
# North South University

## Department of Electrical & Computer Engineering

## Lab Report

| | |
|---|---|
| **Experiment No:** | **06** |
| **Experiment Title:** | **Design of an ALU** |
| **Course Code:** | **CSE332L** |
| **Course Name:** | **Computer Organization & Architecture Lab** |
| **Name & ID:** | **Ishrat Jahan,1921909042** |
| **Date of Experiment:** | **09/12/2021** |
| **Date of Submission:** | **14/12/2021** |

## Objective:

1: Building 1-bit ALU with specific set of instructions.

2: Building 16-bit ALU by connecting 16 one-bit ALU.

3. Designing the ALU to perform operations:
   Add, Sub, And, Or.

4. Incorporating zero flag in the ALU.

## Theory

**ALU:** An ALU stands for arithmetic-logical unit. It is a combinational circuit that can compute arithmetic operations like addition and subtraction as well as logical operations like AND and OR. The MIPS word in an ALU refers to the bit width of the ALU. For example: If the MIPS word is 32 bits wide, we need a 32-bit wide ALU. The input and output will be 32 bit each.

**1-bit ALU:** An 1-bit ALU have two 1-bit inputs A and B and an 1-bit output S(Sum). The 1-bit ALU performs Addition, Subtraction, AND and OR operations.

**Operations:** To perform the logical operations AND and OR, the 1-bit ALU uses AND gate and OR gate respectively for each of them.

For performing the arithmetic operation:

**Add:** An 1-bit adder is used. One input is same for both addition and subtraction, therefore it doesn't need varying.

**Subtract:** For subtraction we know: $A + B' + 1$.

To change B we can keep a selection by using a MUX. The MUX will have two inputs, B and B' and the selection is Binvert. When the Binvert is 1 and the Cin of the adder is 1, the output of the MUX is the 2's complement of B. Therefore, the adder will carry out the subtraction process.

For selecting which operation will be carried out, there will be a second MUX to choose operations: Add, Subtract, AND, OR by varying the two bit's selection bit. It will have 4 inputs, one from AND gate, other from OR gate and the last two from the Adder.

Here, 00 performs AND operation.
01 performs OR operation
10 performs ADD operation.
11 performs SUB operation.

16-bit ALU: By connecting 16 1-bit ALU, a 16-bit ALU can be designed. It will have two inputs each of 16 bits and an output (S) of 16-bit as well. The 16-bit ALU will also have a zero flag.

Zero flag: It is a single bit flag that is used to check the result of an arithmetic operation. The zero flag is set (1) when the result of an operation is zero. The zero flag will be 1 only if the entire result has all zeroe's. Otherwise the zero flag will be zero in presence of 1.

For example: For a 4 bit-result

Sum = 0000, Zero flag = 1.
Sum = 1000, Zero flag = 0.

The zero flag only considers the output sum and excludes the carry out. It doesn't matter if the carry out is 1/0.

## Advantage of using 1-bit ALU to construct n-bit ALU:

While using 1-bit ALU to build a n-bit ALU, the change of things in the circuit is simple and easy. If we were to build a 16-bit ALU directly, it would have required 16 AND gates, 16 OR gates, 16 Adders and the whole circuit would have been messy and hard to figure out if anything went wrong. Therefore, the approach of 1-bit ALU to build n-bit ALU was used. So that changes could be easily made.

