |

The following is the waveform for Problem Set 1:

Figure 14: Waveform for Problem Set 3



| Name | Value at 0 ps |
|---|---|
| A | H 77 |
| B | H 56 |
| Clock | B 0 |
| data_in | B 1 |
| Enable_... | B 1 |
| FSM_reset | B 0 |
| Reset_A | B 1 |
| Reset_B | B 1 |
| Result | B 0010101 |
| student_id | B 1011011 |

## Conclusion

For this lab, the goal is to create a General-Purpose processor and an ALU using several components, which would come together at the end to produce the above. For each component, a VHDL code was created.  Following that, each component had a block diagram created and was implemented onto a BDF schematic, which was connected to provide outputs from inputs. After, a waveform was created for each problem set to analyze and test results. Overall, the results from the lab were satisfying, as results were accurate and all block diagrams along the VHDL code compiled without presenting any errors. Overall, from the results of this lab, since everything worked properly, the outcome is positive.

# References

"Lab 6 Report Outline." Ryerson University, Toronto, ON.

"Lab 6: Design of a Simple General-Purpose Processor." Ryerson University, Toronto, ON.

S. D. Brown and Z. G. Vranesic, *Fundamentals of digital logic with VHDL design*. Chennai: McGraw-Hill Education (India) Private Limited, 2012.