Custom ALU
Deadline: 17:00 Monday 21st of November, 2022

Collaborating in small groups of up to three students is permitted, but you must implement your own programs (absolutely *do not* copy and paste from others) and provide your own answers where appropriate.

Note that lacking proper comments will lose mark.

Description

Previous exercises have used combinatorial logic to construct many of the various logic circuits that form the basis of a computer. In this exercise, we will put some of these circuits together to build an enhanced 16-bit ALU chip, with the capacity to perform more arithmetic operations and detect overflows.

The core of our enhanced ALU is represented pictorially below and consists of two 16-bit inputs, x and y, one 16-bit output out, and one 1-bit overflow flag of. It also has a series of control inputs, which can be used to select what function the ALU performs. Specifically, you need to implement the following features:

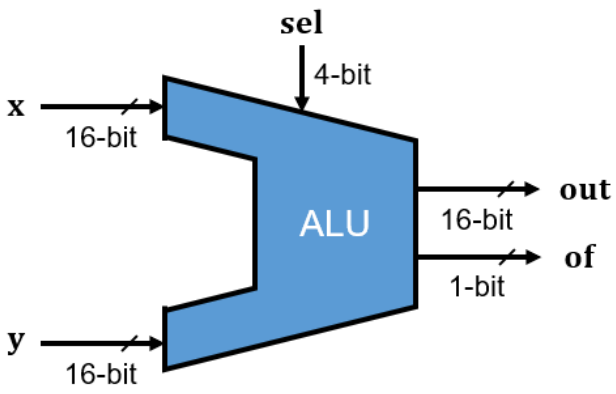

Figure 1: ALU Schematic

1. Bit-wise Logic Operations: x And y; x Or y; x Xor y. No overflow case occurs and of should always be 0.

(6 marks)

2. Shift Operations: Left shift x by 1 bit $x << 1$ (e.g. $1100, 0000, 0000, 1111 << 1 = 1000, 0000, 0001, 1110$); Right shift x by 1 bit $x >> 1$ (e.g. $1100, 0000, 0000, 1111 >> 1 = 0110, 0000, 0000, 0111$). No overflow case occurs and of should always be 0.

(4 marks)

3. Arithmetic Operations: $x + y$; $x - y$. x, y can be any signed integers using 2's complement representation. For any overflow cases, you should set of to 1 and out to -1.

(4 marks)

4. Comparison Operations: $x > y$; $x == y$. x, y can be any signed integers using 2's complement representation. The comparison result should be stored in $out[0]$, the remaining bits $out[1..15]$ should set to 0. No overflow case occurs and of should always be 0.

(4 marks)

5. Arithmetic Operation: $x \times y$. x, y can be any signed integers using 2's complement representation. For any overflow cases or inputs outside the domain, you should set of to 1 and out to -1.

(2 marks)

6. Arithmetic Operation: $x \div y$ (fraction results should **round down** to the nearest integers e.g. $7/3 = 2, 7/-3 = -3$). x, y can be any signed integers using 2's complement representation. For any overflow cases or inputs outside the domain, you should set of to 1 and out to -1.

(1 mark)

The 4-bit control signals, sel[0..3], are used to control the output produced by the ALU via selecting between the output of the 13 possible functions it can perform. Specifically, we have:

```
if (sel == 0000) set out = x And y, of = 0
if (sel == 0001) set out = x Or y, of = 0
if (sel == 0010) set out = x Xor y, of = 0
if (sel == 0011) set out = x << 1, of = 0
if (sel == 0100) set out = x >> 1, of = 0
if (sel == 0101) set out = x + y, of = 0 (no overflow)
                         = -1   , of = 1 (overflow occurs)
if (sel == 0110) set out = x - y, of = = 0 (no overflow)
                         = -1   , of = 1 (overflow occurs)
if (sel == 0111) set out = (x > y), of = 0
if (sel == 1000) set out = (x == y), of = 0
if (sel == 1001) set out = x * y, of = 0 (no overflow)
                         = -1   , of = 1 (overflow occurs)
if (sel == 1010) set out = x / y, of = 0 (no overflow)
                         = -1   , of = 1 (overflow occurs)
```

You can assume that the undefined selections (1011, 1100, 1101, 1110, 1111) will never occur in the test script. To simplify your work, you may use any built-in chips defined in nand2tetris as you wish. You can also copy and paste codes from the module materials freely. The project skeleton is supplied in the file **CUSALU.hdl**

Submission

You should zip all your files into one zip file to "Coursework1 Assignment". You should name your file as: `YOURSTUDENTID_YOURNAME`.zip. Submit your zip file onto Moodle submission page. Please note that every next submission overwrites all the files in the previous one. If you submit several times, make sure that your last submission includes all the necessary files. Include all required chips, instructions for use, and any instruction text file if necessary. For late submission, the standard late submission policy applies, i.e. 5% mark deduction for every 24 hours.