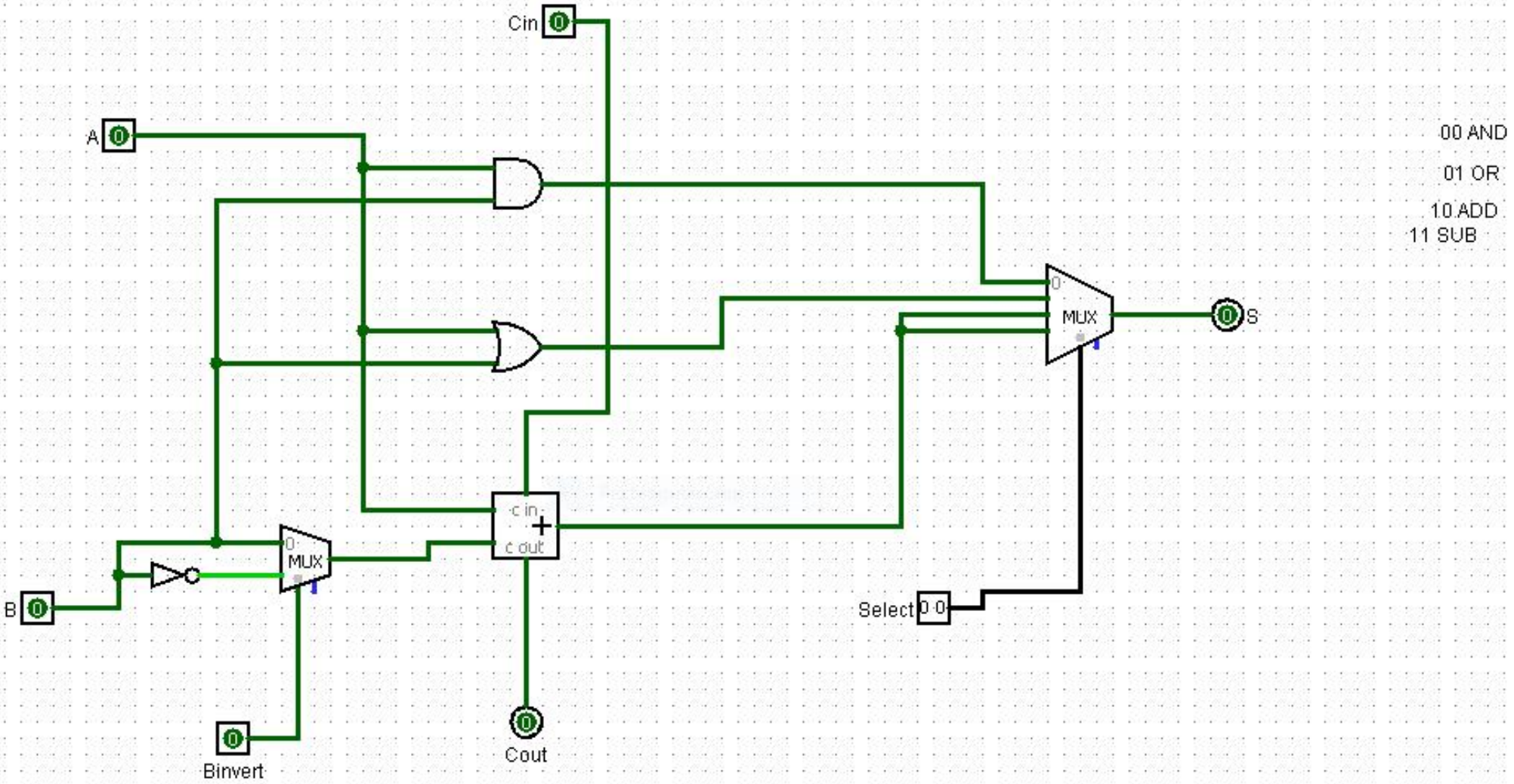


Figure:1-BIT ALU…

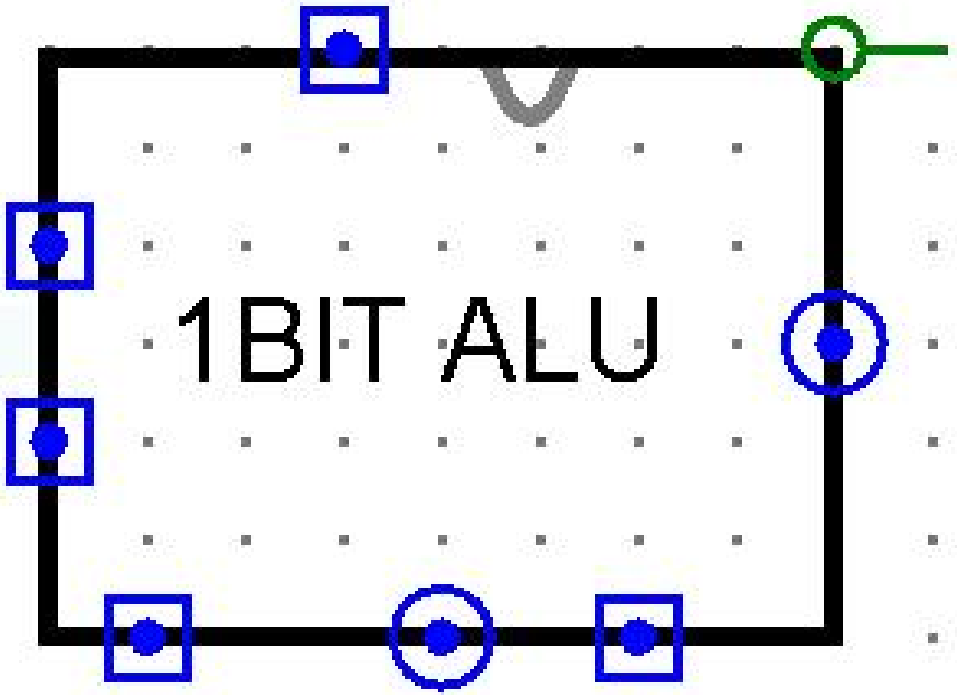


Figure:Subcircuit of 1-bit alu…

