Arithmetic Logic Unit

Introduction

A processor is a logic circuit that processes instructions. Some processors can be divided into the following submodules:

1. Control Unit: which controls the rest of the submodules and orchestrates their efforts to execute the instructions.
2. Instruction Pointer: which contains the address of the next instruction in memory.
3. Register File: which is a small store for data which can be accessed or overwritten by the processor. Compared to the system memory (RAM), the register file is much smaller but also much faster.
4. Arithmetic Logic Unit: which does the arithmetic and logical operations requested by the controller.
5. I/O Ports: which is used to connect between the processor and other components of the computer (e.g. printers, keyboards, other chips).

Processors can execute many types of instructions such as:

- Data Handling Operations such as moving data between registers and memory, writing immediate values and communicating data via I/O ports.
- Arithmetic and Logic Operations such as addition, subtraction, bitwise AND and OR.
- Control Flow Operations such as conditional jumps, procedure calls and returns.

Description

In this project, we will limit our scope to the Arithmetic Logic Unit.

The circuit must have the following three inputs: Operation, Operand 1, Operand 2. Each input must be 4 bits.

The circuit must have one output: Result. The output must be 8-bits.

The operations will be as follows:

Operation Name	Operation Code (4-bits)	Description
Move	0000 (Hex: 0)	Result = Operand1
Negative	0001 (Hex: 1)	Result = -Operand1 (2's complement)
Add	0010 (Hex: 2)	Result = Operand1 + Operand2
Subtract	0011 (Hex: 3)	Result = Operand1 - Operand2
Multiply	0100 (Hex: 4)	Result = Operand1 * Operand2
Not	1000 (Hex: 8)	Result = ~Operand1 (1's complement)
And	1001 (Hex: 9)	Result = Operand1 & Operand2
Or	1010 (Hex: A)	Result = Operand1 | Operand2
Xor	1011 (Hex: B)	Result = Operand1 ^ Operand2
Shift Left	1100 (Hex: C)	Result = Operand1 << Operand2
Shift Right	1101 (Hex: D)	Result = Operand1 >> Operand2

Notes:

- We will assume the carry in (and borrow in) for "Add" (and "Sub") to be always 0.
- The carry out (and borrow out) should be added to (or subtracted from) the extra 4 bits.
- For "ShL", the output will be zero if "Operand2" value is greater than 7.
- For "ShR", the output will be zero if "Operand2" value is greater than 3.

Test Example

These are a few examples that cover how the operations should work (Everything is written in hexadecimal):

Operation	Operand 1	Operand 2	Result
0 (Move)	A	0	0A
1 (Negative)	A	0	F6
1 (Negative)	0	0	00
2 (Add)	1	2	03
2 (Add)	A	F	19
3 (Subtract)	F	A	05
3 (Subtract)	A	F	FB
4 (Multiply)	3	4	0C
4 (Multiply)	F	F	E1
8 (Not)	A	0	F5
8 (Not)	0	0	FF
9 (And)	A	3	02
A (Or)	A	3	0B
B (Xor)	A	3	09
C (Shift Left)	F	0	0F
C (Shift Left)	F	3	78
C (Shift Left)	F	7	80
C (Shift Left)	F	8	00
D (Shift Right)	F	0	0F
D (Shift Right)	F	3	01
D (Shift Right)	F	4	00
D (Shift Right)	F	8	00

Requirements:

Design and implement the Arithmetic Logic Unit in logisim evolution (or logisim) using only the following components:

- Basic Gates: NOT, AND, OR, NAND, NOR, XOR, XNOR.
- Wiring: Any component in the wiring folder.

We will use a test bench to test the ALU. The test bench uses a ROM to store the test cases, a counter to pick the test case and hex displays to display the inputs and the output.

Deliverables

- The circuit file ".circ" containing the processor and all its submodules.
- An report by each team containing the following:
    - The team number and member names.
    - The work distribution (circuit and report) over the team members.
    - For each module in the project, do the following:
        - Add a captioned screenshot of the module from logisim.
        - Report the number of gates (each gate type individually, and the sum of all gates). Note that If the gate has N inputs, it is counted as N-1 gates. If the gate has 2 inputs and receives M bits on each input, then it is counted as M gates. If the gate has N inputs and receives M bits on each input, then it is counted as M*(N-1) gates.
        - Briefly describe how the module was designed.

Grade Distribution

The total grade is calculated from 20 points. It will be distributed as follows:

- Simulation: 12 points
    - No Errors: 1 point.
    - Operations: 11 points (1 point for each operation).
- Design: 5 points
- Report: 3 points.

Each team should include either 3 or 4 students. Each member will be evaluated for their contributions and understanding individually. The delivery date is Saturday May 29th 2021 11:59 PM. The discussion will be on the next day during the tutorial.

Hints

- Multiplexers, Decoders and Adders are your friends. Whenever you need them, build them in a reusable block. It is very likely that you will need to reuse them.
- Try to split the ALU into an Arithmetic Unit and a Logic Unit.

- For the shifting operations, start by creating a “shift-1” module that shifts the bits by 1 only. You can stack multiple “shift-1” together to shift with greater values, then use a multiplexer to select the stage from which the result should be retrieved.
- Sometimes, you need to reset the simulation on logisim to recover from error signals if the circuit was being modified during simulation.
- To modify the code in a ROM, right click and choose to edit or load from file. You can also save the file after loading.