Jetic Gū

1. Handwritten submissions and proprietary formats (e.g. Pages or MS Word) **will not be graded**.
2. Mathematical expressions must be written entirely using LaTeX, otherwise **50%-100%** of marks will be deducted.
3. Circuits must be **tested**. Untested circuits will receive 0. You must show your tests with waveforms.

Submission File structure:

```
submission.zip
    — myfloat.py
    — adder16.v
    — addsub16.v
    - adder16.sim.vwf
    - addsub16.sim.vwf
```

The .vwf files are 2.5pt each, `myfloat.py` is worth 5pt.

Lab 1

1. Float point conversion. In this question, you are required to programme a custom float point converter/adder in python (5pt). Here are the instructions:
    1. Download jetic.org/dl/myfloat.py and jetic.org/dl/myfloat_test.py
    2. You should modify the `add` and `todec` methods in `myfloat.py`, such that your float class can accept any number of bits for exponent and mantissa.
    3. You can test your implementation by using the provided `myfloat_test.py`. You should add more test cases, and do not change the interface of the `MyFloat` class.
    4. Only normal float numbers will be tested, not subnormal numbers. The exponent offset is set to $-2^{e-1}+1$, where e is the number of bits for the exponent.
    5. Submit `myfloat.py` only for this question.
2. Implement a 16bit Unsigned Binary Adder (2.5pt).
    1. Create a component called `adder16`;
    2. Put input `X, Y` as 16bit buses, input `Z` as single bit; put output `S` as a 16bit bus, output `C` as single bit;
    3. Implement the addition, make sure `C` outputs the correct value;
    4. Show your component working in `adder16.sim.vwf`. You must contain at least 3 test cases, one of which much show overflow working correctly.
3. Implement a 16bit Unsigned Binary Adder-Subtractor (2.5pt).
    1. Create a component called `addsub16`;

2. Put input `X, Y` as 16bit buses, input `AS` as 1bit; put output `O` as a 16bit bus, output `C` as single bit;
3. Implement the adder-subtractor using addition and XOR. Do not use subtraction (1pt). Make sure `C` outputs the correct value;
4. Show your component working in `addersub16.sim.vwf`. Your test must contain at least 5 test cases, 2 of which much show overflow/underflow working correctly, 2 show subtraction working correctly, 2 show addition working correctly.