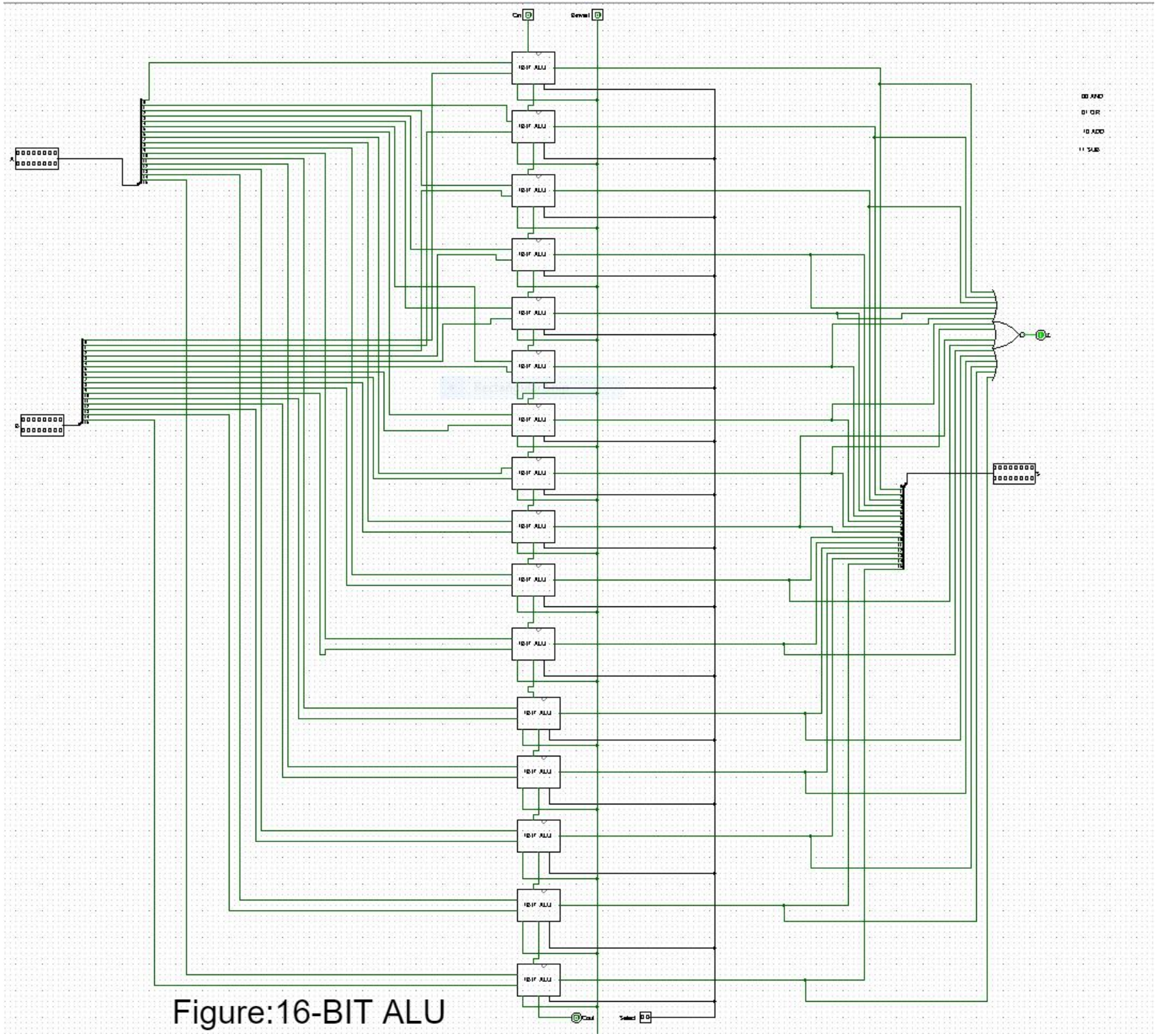

Figure:16-BIT ALU

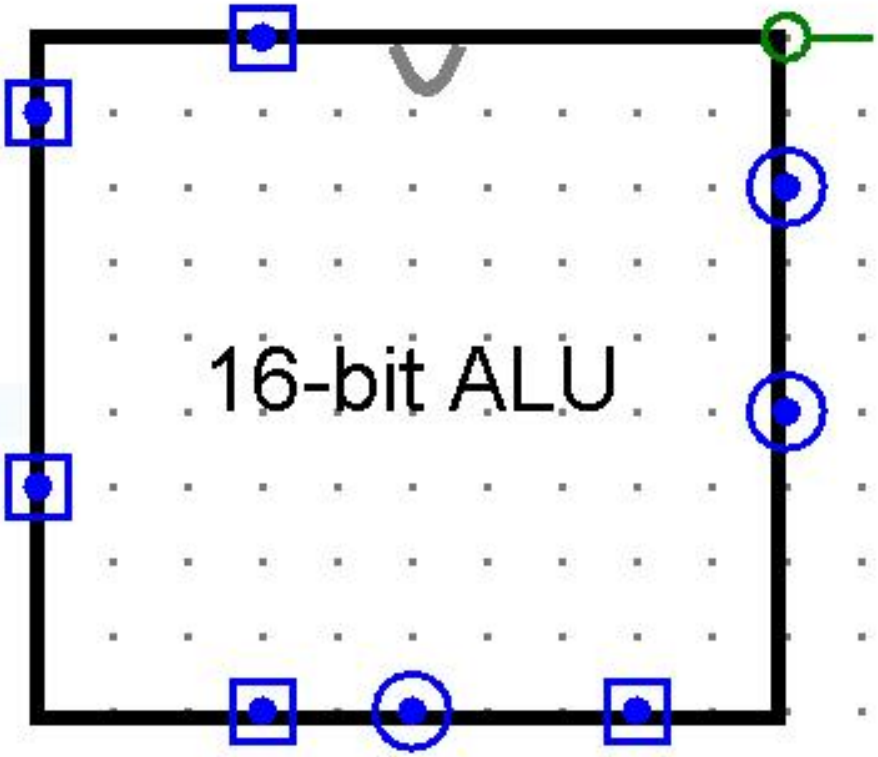


Figure:Subcircuit of 16-bit alu

