

|  |  |
| --- | --- |
| Course Number | COE 328 |
| Course Title | Digital Circuits |
| Semester/Year | Fall 2021 |
| Instructor | Dr. Reza Sedaghat |
| TA Name | Sajjad Rostami Sani |

|  |  |
| --- | --- |
| Lab/Tutorial Report No. | 6 |
|  |  |
| Report Title | |  | | --- | | *Simple General-Purpose Processor* | |

|  |  |
| --- | --- |
| Section No. | 03 |
| Submission Date | 19-Dec-2021 |
| Due Date | 20-Dec-2021 |

|  |  |  |
| --- | --- | --- |
| Student Name | Student ID | Signature\* |
| Ahmad El-Gohary | 501011852 |  |

\*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:

http://www.ryerson.ca/content/dam/senate/policies/pol60.pdf

## Contents

[Contents 2](#_Toc90842658)

[Introduction 3](#_Toc90842659)

[Components 3](#_Toc90842660)

[ Basic latch 3](#_Toc90842661)

[ 4:16 Decoder 5](#_Toc90842662)

[ Finite State Machine (FSM) 7](#_Toc90842663)

[ 7 Segment display 13](#_Toc90842664)

[ Modified 7 Segment display (used in part 3 method 1): 17](#_Toc90842665)

[ Modified 7 Segment display (used in part 3 method2): 20](#_Toc90842666)

[Arithmetic logical unit (ALU) 23](#_Toc90842667)

[ ALU for Part 1 23](#_Toc90842668)

[ ALU for part 2: 26](#_Toc90842669)

[ ALU for part 3 (method 2): 30](#_Toc90842670)

[Circuit Diagrams 33](#_Toc90842671)

[References 35](#_Toc90842672)

# Introduction

This lab was to design and construct a simple microprocessor that does an operation to 2 binary numbers based on the state of the Finite State Machine.

# Components

## Basic latch

The basic latch is the main component of the storage unit which takes in a binary number as an input and gives it out as an output for each cycle.

The circuit uses 2 basic latches 1 for each binary number.

Code for latch:

Text

Description automatically generated

Component’s Circuit Diagram:

Text

Description automatically generated with low confidence

Waveform for Latch:

Chart, box and whisker chart

Description automatically generated

Truth table for latch:

A picture containing text, clock

Description automatically generated

## 4:16 Decoder

The 4:16 decoder takes in the 4-bit state output of the FSM and gives out a unique 16-bit output for each state.

Code for 4:16 decoder:

A picture containing text

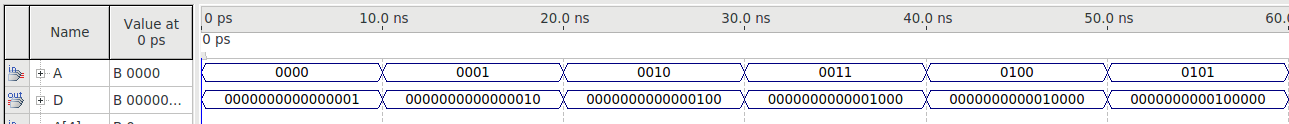
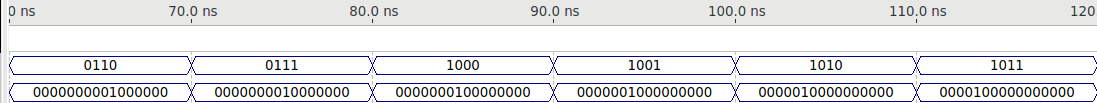
Description automatically generated

Component’s Circuit Diagram:

Text

Description automatically generated

Waveform for 4:16 decoder:

Diagram

Description automatically generated

Truth Table for 4:16 decoder:

Table

Description automatically generated

## Finite State Machine (FSM)

The FSM is a Moore machine cycles through different states from S0 to S8, when the data input is 1 it moves from 1 state to the next and it gives a respective digit of my student ID for example the third sate would give the third digit of my student ID.

