Lab Manual
Department of Electrical and Computer Engineering
School of Engineering and Physical Sciences
North South University, Bashundhara, Dhaka-1229, Bangladesh

Experiment No: 06
Experiment Name: Design of an ALU

Introduction:

An ALU (arithmetic-logical unit) is a combinational circuit capable of computing a variety of arithmetic and logical functions. ALU is the brawn of the computer, the device that performs the arithmetic operations like addition and subtraction or logical operations like AND and OR. This section constructs an ALU and illustrates how combinational logic works. Because the MIPS word is 32 bits wide, we need a 32-bit-wide ALU. It can be done by connecting 32 1-bit ALUs to create the desired ALU. We'll therefore start by constructing a 1-bit ALU. But to make things simpler we will build a 16 bit wide ALU by later connecting 16 copies of 1 bit ALU.

Objective:

We will have following objectives to fulfill:

1) Build 1 bit ALU with specific set of instructions
2) Incorporate equality check (zero signal), overflow detection and other necessary flags
3) Build 16 bit ALU by connecting 16 one bit ALU.

Experiment Details:

Assume, a 16 bit ISA with following fields. The formats of the instruction are as follows:

R-type

op (4 bit)	rs (4 bit)	rt (4 bit)	rd (4 bit)

I-type

op (4 bit)	rs (4 bit)	rt (4 bit)	immediate (4 bit)

J-type

op (4 bit)	Target (12 bit)

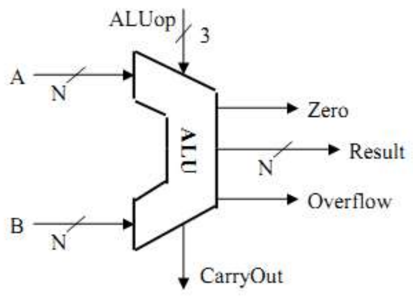
Functional Specifications:

Inputs : 2x16 bit operands – A, B. 1 bit carry input – Cin

Outputs : 1x16 bit result – S, 1 bit Carry Output – Cout

Operations : Add, Sub, And, Or

Functional Specification of the ALU



Equipment/Tool

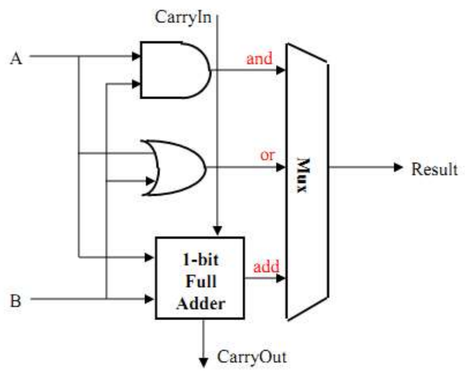
Logisim Tool

1 Bit ALU:

Following is the diagram of 1 bit ALU.

A One Bit ALU

◦ **This 1-bit ALU will perform AND, OR, and ADD**



Task Lists A

1. Implement the above diagram in logisim. Label each of the input/output/mux selections. Create a table showing necessary control selections required for the execution of each instructions. Create a sub circuit of 1 bit ALU

2. Incorporate SUB instruction into this. Update the table.

3. Build 16 bit ALU based on this. Create a subcuircuit for the 16 bit ALU

Overflow and Zero Signal:

Overflow Detection:

Overflow means the result is either too large or too small to represent properly.

Example : -8 <= 4 bit binary number <=7

When adding operands with different sign, overflow can’t occur.

Overflow occurs when adding:

- 2 positive numbers but the sum is negative
- 2 negative numbers but the sum is positive

Task Lists B:

1. Implement a logic for overflow detection and incorporate it to the built ALU
2. Implement a logic for equality check (zero signal) and incorporate it.

Assignment:

1) Prepare the lab report.
2) Take a screenshot of your implementation and later include it in your lab report.