Week 3 assignment:
Digital Signal Processing Architectures

Introduction

The goal of the present assignment is to help you familiarize yourselve with the practical implications of choosing fixed point or floating point DSP processors for a particular problem.

First, some research and readings will be done to learn about efficiency comparisons between both data representations.

Next, simulations of both fixed point and floating point implementations of a FIR filter will be developed in Python and used to assess the accuracy differences between both approaches.

Preliminary readings:

- FIR filters:
  https://nbviewer.jupyter.org/github/unpingco/Python-for-Signal-Processing/blob/master/Filtering.ipynb
- Fixed vs. floating point representations:
  http://www.dspguide.com/ch28/4.htm
- Optimized fixed point FIR filters:
  https://nl.mathworks.com/help/dsp/examples/optimized-fixed-point-fir-filters.html
- Low-pass FIR filter example
  (provided in `fir_filter.py` file)

Research Questions

1. Describe the main characteristics of fixed and floating point data representations, and discuss advantages and disadvantages of each in the context of DSP algorithm implementations.
2. What are the most typical number representation settings in modern DSPs? Choose two fixed-point and two floating-point cases, and detail number structure, number of bits, represented range and modes.

Lab assignment:
Fixed point vs. Floating point accuracy comparison

Requirements

- Python 2.7 or newer
- matplotlib, scipy.signal
  it is strongly recommended to use a scientific Python distribution like **Anaconda**, where all necessary libraries come pre-installed (see http://docs.continuum.io/anaconda/install.html)
- **pyfixedpoint** library (linux instructions follow):

```
wget http://downloads.sourceforge.net/pyfixedpoint/spfpm-0.5.tar.gz
tar xzvf spfpm-0.5.tar.gz
cd spfpm-0.5
python setup.py install
python
>>> import FixedPoint
>>> help(FixedPoint)
```

Development

1. Implement a parameterizable FIR filter object, providing:
   - an `__init__` method that allows to specify:
     - fixed or floating point data representation,
       with their corresponding bit length parameters
       (you will have to use **pyfixedpoint** to limit the precision to the chosen representation)
     - an array of filter weights
   - an `apply` method that takes as input an arbitrary long array of samples (longer that the array of weights) and outputs the result of applying the FIR filter to it.
2. We will use the 4 floating point configurations obtained in research question #2 and run simulations of FIR filter under those settings.
   You have to reuse the lowpass FIR-filter provided in `fir_filter.py` and use your implementation to ensure that the required data representations are used. Generate a random input of 300 samples and use the same for all experiments.
   Create an additional FIR filter with no data restrictions (arbitrary precision). We will use it to measure the error of the other implementations.
3. For each of the created FIR filters, generate an array showing the relative error of each output sample when compared to the arbitrary precision case.
4. Using the data from the previous step, generate error plots using matplotlib.
5. Write a brief report (between 3 and 5 pages long), including the following:
   a. Answers to research questions
   b. Any design decisions taken when designing the simulation
   c. All the required plots
   d. Conclusions on the observed error behaviour of different implementations of FIR filters.