North South University

Department of Electrical & Computer Engineering

LAB REPORT

Course Name: **CSE332L- Computer Organization and Architecture Lab**

Experiment Number: 06

Experiment Name: Design of an ALU

Experiment Date: 03/08/2022

Report Submission Date: 09/08/2022

Section: 02

Group Number: 01

Student Name: **Md. Baker**	Score
Student ID: **1911672642**	
Remarks:	

Exp Name: **Design of an ALU**

Objectives:

- We have learnt about the basics of ALU.
- We have learnt about micro-operations in MIPS ALU
- We have learnt about overflow detection and zero signal.
- We have learnt about R-type, I-type Datapath
- We can build 1-bit ALU and 16-bit ALU with specific set of instructions.

Apparatus:

- ✓ Logisim evolution application
- ✓ 1-bit & 16-bit of Register
- ✓ 1-bit &16-bit of inputs and outputs
- ✓ AND, OR, NAND, XOR, NOR and NOT Gate, MUX Gate
- ✓ 2-Input Adder, Multiplier, Divider, Shifter and Bit Extender

Theory:

The ALU is a digital circuit that provides arithmetic and logic operations. It is the fundamental building block of the central processing unit of a computer. A modern CPU has a very powerful ALU and it is complex in design. In addition to ALU modern CPU contains a control unit and a set of registers. Most of the operations are performed by one or more ALU's, which load data from the input register.

Registers are a small amount of storage available to the CPU. These registers can be accessed very fast. The control unit tells ALU what operation to perform on the available data. After calculation/manipulation, the ALU stores the output in an output register.

Micro-operations, also known as micro-actions [are detailed low-level instructions used in some designs to implement complex machine instructions. Micro-operations perform basic operations on data stored in one or more registers, including transferring data between registers or between registers and external buses of the central processing unit (CPU), and performing arithmetic or logical operations on registers.

Arithmetic operations have a potential to run into a condition known as

overflow. Overflow occurs with respect to the size of the data type that must accommodate the result. Overflow indicates that the result was too large or too small to fit in the original data type and the Zero signal tells us whether the ALU just computed a result equaling zero.

The Datapath is the "brawn" of a processor, since it implements the fetch-decode-execute cycle. The general discipline for Datapath design is to (1) determine the instruction classes and formats in the ISA, (2) design Datapath components and interconnections for each instruction class or format, and (3) compose the Datapath segments. Implementation of the Datapath for R-format instructions are fairly straightforward - the register file and the ALU are all that is required. The ALU accepts its input from the Data Read ports of the register file, and the register file is written to by the ALU result output of the ALU, in combination with the Reg Write signal.

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, Nand, Nor, Xor, Multiplier, Divider, Shifter etc.

Circuit Diagrams:

