**ALU Lab Report**

ITSC 3181 - 11/1/21

[Noah Parkes](mailto:nparkes@uncc.edu), Sean McLellan, Shashank Mondrati

**Objectives**

A function in our ALU is the A+B. This function works like an OR function. It requires that one or both of the inputs are one for the output to be one. This ALU function could be useful for survey forums. Some surveys require that people pick one or multiple options from a list in order to answer the question. The A+B function could be able to check whether the user selected anything from the list of options. Let’s say that one question has two answers that a user could select. If the user selects the first option they would get the output 01. If the user selected the second option they would get the output 10. But, if the user selected both options it would output 11. This allows the computer to tell which option the user selected or if they selected both options.

Another function in our ALU is the A-B. This function works like an exclusive OR function. It requires that only one of the inputs are one for the output to be one. This ALU function could be useful for surveys for picking an answer out of a list of questions. The ALU function allows the computer to check to make sure that the user only selected one option from a list of options. If the user selects the first option, 01, they would get an output of 1. If they selected the second option, 10, they would get an output of 1. But, if they selected both, 11, they would get an output of 0. This would prevent users from selecting both options for the question.

We also implemented in our ALU an A&B. This ALU function works similarly to an AND function. The function requires that both inputs are one in order for the output to be one. It could be useful for making sure that a user answers all the required questions in a survey. If there are two questions in a survey and both questions have to be answered. If you answer a question it will output a 1. So if you answer both questions, you will get an output of 1. But if you do not answer both questions you will get an output of 0. This could prevent the user from submitting the survey before answering all the questions.

Another function we implemented in our ALU is passing A. This function is used for passing a value. This could be useful for a calculator. If the user presses the five button on a calculator the input for the computer would be 1001. This value would be passed through and allow the user to perform an operation on it.

The last function we implemented in our ALU was passing the inversion of A, -A. This function would be useful for finding the negative version of a number. This could be useful for a calculator for finding the negative version of a number. So if the user types in -5 into a calculator. The function would make the negative version of 5.

**Technical Details:**

For bitwise NOT, the purpose is to pass one input and get the inverse output according to the following table.

| A | **NOT** A |
| --- | --- |
| 0 | 1 |
| 1 | 0 |

For binary Addition, the purpose is to add inputs, regardless of what they are, and combine them using the Binary addition rules that are in the following table.

| **Addition** | **Result** | **Carry** |
| --- | --- | --- |
| 0+0 | 0 | 0 |
| 0+1 | 1 | 0 |
| 1+1 | 0 | 1 |
| 1+0 | 1 | 0 |

For bitwise A-B, the operator works like an OR condition, taking in two operations and works the same way as Binary operation ADDITION, but the difference is subtracting instead of adding. For example in addition 1+1 result is 0 and carry is 1. But in bitwise A-B, 1-1 would yield 0.