Code for FSM:Text

Description automatically generated Graphical user interface, application

Description automatically generated Graphical user interface, text, application

Description automatically generated Text

Description automatically generated with medium confidence

Component’s Circuit Diagram:

Text

Description automatically generated

Waveform for FSM:

A picture containing diagram

Description automatically generated

Truth Table for FSM:

|  |  |  |  |
| --- | --- | --- | --- |
| Present State | Next State | | Output  student\_id |
| data\_in=0 | data\_in=1 |
| 0000 | 0000 | 0001 | 0101 |
| 0001 | 0001 | 0010 | 0000 |
| 0010 | 0010 | 0011 | 0001 |
| 0011 | 0011 | 0100 | 0000 |
| 0100 | 0100 | 0101 | 0001 |
| 0101 | 0101 | 0110 | 0001 |
| 0110 | 0110 | 0111 | 1000 |
| 0111 | 0111 | 1000 | 0101 |
| 1000 | 1000 | 0000 | 0010 |

## 7 Segment display

The 7-segment display shows numerical output in hexadecimal starting from 0 to F, it can show up to 15 numbers on 1 7-segment display.

7 Segment display Code:A picture containing text

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence

Component’s Circuit Diagram:

Graphical user interface, text, application

Description automatically generated

Waveform for 7-Segment display:

Graphical user interface, calendar

Description automatically generated

Table

Description automatically generated

Truth Table for 7-segment display:

|  |  |
| --- | --- |
| Input | leds |
| 0000 | 1111110 |
| 0001 | 0110000 |
| 0010 | 1101101 |
| 0011 | 1111001 |
| 0100 | 0110011 |
| 0101 | 1011011 |
| 0110 | 1011111 |
| 0111 | 1110000 |
| 1000 | 1111111 |
| 1001 | 1110011 |
| 1010 | 1110111 |
| 1011 | 0011111 |
| 1100 | 1001110 |
| 1101 | 0111101 |
| 1110 | 1101111 |
| 1111 | 1000111 |

|  |  |
| --- | --- |
| neg | Ledss |
| 0 | 0000000 |
| 1 | 0000001 |

## Modified 7 Segment display (used in part 3 method 1):

The 7-Segment checks if the number is even or odd and no changes needed for the ALU.

Code for 7-segment display:

Text

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Component’s Circuit Diagram:

Text

Description automatically generated with medium confidence

Waveform for modified 7-segment display:

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Truth Table for 7-segment display:

|  |  |  |
| --- | --- | --- |
| Input | leds | even |
| 0000 | 1111110 | 0111011 |
| 0001 | 0110000 | 0010101 |
| 0010 | 1101101 | 0111011 |
| 0011 | 1111001 | 0010101 |
| 0100 | 0110011 | 0111011 |
| 0101 | 1011011 | 0010101 |
| 0110 | 1011111 | 0111011 |
| 0111 | 1110000 | 0010101 |
| 1000 | 1111111 | 0111011 |
| 1001 | 1110011 | 0010101 |
| 1010 | 1110111 | 0111011 |
| 1011 | 0011111 | 0010101 |
| 1100 | 1001110 | 0111011 |
| 1101 | 0111101 | 0010101 |
| 1110 | 1101111 | 0111011 |
| 1111 | 1000111 | 0010101 |

## Modified 7 Segment display (used in part 3 method2):



ALU checks if the number is even or odd.

Code for 7-Segment display:

A picture containing text

Description automatically generated

**Text

Description automatically generated with medium confidence**

Waveform for 7-segment display:

Graphical user interface, application, table, Excel

Description automatically generatedGraphical user interface, application, table

Description automatically generated

Truth Table for 7-segment display:

|  |  |
| --- | --- |
| Input | leds |
| 0000 | 1111110 |
| 0001 | 0110000 |
| 0010 | 1101101 |
| 0011 | 1111001 |
| 0100 | 0110011 |
| 0101 | 1011011 |
| 0110 | 1011111 |
| 0111 | 1110000 |
| 1000 | 1111111 |
| 1001 | 1110011 |
| 1010 | 1110111 |
| 1011 | 0011111 |
| 1100 | 1001110 |
| 1101 | 0111101 |
| 1110 | 1101111 |
| 1111 | 1000111 |

|  |  |
| --- | --- |
| E | Even |
| 0 | 0111011 |
| 1 | 0010101 |

Component’s Circuit Diagram:

