

# MAT 3670: Lab 6

## Digital Logic Structures — Part II

### Background

This lab will give you an opportunity to explore various logic circuits, including the multiplexer, decoder, full adder, and a simple ALU<sup>1</sup> using a bit-slice design.

### Pre-Lab Exercises

Read all of the laboratory exercises in order to understand what you will be asked to do in the lab.

### Laboratory Exercises

1. Login to your Linux account and descend to your 3670 directory. Create a **lab6** directory to store the circuits you will create for this lab.
2. From a terminal window, descend to your **lab6** directory, then launch the **Logisim** program:

```
logisim &
```

3. We have seen how it is possible to use logic circuits as useful building blocks. For example, we can design a 4-to-1 multiplexer by using only 2-to-1 multiplexers. Show how to construct an 8-to-1 multiplexer by using only 2-to-1 multiplexers. Your circuit should have eight data lines,  $D_0$  through  $D_7$ , with three select lines,  $S_0$ ,  $S_1$ , and  $S_2$ . It is simplest to use the multiplexer found in Logisim's "plexer" library<sup>2</sup>. Store your circuit in **mux8.circ**. Test your circuit thoroughly to verify it works as desired.
4. Figure 1 shows the circuit diagram for a two-input decoder. Create this circuit, saving it in **decoder.circ**. Viewed as a black box, this decoder has two inputs and four outputs. By simulating the circuit, construct the truth table that describes the behavior of the two-input decoder. Then, in words, describe the effect of this circuit. Put your truth table and explanation in a file named **README**.
5. Figure 2 shows the circuit diagram for a full adder. Create this circuit, saving it in **fa1.circ**. Verify this circuit behaves as a full adder by trying all possible combination of input values. Produce a truth table for this circuit. Add the truth table for this circuit to your **README** file.
6. Figure 3 shows how the full adder can be modified. Create this circuit, saving it in **fa2.circ**. How does the behavior of this circuit differ from the full adder? Simulate the circuit on all possible input values, then determine its associated truth table. Can you describe the overall effect in words? Add the truth table and your explanation to your **README** file.
7. Figure 4 shows how a simple one-bit ALU can be constructed. Notice how this circuit uses basic logic gates, a decoder circuit, and a modified full adder. Implement this circuit and experiment with it with the goal of determining its behavior. (You can access the circuits you saved earlier, using them as sub-circuits by making use of Logisim's ability to load libraries of circuits. In this case, you want to load a Logisim library, not a built-in library.) In words, explain what this circuit does. It helps to realize that two of the inputs,  $S_1$  and  $S_0$  act as an

---

<sup>1</sup>At the heart of every computer is an ALU — the **arithmeti**c logic unit.

<sup>2</sup>To access this, choose **Project | Load Library | Built-in Library**. You will then be able to expand "Plexers" in the explorer pane, where you will find a collection of multiplexers.

“operation code”—each two-bit pattern specifies one specific action. Add your explanation to your README file.

- The “payoff” for your work is just around the corner! Now that you have a design for a one-bit slice of an ALU, they can be chained together to operate on multi-bit operands. Design a four-bit ALU, shown in Figure 5, with the same four operations of our one-bit ALU. Each slice of the ALU will have the responsibility of computing one bit of the result. Using the block diagram for the one-bit ALU, design the connections needed within a chain of four ALU units, then implement your circuit in Logisim, saving the result in `alu4.circ`. Be sure to test your final circuit to ensure it has the desired behavior.

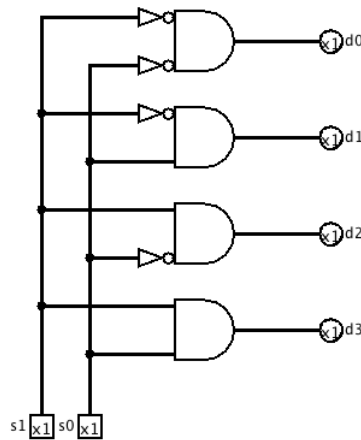


Figure 1: A two input decoder. Among the four AND gates, every pattern of negated and non-negated inputs appears.

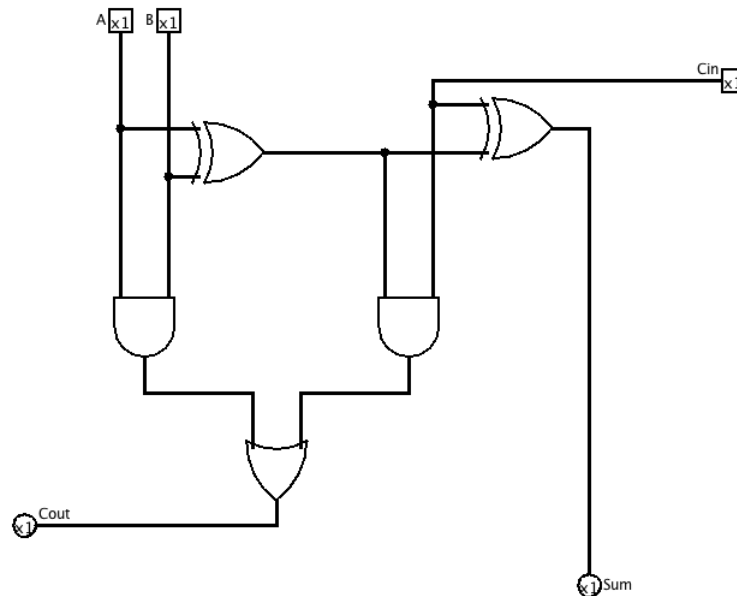


Figure 2: A full adder. There are three inputs and two outputs.



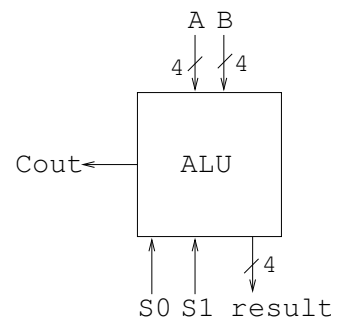


Figure 5: A block diagram for a four-bit ALU, showing a high-level view. Within this ALU, there is a chain of four one-bit ALU units.

## Submissions

- Create a **tar** archive file consisting of the **lab6** directory:

```
cd ~/3670
tar -cvf lab6.tar lab6
```

- Submit your work:

```
submit lab6.tar mat3670
```