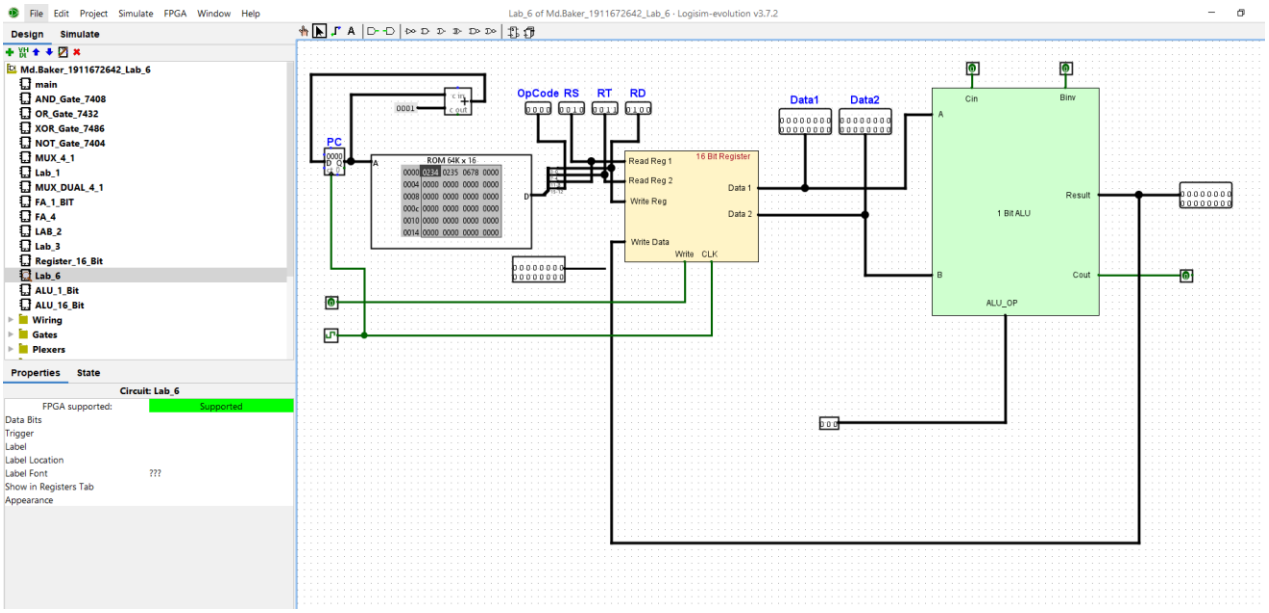

Figure: R-Type Datapath

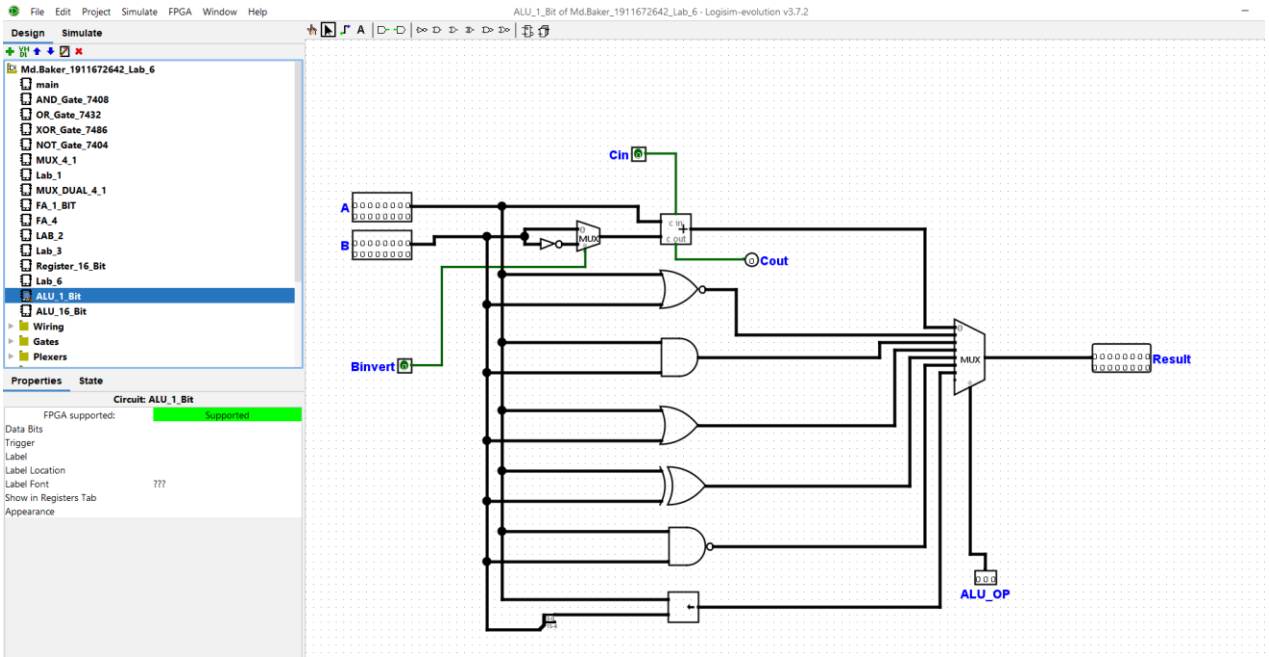

Figure: 1-bit ALU

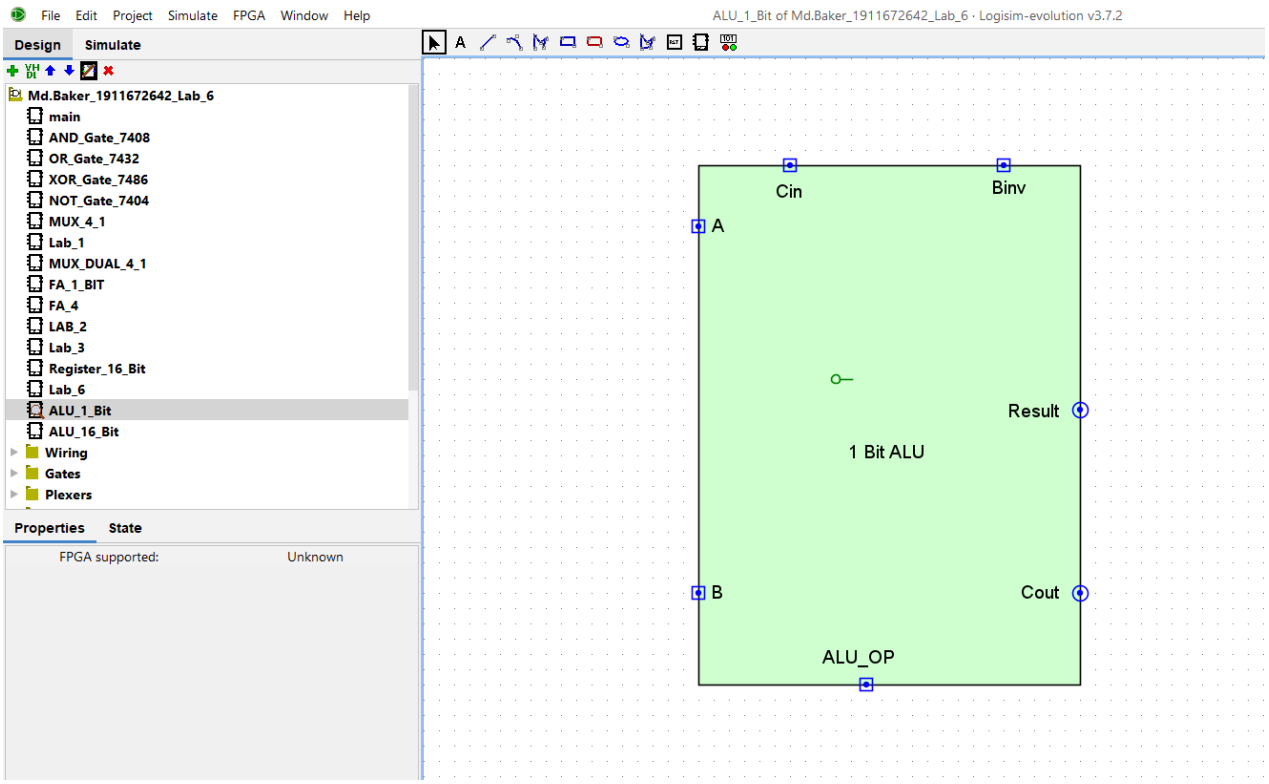

Figure: 1-bit ALU IC

Discussion:

In lab we design a basic ALU and also an interaction fetching R-Type and J-type Datapath. First, we design a 1-bit ALU. Our 1-bit ALU has zero signal and overflow detector. Then we design our R-type Datapath using our own designed 1-bit ALU.

At First, we design a 1-bit ALU which can do basic operations like AND operation, OR operation, add operation and Subtraction operations using gates and MUX. Combining the 1-bit ALU we created the 16bit ALU which we used in R-type and I-type Datapath. In 1-bit ALU design we used AND and OR gates for AND and OR operations and Adder for addition and Subtractor for subtraction operation. For Zero Signal we add all the output bits with an NOR gate and output shows 1 if all the output bits are 0. For overflow detection, we connect the last bit ALU's cin and cout with and XOR gate. It shows 1 if there is any overflow in the output.

Then we Design the R-type Datapath implementing our designed 1-bit ALU. In R-type data path - we have a single register in the beginning which will compute the instructions count. And next we have a ROM storage for storing multiple instructions. Then we have split the instruction into OP-code, RS, RT, RD and connected them to the register file inputs sequentially. We have other inputs connected to register file like clock, writing enable, writing data all. Attaching the 1-bit ALU in R-type and tested our ALU reading input and output. Using 3-bit operation inputs we chose our basic operations 000 for ADD and SUB,001 for NOR,010 for AND, 011 for OR, 100 for X-OR, 101 for NAND and 110 for shifting. After that completed the Design of ALU and the experiment for ADD, SUB, AND, OR, NOR, NAND, X-OR, Shifting.