Table

Description automatically generated

# Arithmetic logical unit (ALU)

## ALU for Part 1

The ALU is the processor that decides what operation to do on the inputs based on the state of the FSM. It has 5 inputs:

1. Clock
2. A
3. B
4. student\_id
5. OP

Input A and B are binary numbers which are the input of the storage unit as long as the data\_in is 1. The clock input alternates between 1 and 0, when the clock input changes from 0 to 1 (rising edge) the ALU checks the value OP input (which is the output of the decoder) and runs the switch statement matching the value of the OP input to the case and doing its respective operation on A and B according to the table in the lab manual. The student\_id input has no purpose in part. Table

Description automatically generated

The ALU has 3 outputs:

1. Neg
2. R1
3. R2

Neg outputs 1 if the number is negative and lights up the g led on the 7-segment display

R1 and R2 are 4-bit outputs when combined gives an 8-bit output each one goes to a 7-segment display to display the binary output but in hexadecimal.

Code for ALU part 1: Text

Description automatically generated

Text

Description automatically generated

Component’s Circuit Diagram:

Text

Description automatically generated

## ALU for part 2:

The ALU is the processor that decides what operation to do on the inputs based on the state of the FSM. It has 5 inputs:

1. Clock
2. A
3. B
4. student\_id
5. OP

Input A and B are binary numbers which are the input of the storage unit as long as the data\_in is 1. The clock input alternates between 1 and 0, when the clock input changes from 0 to 1 (rising edge) the ALU checks the value OP input (which is the output of the decoder) and runs the switch statement matching the value of the OP input to the case and doing its respective operation on A and B according to the table in the lab manual. The student\_id input has no purpose in part.

|  |  |
| --- | --- |
| Function # | Operation / Function |
| 1 | Swap the lower and upper 4 bits of A |
| 2 | Produce the result of ORing A and B |
| 3 | Decrement **B** by 5 |
| 4 | Invert all bits of A |
| 5 | Invert the bit-significance order of **A** |
| 6 | Find the greater value of **A** and **B** and produce the results (Max(**A**,**B**) ) |
| 7 | Produce the difference between **A** and **B** |
| 8 | Produce the result of XNORing **A** and **B** |
| 9 | Rotate **B** to left by three bits (ROL) |

For the third function (difference) the output would normally be in 2’s compliment however in my approach I decided to make the output be in decimal for simplicity, so instead of giving -FF for -3 I decided to make it -3, by adding an if statement.

Code for ALU part 2:Text

Description automatically generated A picture containing text

Description automatically generated Graphical user interface, text

Description automatically generated with medium confidence

Component’s Circuit Diagram:

Text

Description automatically generated

## ALU for part 3 (method 2):

In this part the student\_id input is used and the ALU checks if its LSB is 1 or 0 since 1 means it is an odd number and 0 means it is an even number. The ALU also has a new output E that sends 1 if the number is odd and 0 if even which then goes into the 7-segment display and goes through an if statement to show either y for even or n for odd.

Code for ALU: Text

Description automatically generated A picture containing text

Description automatically generated A picture containing text

Description automatically generated A picture containing table

Description automatically generated Graphical user interface, text, application

Description automatically generated

Component’s Circuit Diagram:

Text

Description automatically generated

# Circuit Diagrams

Circuit for part 1:

**Diagram, schematic

Description automatically generated**

Waveform output for part 1:

A picture containing timeline

Description automatically generated

A picture containing calendar

Description automatically generated

Circuit for part 2 and 3(method1):

Diagram

Description automatically generated

Circuit for part 2 and 3(method2):

Diagram

Description automatically generated with medium confidence

Waveform output for part 2 and 3:

A picture containing timeline

Description automatically generated Table

Description automatically generated

# References

Brown, S. D., & Vranesic, Z. G. (2009). *Fundamentals of Digital Logic with VHDL Design*. New York, United States: McGraw-Hill Education.