**Lab 8: ad-hoc Design**

**Primary Objectives**

1. Implement an ad-hoc system using Logisim

2. Test to verify the functionality of system

*Objective 1 Implementation*

In this system, there are three inputs, a 4-bit input IN, a 2-bit input F, and a 1-bit input GO. There is also a 4-bit output OUT. This system is supposed to function as a simple calculator which can either load a value, or add/subtract/multiply that loaded value with IN. F controls which of the four operations is done. OUT displays the result of the operation and the value currently stored in memory and is only changed when GO switches from 0 to 1. OUT holds even if GO remains 1 for several clock ticks. To implement this system, I started by making the simplest operation: the load operation. After verifying that OUT would display whatever was held in the register, I realized that OUT and the register were changing constantly whenever the clock ticked. To solve this, I made a brief state diagram and implemented a sequential system attached to a multiplexer to allow GO to only change OUT once per positive value change. I then set up a four-input multiplexer to handle the four operations controlled by F. I then simply attached IN to 00, and an adder, subtractor and multiplier to the remaining three mux-inputs to accomplish the final three operations.

*Objective 2 Testing*

Figure 1 shows one way the system could be implemented via Logisim.

A diagram of a circuit

Description automatically generated

Figure 1 Implementation

To test this circuit, I logged various values for IN for each of the four operations. The results of these logs can be seen below.

A screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

Loading Log Addition Log Subtraction Log Multiplication Log

As can be seen, all four logs display the expected behavior for each mode of the system. While this circuit was designed with unsigned magnitude binary representations in mind, the system also functions perfectly with two’s complement representation. In any of these logs, the numbers can be viewed as either representation, and the system still functions as expected.

**Conclusion**

The lab went relatively smoothly. I did not initially expect to have to use sequential logic, but it was not too difficult to implement. I also think it may be possible to implement the system without using sequential logic, but this was the quickest method of getting GO to work that I could think of on the spot. Regardless, the system functions as intended.