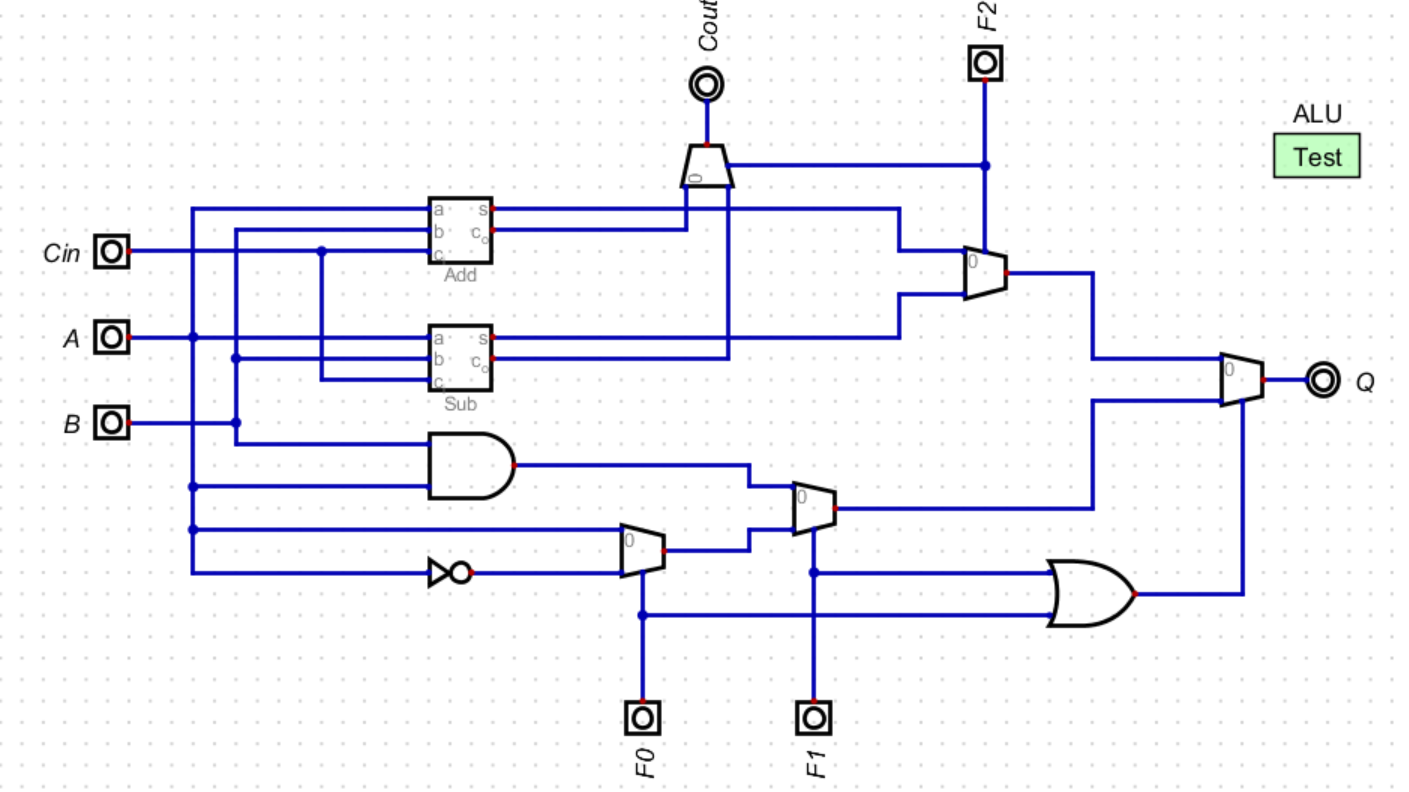
| **Subtraction** | **Result** |
| --- | --- |
| 0-0 | 0 |
| 0-1 | 1 |
| 1-1 | 0 |
| 1-0 | 1 |

**For bitwise A&B, the purpose is to take inputs, and return the AND condition operation, given in the following table.**

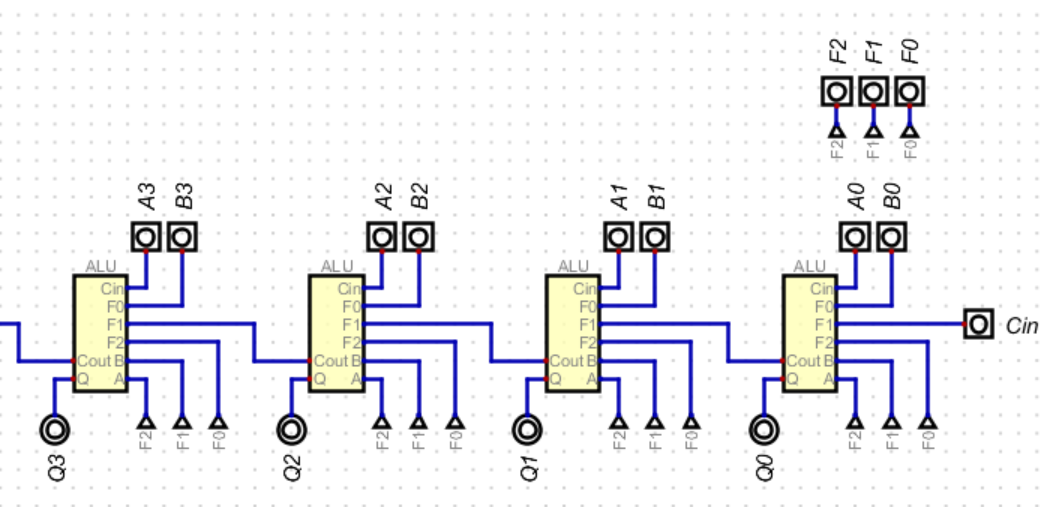
| **A** | **B** | **A&B** |
| --- | --- | --- |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |

**Screenshots:**

This is the schematic of the 1-bit ALU that we implemented. It has the inputs from A, B, and Cin. F0, F1, and F2 determine which function to perform in the ALU. The output goes to Q and Cout has the carry for the add function.



This is the 4-bit ALU function. Each bit contains the 1-bit ALU schematic. Each bit has its own A and B input. It also gets input from F0, F1, and F2. It then outputs to their Q output. We used this schematic to make sure the ALU worked before implementing the 16-bit ALU.



This is the 16-bit ALU. We just took the 4-bit ALU and did it four times to make the 16 bit ALU. To test the function we used the 4-bit ALU to see if it would work.