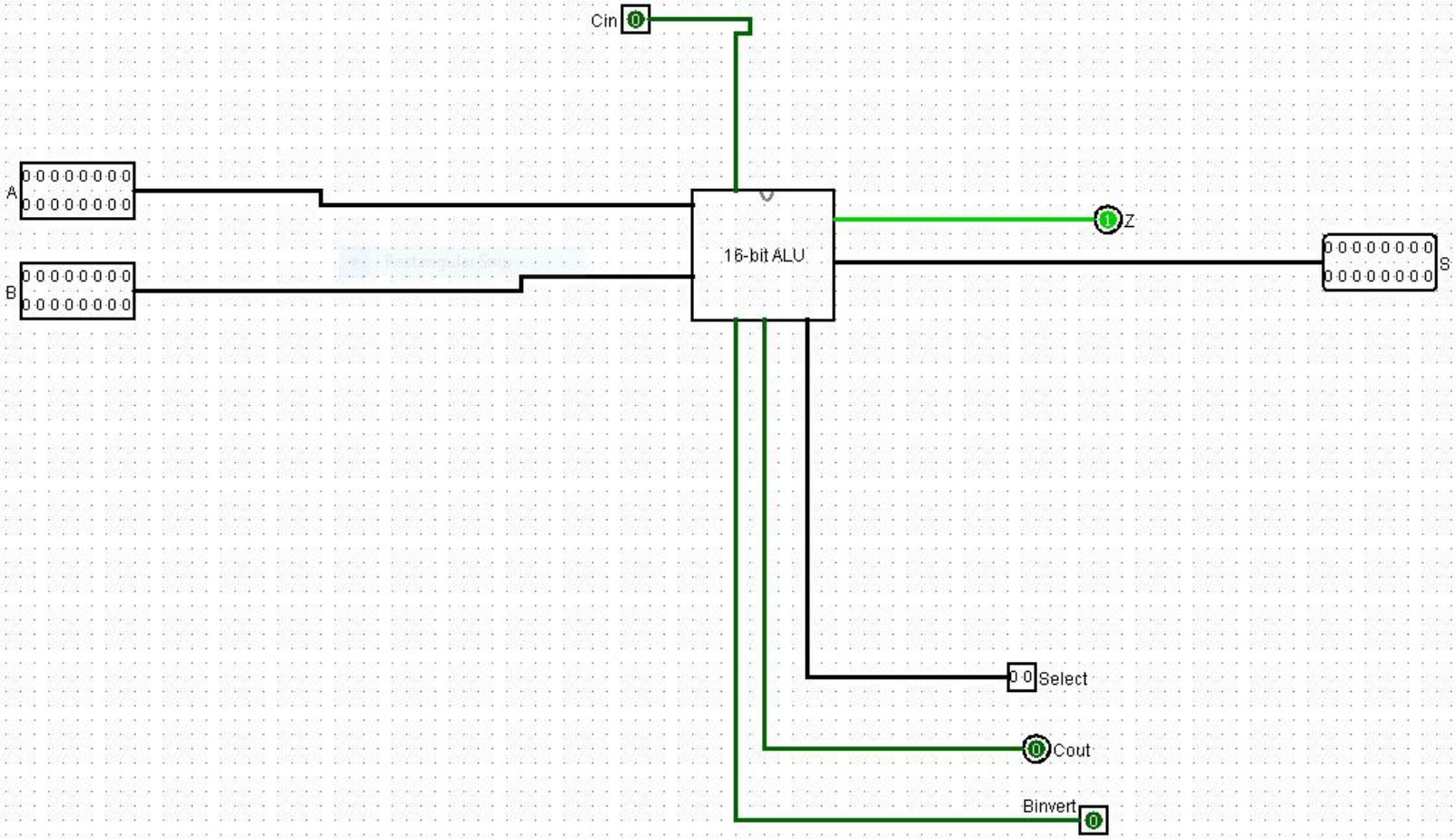


Figure:16 BIT-ALU(Final Circuit)

## Discussion:

The experiment of Lab06 was based on "Design of an ALU". In this lab, we learned how to build an 1-bit ALU and by using it how we can design a 16-bit ALU.

