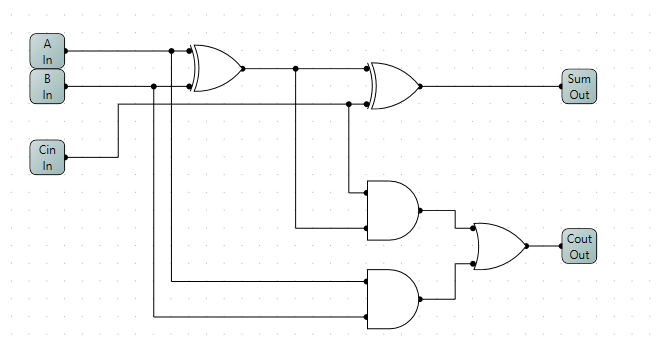
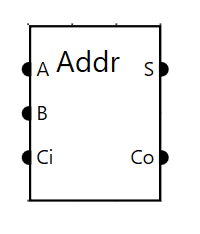
# Lab 3

# Building a ALU

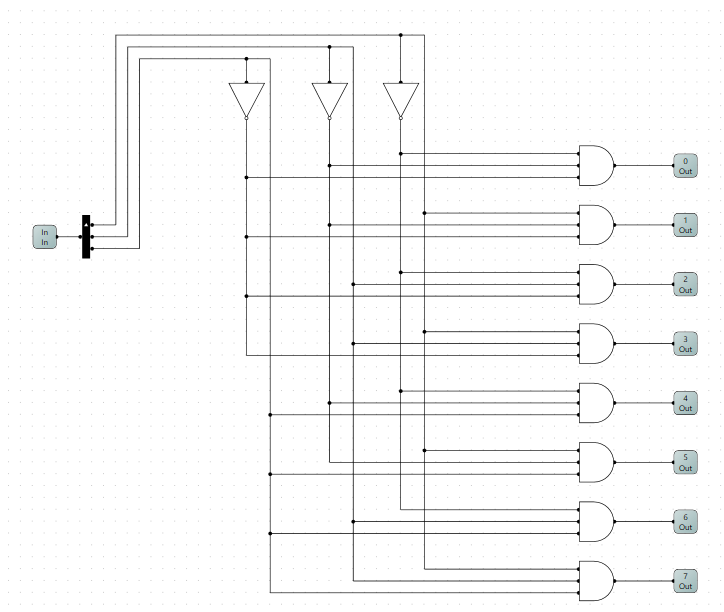
Welcome Back. In this lab we will replicate what was done in class. The goal is to create a functional 4 Bit ALU with 5 functions, ADD, SUB, NOT, AND, OR. We will build it in functional units.

Part 1.

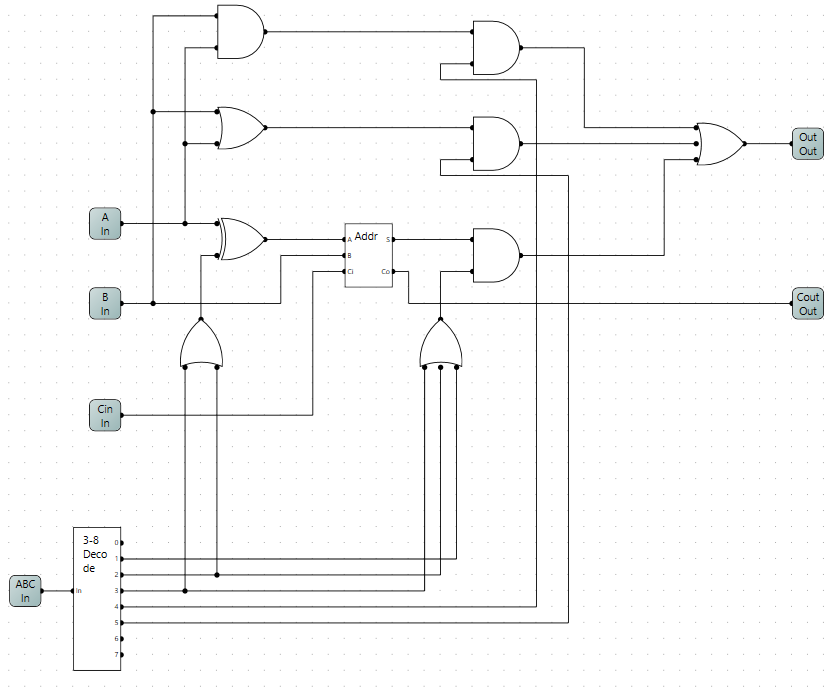
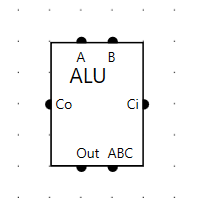
The basis for any ALU is an ADDER. From the circuit given create a package called Full Adder. Most of what we do will depend on this one unit. To ensure it is functional, create a small test circuit to ensure it functions as expected.

