Jason Ou (jaou 1385128)
Partner: Vincente Orellana
CMPE 12L
2 FEB 2015
TA: Daphne Gorman
Section: 6

Lab Write-Up #1

OVERVIEW/INTRODUCTION

The purpose of this lab was to design and build a multi-page schematic for the operation of a 4-bit ALU that is capable of performing addition (ADD), negation (NOT), and bitwise (AND).

METHODOLOGY/PROCEDURES

PART A: Memory

For the memory portion of the lab, we had to implement an address decoder using logic gates built from a given truth table. Along with the logic gates controlling the memory address, we also had to create four four-input multiplexers using senders from the registers as input for the multiplexers and outputted values based on the values stored in the registers by the user.

PART B: ALU

The ALU portion of the lab requires the implementation of operations depending on the currently selected opcode to perform either addition (AND), negation (NOT), or bitwise (AND). If neither of these were to be selected, then a LED for BC (bad code) turns on. Using logic gates, we can create these operations. The operations receive input from two keypads which the user decides what to input into the operation. The ALU should also be able to check for if the value outputted is a Z (zero), P (positive), N (negative), or a BC (bad code). A multiplexer should also be used to control the inputted values with the currently selected opcode to create an output for the user to see with a 7 segment LED.

PART C: Memory + ALU

The combination of the memory and ALU makes it so that the user is only able to use one keypad, where the keypad can store values into addresses of the registers and using the register 3 and whichever is the current register the user is selecting, perform operations based off of the current opcode. The user can then select whether the output of the operation perform or the current keypad input is stored into the currently selected register address.

RESULTS

PART A: Memory

The memory portion worked as expected. It saved memory and was able to output memory, as well as reset, and replace already existing memory.

PART B: ALU

The ALU portion worked as expected. Using the keypad and the switches, we were able to perform arithmetic operations based on keypad inputs from the user and opcode selections from the user.

PART C: Memory + ALU

The memory + ALU worked after combining the memory and ALU together properly. Using a single keypad and switches, we were able to store values into registers 0, 1, 2, 3. We were able to store between the user output or a value from the keypad into our registers as well as perform operations based on our current register combined with register 3, and depending on our opcode as well.

DISCUSSION

Jason Ou (jaou 1385128)
Partner: Vincente Orellana
CMPE 12L
2 FEB 2015
TA: Daphne Gorman
Section: 6

1. **How is sequential logic useful; i.e., why are the inputs and outputs latched?**

    - Sequential logic is useful since it allows us to reuse data, making it so that it allows for larger, and more organized, and more efficient projects to be constructed.

2. **What materials did you need to implement the ALU?**

    - Materials used to implement the ALU includes:
    - Use of logic gates to implement AND, NOT, and ADD for the ALU
    - Multiplexers in order to receive outputs from these logic gate operations
    - Keypads to receive user input to do operations with the keypads
    - Switches to select the opcode, and control carry ins
    - LEDs to monitor the status of the operations being used and the properties of the output
    - 7 Segment LED displays to view the input/output/results

3. **How many opcodes (types of instructions) does LC-3 have?**

    - There are 16 opcodes in the LC-3.

4. **Why isn't a SUB instruction (subtract) included in the LC-3 instruction set?**

    - A SUB instruction is not included in the LC-# instruction set because of the similarities between the ADD instruction and the SUB instruction, making it so that if the SUB instruction were to be included in the LC-3 instruction set, it would be less efficient circuit wise.

5. **Why isn't NAND instruction included in the LC-3 instruction set?**

    - The NAND instruction isn't included in the LC-3 due to reasons similar as to why the SUB instruction is not included in the LC-3. It is similar to the AND instruction, making it less efficient to implement both.

6. **How would you modify your particular design to implement a NAND instruction also?**

    - In order to modify our design to implement a NAND instruction, we could replace the AND circuit gate implementations with the proper NAND circuit gate implementations.

7. **What is the purpose of the N, Z, P, and BC LEDs?**

    - The N, Z, P, and BC LEDs were used to tell the user if the output of their operation were N (Negative), Z (Zero), P (Positive), or BC (A bad code, or invalid opcode).

CONCLUSION

This lab was used to learn more advanced MML techniques by creating our own ALU. I was able to learn how to use MUX and create registers to do operations and store memory depending on different operations. In the end, I was able to create my own ALU that was able to access and create new memory, while at the same time be able to perform different operations.