1 Prac 5 - ALU[1]

1.1 Overview

An arithmetic logic unit (ALU) is a digital building block which can perform a variety of arithmetic, logical and bit shift operations. It is the base building block of a central processing unit and operates on integer binary numbers. The unit has both data and control signals. A basic ALU has two parallel input data busses (A) and (B) and one parallel output data bus (Y). It also has a parallel bus (F) which takes in an opcode or binary value which selects a specific arithmetic, logical or shift operation to be performed by the ALU on the two inputs A and B, it may also have a status output bus S which gives more information about the previous operation.

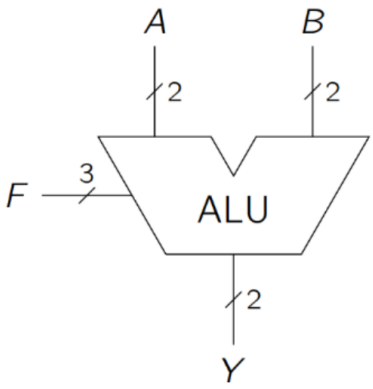

Figure 1: An Example of a simple ALU with a 2-bit wide data bus

1.2 Pre-Practical Requirements

- Logisim
- Understanding of the following components:
  - AND Gates
  - OR Gates
  - NOT Gates
  - XOR Gates
  - 1-bit full adders
  - 2-input multiplexers (MUX)
- There is no pre-prac assosciated with Practical 5

[1] *Orignal practical by Robyn Verrinder*

1.3 Tasks

- Design the ALU using the logical building blocks mentioned in the pre-practical requirements. Explain each step in your report. (30) [2]
  - 3 marks per sub-circuit for each operation. It must include a description of how the circuit works. 6 marks for the interlinking circuits including the multiplexers.
- Simulate your design in Logisim. Clearly label all inputs, outputs etc. Export a jpeg of the circuit and include it in your report. (10)

1.4 Deliverables

- At the end of the practical 5, you are to submit one .circ file (simulation file) and one .pdf file including your answers on Vula. The pdf and circ file must have the following format:

```
prac_<prac_number>_<std_number1>_<std_number2>.<extension>
```

- Marks will be deducted for incomplete or untidy reports.
- DO NOT COMPRESS THE FILES - Submit two files to Vula

1.5 Further Instructions

You are required to design and implement a 2-bit wide ALU with a 3 bit opcode, as shown in Table I below.

Table I: Three bit opcode and equivalent operation

Opcode			**Function**	
F_2	F_1	F_0	F	
0	0	0	A+B	Addition
0	0	1	A-B	Subtration
0	1	0	A+1	Increment
0	1	1	A-1	Decrement
1	0	0	A $\times$ 2	Multiply by 2
1	0	1	A $\div$ 2	Divide by 2
1	1	0	A $\wedge$ B	Bit-wise AND
1	1	1	A $\vee$ B	Bit-wise OR

[2] This is not a technical report as the one expected in practical 2. However, each building blocks must be represented in a figure, the functionality and the role of each building block in the bigger system must be explained.