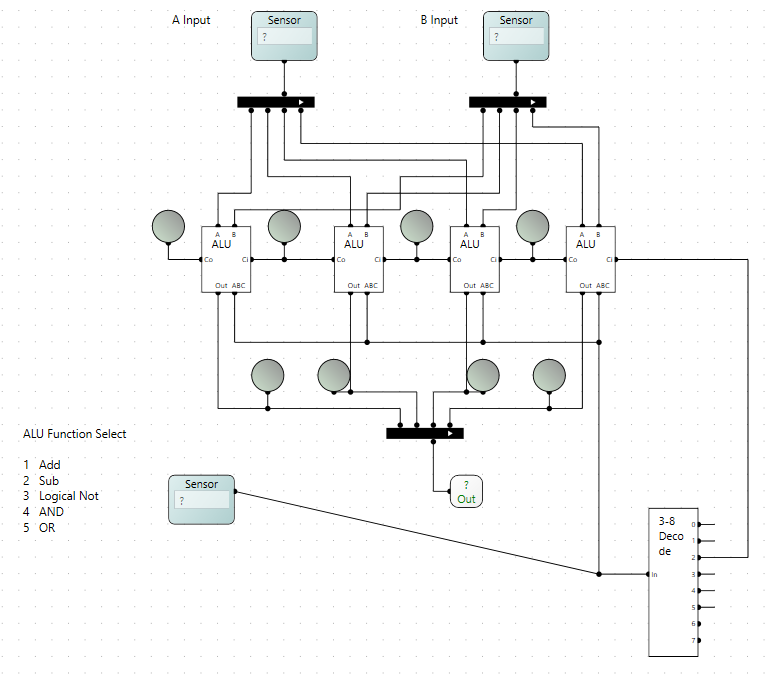
Part 2

Before we can create a ALU we need to create a component called a 3 to 8 decoder. This will be used to select the functions of our ALU. In terms of tedious this implementation rings all the bells. One AND gate will become active for each unique 3 bit input. Once you have created the decoder, put it into a test circuit to ensure it does what it was intended to do. This is very similar to debugging code, write a little code, write a small test method. DO NOT PROCEED UNTIL IT WORKS.



Part 3.

Here is a 1 bit ALU with all 5 functions implemented. There are 3 main units to this ALU. 1) The ADDER implements ADD, SUB and NOT. Notice that the decoder generates the appropriate functions for each. The XOR on the input to A will provide the bit inversion for SUB and NOT. The AND gate exiting the ADDER acts as an ON/OFF switch and is enabled in the on position when any of the 3 above functions are selected. 2) and 3) the AND and OR functions are implemented as parallel functions, both connecting to the A and B inputs. The AND gates on their outputs are switches which are enabled appropriately depending on which function we want. When you package the ALU do so as below, this makes the next part much cleaner. You can select which side of the chip the pin in to appear on.

Part 4

The circuit to the right is a completed 4 Bit ALU. The sensor input at the bottom will select the ALU function, to implement subtract, the 2’s complement requires that the A input be inverted, which was taken care of in the circuit above, but also the addition of a +1. This is accomplished by forcing the carry in to bit 0 of the ALU to 1, hence the reason for the decoder on the bottom right. LEDs have been placed on the carry outs and the ALU outs to make it easier to see what is happening.

As a quick exercise replace that decoder with explicit logic gates, which will output a 1 when the input is 2.