Lab 9: Design an Arithmetic Logic Unit (ALU)

Goal of the laboratory is the design of an arithmetic logic unit (ALU), that can perform several operations depending on the input selection.

This are the ALU interfaces

Port	Direction	Type	Length
A	in	std_logic_vector	12
B	in	std_logic_vector	12
opcode	in	std_logic_vector	5
result	out	std_logic_vector	12
zero	out	std_logic	1
neg	out	std_logic	1
overflow	out	std_logic	1

The ALU should perform the following operations, depending on the input `opcode`.

opcode	Operation
00000	result = A + B
00010	result = A
00011	result = A + 1
01000	result = Logical Shift Left (A)
01001	result = Arithmetic Shift Right (A)
10001	result = A AND B
10010	result = NOT(A) and B

On top of that it needs to calculate values for:

- `zero`: Set to 1, if result is zero
- `neg`: Set to 1, if result is negative
- `overflow`: Set to 1, if the result requires an additional bit

A test bench is provided to test the implemented functionalities.

Exercise

1. Implement the ALU

Go to `~/labs/lab09/` and open `src/alu.vhd` with a text editor, and implement the module functionalities, following the comments in the code.

```
kate src/alu.vhd &
```

For the first exercise, consider A and B as unsigned numbers. Ignore the `overflow` output for the moment.

Once you are finish, test your code by running the simulation.

```
./run_sim.sh
```

2. Signed Operations

Add a `std_logic` port `is_signed` to the design, which signals wether the operation is between signed or unsigned numbers.

Implement also these additional operations

opcode	Operation
00001	result = A + B + 1
00101	result = A - B
00110	result = A - 1
01010	result = 0
01100	result = A * B
01101	result = A / B
01110	result = A mod B

N.B. The `neg` signal can now be high.

Check your code with the simulation script.

```
./run_sim2.sh
```

3. Overflow

Drive also the overflow output. The `overflow` bit can be calculated as following,

- Unsigned: Captures the 13th bit of the operation.
  - You need to resize the result of the operation
- Signed:
  - If adding two positive numbers, it results in a negative number

        - E.g. 0100 + 0100 = 1000 (overflow is high)
    - If adding two negative numbers, you get a positive
        - E.g. 1000 + 1000 = 0000 (overflow is high)
    - Adding a positive and a negative number cannot result in an overflow

Check the results with the simulation script

```
./run_sim3.vhd
```