At first we studied what an ALU is and how it works. We learnt that ALU is a combinational circuit that can perform various of arithmetic and logical operations. The MIPS word refers to the bit width of an ALU. For example: If the MIPS word is 16-bits wide, we need a 16-bit wide ALU. As per the instructions in the lab manual, we were suggested to design a 1-bit ALU and combine 16-1-bit ALU to build a 16-bit ALU. The reason for doing so is changes can be easily made to the circuit. If I change anything in the 1-bit ALU all other 1-bit ALU are automatically updated. Moreover, if we build a 16-bit ALU, it will require 16 AND gates, 16 OR gates, 16 Adders and for every change we need to update everything 16 times.

Then we studied the arithmetic and logical operations the ALU will perform. For the logical operations: AND and OR, it can be done using AND gate and OR gates respectively. For addition, we can use an Adder. For Subtraction we know the equation is:- $A + B' + 1$. The input A will be same for both Addition and Subtraction but we need to change B; that is for Addition it will be B and for subtraction it will be B'. It can be done by using a 2x1 MUX which will have input B and B' and a selection known as Binvert which is turned to 1, the output of the MUX is the 2's complement of B and the subtraction operation will be

carried out. And for choosing between these four operations we will be using a 4x2 MUX where the select bits will control which operation will be carried out. Here, 00 carries AND operation, 01 does OR, 10 does ADD and 11 does SUB. For SUB the carry-in must be 1.

Then we discussed what a zero flag is and how to design it in the circuit. A zero flag is a single bit flag that is used to check is the output of an operation is zero or not. If the zero flag is 1, the output consists of all zeroes. otherwise if it is 0, the output might be all 1 or mixture of 1 and 0. To incorporate a zero flag in a circuit, we noticed for a two bit sum

| (S1 S0) | Z | AND | OR |
|---|---|---|---|
| 0 0 | 1 | 0 | 0 |
| 0 1 | 0 | 0 | 1 |
| 1 0 | 0 | 0 | 1 |
| 1 1 | 0 | 1 | 1 |

that, zero flag is when the circuits works in the invert of OR, that is NOR. Therefore, if we connect a NOR gate with the circuit, the output will show the zero flag. The theoretical part of the lab ended after this.

To make it easier for us, our lab instructor showed us how a create a 4-bit ALU. The first step was to design a 1-bit ALU. It has two input of 4-bits each and the 4-bit output. The rest of the circuit was constructed as per explained in the theory. All these circuits were created with 'Add circuit' to be used later for another lab.

Then we were taught to create seperate file's for the 4-bit ALU which was created using the sub-circuit of the 1-bit ALU. After showing the whole procedure we were asked to complete our lab task which was to design a 16-bit ALU.

The first step in creating a 16-bit ALU, was to create a 1-bit ALU. For creating 1-bit ALU, I opened the Lab-file that have the Register File, right clicked on the folder name in logisim, selected the 'Add Circuit' option and named it as 1-bit ALU. To create 1-bit, first I took two input pin of 1-bit and labelled them A and B. For AND and OR operation, I took two gates from gates section, one AND gate and another OR gate both of data bits 1. Then I connected input A and B to them. For addition, I took an adder from the arithmetic section of data bits 1, and connected it to the adders one input port. For the other input B, we need to select among B and B' so I took a MUX of select bit 1 and connect B and B' (B with NOT gate) to the MUX's second input port. Then I took anothere two 1-bit input pin and connected one of them with the MUX's select port and labelled it as Binvert. Binvert will be used to control what will be the output of the MUX (B or B'). The othere input pin was labelled Cin and connected to the adder's Cin. For the Carry out, I took an output pin labelled it as Cout and connected it to the Cout port of the Adder. As there was suppossed to be a choice, the MUX's output was inputted to the adders second output. As we were going to control

which operation out of these 4 operations will be carried out, we needed another mux of 4 inputs and selection bit 2. The output from the AND gate, OR gate and Adder was added to the MUX. The last input was also the output from the adder. After that for the selection bit, I took an input pin, changed it's data bit to 2, labelled it as Select and connected it to the MUX. For displaying the output, I took an output pin, labelled it as 'S' and connected with the MUX's output. Then I created the sub-circuit of the 1-bit ALU using the sub-circuit option in Logisim in the left most corner. I placed the input, output, select bit, Cin, Binvert, Cout and S as per my convenience. With it, I completed the design of 1-bit ALU.

The second step was to create the 16 bit ALU with 16 1-bit ALU. For that, I again created another file, named it 16-bit ALU. Then I drag and dropped 16 1-bit ALU subcircuits. For the input, I took two input pin, changed there data bits to 16 and labelled them A and B respectively. Then I took two splitter's, changed their bit width and fan out to 16 and connected to A and B. Then I connected the input's with the ALU's in order; $A_0B_0$ with the 1st ALU, $A_1B_1$ with the second, till the MSB's are connected with the last ALU. For the Cin, I took an input pin of 1-bit, and connected it with the Cin part of the ALU. The Cout of the first ALU will be the Cin of the second ALU and so on. So I connected all the Cin's and Cout's in that order. Then I took another input pin of data bit 1, labelled it as Binvert and connected it with the Binvert port of the ALU. I did the

same with the select port. For that I took an input pin, changed it's data bit's to 2 and connected it to all the ALU's selection port. For displaying Cout, I took an output pin, labelled it as Cout and connected it to the last ALU's Cout port.

For displaying the output, I took an output pin, changed it's data bits to 16 and labelled it as S. Then I took a splitter from the wiring section, changed it's bit width and fan out to 16 and connected it with the output pin. The 1st ALU is the LSB so it was connected with the splitter's 0 part. The second ALU was connected with the splitter's 1 part and so on. This way I connected all the ALU output to the splitter. The output 'S' was completed.

For displaying the zero flag, I took a NOR gate from the gates section and changed it's number of inputs to 16. Then I connected the ALU output with the inputs of the NOR gate in sequential order. The first ALU was connected with the 1st input and so on. The result was displayed by using an output pin. I took the output pin of data bit 1 and connected it to the NOR gate's output and labelled it as Z. The diagram of 16-bit ALU was completed. Then I created the sub-circuit of the 16-bit ALU with the inputs, outputs and select bits, binvert, Cin in specific positions.

After that I created another file using 'Add Circuit' and named it 16-bit ALU final. I dragged and dropped the sub-circuit of the 16-bit ALU in that file. After that I took two input pin, changed

there data bits to 16 and connected it with the sub-circuits input ports. I took two input pins again with data bits 1 and connected one of them with Cin and the other with the Binvert port. For the select option, I took an input pin changed it's data bits to 2, labelled it as Select All the inputs were labelled. For displaying Cout, I took one output pin, labelled it Cout and connected it with the Cout port. For 'Z' (zero flag, I did the same and labelled the output pin as Z. For displaying 'S', I took an output pin, changed it's data bit's to 16 and labelled it as S and connected it with the output port. The circuit was completed. After completion, I checked the outputs. The circuit worked correctly.

The only problem I faced in this experiment, is the circuit of 16-bit ALU was too big and congested. Moreover, there were very less space as there were many components in the circuit as well as many connections. So, I had to be very careful while connecting everything. Overall, it is easy to design if designed cautiously with enough time or hard.